Managing Complexity in Large-Scale Networks via Flow and Network Aggregation

by

Michael Charles Montgomery, B.S., M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 1998

Managing Complexity in Large-Scale Networks via Flow and Network Aggregation

Approved by
Dissertation Committee:

_____

_____

_____

_____

_____

Acknowledgments

I would like to thank my supervisor, Dr. Gustavo de Veciana, for his invaluable advice and guidance throughout my tenure as a graduate student at the University of Texas at Austin. I would also like to thank Dr. Aristotle Arapostathis, Dr. Vijay Garg, Dr. San-qi Li, and Dr. Harrick Vin for serving on my committee.

Most of all, I am grateful to Jesus Christ, my Lord and Savior, and to my wife Heather for her patience, prayers, and enduring support.

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship and a Du Pont Graduate Fellowship in Electrical Engineering.

MICHAEL CHARLES MONTGOMERY

The University of Texas at Austin
August 1998

Managing Complexity in Large-Scale Networks via Flow and Network Aggregation

Publication No. _____

Michael Charles Montgomery, Ph.D.
The University of Texas at Austin, 1998

Supervisor: Gustavo de Veciana

Two well-known approaches to reducing complexity in large-scale communication networks are flow and network aggregation. Flow aggregation bundles together a group of individual flows and jointly manages and switches them inside a network or subnetwork. Network aggregation hierarchically groups network elements into subnetworks and approximately represents each subnetwork's state in a compact form which reduces the overheads of information exchange in traffic management algorithms.

In the flow aggregation area, we first explore the benefits of aggregating multicast demands on Virtual Path (VP) trees. We show that this can effectively reduce capacity requirements, balance network loads, and reduce the number of VP trees required. Real networks have time-varying demands and finite signaling resources, so we develop adaptive resource allocation algorithms for aggregated flows subject to such constraints.

In some cases it is desirable to modify the manner in which flows are aggregated (the VP layout), so we investigate algorithms to migrate from one layout to another. For incremental changes, the potential for performance losses during migration in terms of call blocking is minimal. However, when dramatic changes in the VP layout are warranted, it is desirable to enhance performance by implementing a simple decentralized algorithm that we have proposed.

In the network aggregation area, we develop an implicit representation of the congestion level of a subnetwork which is based on a distributed computation of the average implied cost to go through or into the subnetwork. We prove that both a synchronous and asynchronous computation of the implied costs will converge to a unique solution under a light load condition, and an alternative, more aggressive approximation based on additional local averaging will converge under any traffic conditions subject to sufficient damping. Our experiments show that our costs are indeed quite accurate.

Based on this representation for congestion, we propose a Quality of Service-sensitive routing algorithm that is able to appropriately route high-level flows while significantly reducing complexity. The algorithm uses effective bandwidths to capture traffic behavior, and it adaptively selects hierarchical routes so as to maximize network revenue, while allowing low-level dynamic routing within subnetworks to respond to traffic fluctuations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Aggregation, the collecting of units or parts into a mass or whole, is a well-known technique for managing "complexity" in many types of large-scale systems [17].[1] The aim of this dissertation is to advance the state of the art in applying aggregation to large-scale communication networks in two domains: the aggregation of network flows and network elements.

With flow aggregation, a group of individual flows is bundled together and jointly managed and switched inside a network or subnetwork. By carefully aggregating flows, we can allocate resources with tolerable losses in efficiency while reaping such benefits as reduced call processing loads and simplified Quality of Service (QoS) provisioning. However, because traffic demands fluctuate and network resources are limited, resource allocations may need to be adapted over time. More extreme changes in the demands may warrant migration to a new arrangement of aggregated flows.

Network aggregation hierarchically groups network elements into subnetworks and represents each subnetwork's state in a compact form. This is done to reduce overheads in controlling and managing the network traffic. Having a representation which captures the congestion level of a subnetwork while taking traffic interdependencies into account is useful both to network operators and to hierarchical routing algorithms.

Aggregation does have a downside: a possible loss in efficiency and/or accuracy. We might ask if the growth in the complexity of communication networks

[1] We will use the term complexity to loosely discuss a variety of problems related to the number of flows and/or size of future large-scale networks.

really warrants its use? While the various players in the telecommunications business are competing to determine how to realize the promises of the "information superhighway," there is no doubt that the network will be geographically widespread and large in capacity. As a reference case, the growth of the Internet is astounding. It has grown from 1.3 million to 30 million hosts in the past five years[2] with the number of users estimated at 10 times the number of hosts. The telephone network has been growing steadily throughout the century, although never at the pace of the Internet. Currently, there are roughly 700 million terminations worldwide, and AT&T handles 210 million voice, data, and image calls per business day. Cable TV is another rising industry. As of 1996, Time Warner had 12.3 million subscribers with a subscription rate of 65% for passed homes, an increase of 5 million subscribers in just two years. These trends suggest that the sheer number of users and heterogeneity of services are likely to dramatically increase the complexity of the communication infrastructure.

As networks grow larger in capacity and extent, companies have embraced the vision of a broadband network which can use the same infrastructure to cost effectively multiplex all kinds of traffic including voice, video, and data. Such multiservice networks are typically modeled as multirate circuit-switched networks at the call level [19, 26, 52, 63], but is this a good model for the future? The answer centers around whether future broadband networks will have characteristics which are akin to "connection-oriented" networks which in turn depends on the roles that resource reservation, call admission, routing, priority schemes, and pricing will play [65]. We will speculate briefly on this below after discussing the impact of a connection-oriented environment on network performance and control.

Today we have both connection-oriented networks, such as the telephone network, and connectionless networks, such as the Internet. One particularly important difference between connectionless and connection-oriented networks is how congestion might develop and dissipate. Consider a single-packet message sent into a connectionless network providing only best-effort service (e.g., the Internet). If the packet encounters congestion at a particular location, it may be routed around the congestion. Thus, hopefully, the congestion will dissipate locally. If enough packets continue to be sent toward that location, the congestion could spread slowly outward from that point, eventually causing "global" congestion. Now consider the other extreme: a call is set up in a circuit-switched network with resources reserved

[2]Source: Network Wizards (www.nw.com).

along the chosen route for the call's lifetime. Instead of queueing as in the previous case, congestion here results in call blocking. If a call is blocked due to congestion at a particular location, then an alternate, typically longer route may be selected, and resources would be reserved accordingly. The problem here is that if congestion persists, it can spread very quickly into global congestion due to the interactions between overlapping routes. In other words, due to the network-wide interactions among routes, there is a greater potential for "knock-on" effects where local congestion propagates to other parts of the network [36]. Thus traffic management algorithms for connection-oriented networks typically take a global view of network performance in order to make good routing decisions. Of central importance is the notion of an *implied cost* [36] for a connection along a given path which measures the opportunity cost or expected loss of revenue resulting from accepting a connection. Using implied costs, we can quantify the potential for "knock-on" effects in routing and capacity allocation decisions.

In an integrated network supporting multiple types of traffic with multiple Qualities of Service, it is almost certain that at least some classes of applications will desire guaranteed QoS in the form of an explicit or implicit minimal rate guarantee. The question is how will this QoS be provided and will the network indeed look connection-oriented? The Internet today provides only best-effort service, and application throughput can drop to essentially zero during times of extreme congestion. This is acceptable for "elastic" applications [65], but not for applications where the performance degradation is unacceptable below a certain minimum rate, e.g. Internet telephony. Barring the viability of a solution based on overprovisioning, some type of resource reservation and call admission mechanism will be needed. One perhaps overzealous mechanism is used in Asynchronous Transfer Mode (ATM) networks where the route and resource reservations are established at connection setup according to the desired QoS and held for a connection's lifetime. The resource ReSerVation Protocol (RSVP) [11, 77], a proposed QoS extension to the Internet, is an intermediate approach between the "hard" resource reservations in ATM and the nonexistent resource reservations in the current Internet.[3] In RSVP, packet flows reserve resources temporarily at routers along the "current" path from source to destination, but these reservations are "soft" in the sense that they will time out if not renewed, thus allowing flows to be rerouted (or "connec-

[3]Alternatively, resource reservations in the current Internet can be thought of as occurring implicitly for a packet right before being served at a router.

tions" repacked [38]) as necessary. Combinations of the above approaches exist in various proposals for an IP over ATM switching environment, where IP flows are mapped to ATM virtual circuits [13, 28]. We conclude that if a resource reservation mechanism for network flows is eventually established and/or the route for a flow changes infrequently, the network will look connection-oriented from a system perspective, even if the underlying communication layer is connectionless. Based on this assumption, we rely heavily on models for circuit-switched networks and the computation of implied costs throughout this work.

In light of the above, our goal is to find workable tradeoffs between efficiency, QoS, and "complexity." The first method we explore for reducing complexity is flow aggregation. One widely known example of flow aggregation is the use of Virtual Paths (VPs) in ATM networks which can carry multiple Virtual Circuits (VCs). VPs are especially applicable in large-scale networks, and although efficiency may be compromised, they can be an effective means for reducing call processing loads as well as simplifying connection admission control, routing, and provisioning of QoS requirements.

In Chapter 2, we consider the aggregation of multicast demands on VP trees. We propose a pre- or post-processing step to the VP multicast layout problem, which either reduces the complexity of the required optimization or further improves upon obtained solutions. This is an important topic because multicast applications, such as videoconferencing, will likely generate a substantial portion of future network traffic.

To handle changing demands on networks with limited resources, such as signaling capacity, bandwidth, buffers, etc., we need to be able to adjust the resources allocated to aggregated flows. Chapter 3 begins by arguing that signaling resources may not be sufficient for future demands on ATM networks. The key issue is how to achieve a tradeoff among the scarce resources in future networks. With this in mind, we present a framework for adapting VP capacities in a variety of settings, and we develop and evaluate algorithms for migrating from one set of VP capacities to another.

In Chapter 4, we consider network aggregation, our second method for reducing complexity in large-scale networks. By exploiting network aggregation, we can significantly reduce the overheads of information exchange needed to form an approximate global view of the entire network state. We present a novel application of implied costs to implicitly represent the state of a subnetwork. The key

element of our scheme is a decentralized computation of the average implied cost to go through or into a subnetwork which reflects the congestion in the subnetwork as well as the interdependencies among traffic streams in the network. We incorporate this implicit state representation into a QoS-sensitive routing algorithm that adaptively selects hierarchical routes so as to maximize network revenue.

Finally, Chapter 5 concludes with a summary of our main results, a discussion of applying our methods to areas outside of ATM, and a list of future research directions.

Chapter 2

Flow Aggregation: Aggregating Multicast Demands on VP Trees

2.1 Introduction

In ATM networks, Virtual Paths (VPs) that can carry multiple Virtual Circuits (VCs) are allocated to reduce the complexity of call setup and traffic management at the possible expense of efficiency. A VP layout consists of a vector of capacities allocated to VPs that are set up on a subset of network routes on a permanent or semi-permanent basis. This logical partitioning may be done periodically for the purpose of adaptive resource allocation due to changing network conditions.

Given a set of multicast demands for a network incorporating multicast-capable switches, a layout of VP trees could be established using an algorithm such as the one found in [42]. Setting up an Switched Virtual Circuit (SVC) tree is a relatively expensive operation, so even with low multicast demands, creating and using VP trees on a slower time scale than connection holding times can be worthwhile. In this chapter we argue that due to statistical multiplexing, one may actually save capacity by aggregating heterogenous multicast demands on the same VP tree. Taking advantage of this fact, we propose a pre- or post-processing step to the VP multicast layout problem, which either reduces the complexity of the required optimization or further improves upon obtained solutions. If the VP multicast layout is already determined, then our procedure could be a post-processing step; if not, it would be a pre-processing step which is discussed further in Sec. 2.5. The need

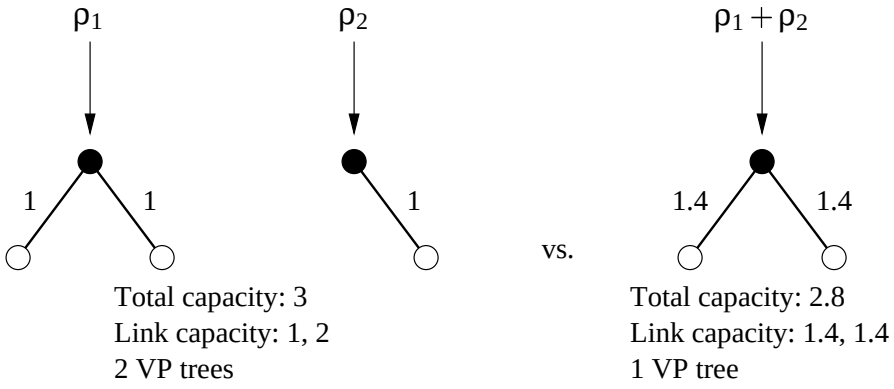


Figure 2.1: In this example, aggregating the multicast demands provides capacity savings, better load balancing, and a reduction in VP setup and management loads.

for aggregating multicast demands onto VP or VC trees to avoid VP/VC explosion has previously been suggested in the context of an IP over ATM environment [10]. Suppose we have a destination set of size 10. Then there are 1023 possible non-empty subsets of destinations, and it is likely to be impractical to set up a separate tree for each subset with nonzero demand.

Consider the situation illustrated in Fig. 2.1. There is a single source and two destinations with a demand ρ_1 for multicast connections to both destinations and a demand ρ_2 for unicast connections to only one destination. Suppose that a capacity of 1 on each link is able to accommodate the demands at the desired call blocking probability. Furthermore, suppose that if the demands are aggregated, a capacity of 1.4 on each link is required. In this case, despite the fact that connections of type 2 are needlessly using both links, we obtain benefits from aggregating the multicast demands in three areas: the total required capacity is less, the link capacities are more evenly balanced, and one VP tree, rather than two, is required.

More generally, suppose we are given demands ρ_1 and ρ_2 for multicast connections requiring unit bandwidth from a common source to destination sets $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively, where $\mathcal{D}_2 \subseteq \mathcal{D}_1$.[1] Assuming the network switches have the proper multicast capabilities, we want to establish VP trees for sets $\mathcal{D}_1$ and $\mathcal{D}_2$. The question is whether we should establish two separate trees or a single tree to ac-

[1]Of course, one destination set does not have to be a subset of another, but this case leads to the simplest algorithms and the greatest potential savings. In the more general case, a shared VP tree would have to reach destinations in the smallest set containing both $\mathcal{D}_1$ and $\mathcal{D}_2$.

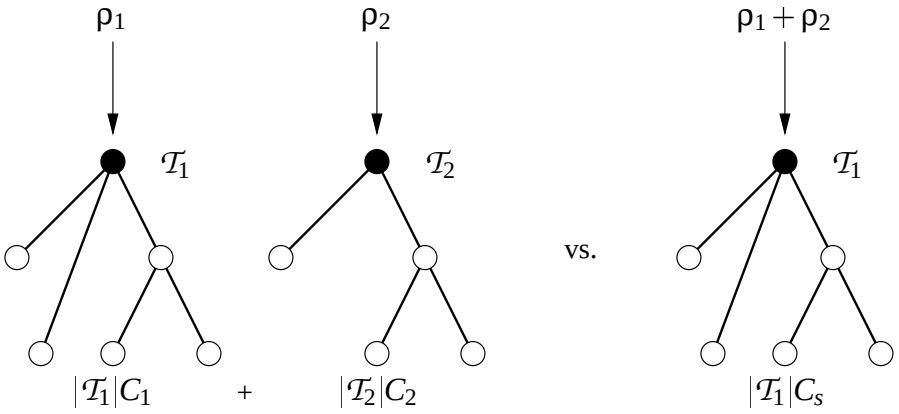


Figure 2.2: Establishing separate VP trees to accommodate demands ρ_1 and ρ_2 would require a total capacity of $|\mathcal{T}_1|C_1 + |\mathcal{T}_2|C_2$. Sharing the larger tree would require capacity $|\mathcal{T}_1|C_s$.

commodate the demands ρ_1 and ρ_2, i.e., will the benefit of additional multiplexing at the call and cell levels outweigh the bandwidth wasted by connections for the smaller set $\mathcal{D}_2$ using the larger tree? This situation is illustrated in Fig. 2.2.

To answer this question, we first introduce the function $\alpha(\rho, B)$ which gives the link capacity needed to accommodate the demand ρ at a specified call blocking probability B. This function may account for statistical multiplexing at the call level, the burst level, the cell level, or some combination of the three. To determine the capacity needed for the VPs in each case, we solve for $C_1 = \alpha(\rho_1, B)$, $C_2 = \alpha(\rho_2, B)$, and $C_s = \alpha(\rho_1 + \rho_2, B)$. For separate VP trees $\mathcal{T}_1$ and $\mathcal{T}_2$, the total capacity needed is $C = |\mathcal{T}_1|C_1 + |\mathcal{T}_2|C_2$, where $|\mathcal{T}|$ is the number of links in multicast tree $\mathcal{T}$. When sharing tree $\mathcal{T}_1$, the total capacity needed is $C' = |\mathcal{T}_1|C_s$. If $C' < C$, it would be beneficial to use a single tree. We can rewrite this condition as

$$\frac{C_s - C_1}{C_2} < \frac{|\mathcal{T}_2|}{|\mathcal{T}_1|}. \tag{2.1}$$

It should be noted that there are additional benefits to sharing VP trees besides capacity savings: e.g.,

- savings in Virtual Path Identifier (VPI) usage — sharing VP trees would reduce the number of VPs, and hence VPIs, needed for a particular layout,
- a reduction in VP setup and management loads as well as setup delays for a connection,

- a more even balancing of load across the network — sharing VP trees tends to reduce the variance in the allocated link capacities, and
- possible savings in the size of buffers needed at the input of each VP tree due to the increased cell level multiplexing from combining demands.

After indicating the related work in this area in Sec. 2.2, we explore the use of the Erlang B formula to implicitly determine the function α in Sec. 2.3, and then further consider a Gaussian traffic model. Given an initial collection of destination sets, heuristics for finding an aggregation of demands requiring the least total capacity are proposed and evaluated through simulation in Sec. 2.4. Finally, in Sec. 2.5, we present methods for dealing with unknown topologies, and Sec. 2.6 concludes the chapter with a few additional comments.

2.2 Related work

Flow aggregation using virtual paths is a key component of the traffic management specification for ATM networks [15, 72]. Although much research has addressed either the VP layout problem or the multicast routing problem separately, e.g. [1, 3, 7, 12, 23, 58, 75], to our knowledge Kim is the only one to specifically address the VP multicast layout problem [42]. The aggregation of multicast demands which we propose in this chapter builds on his work on the VP multicast layout problem by either reducing the complexity of the required optimization or further improving upon an obtained solution. The need for such aggregation has been suggested in the context of an IP over ATM environment [10], but we found no specific research on this topic in the literature. Part of the material in this chapter has been previously presented [54].

2.3 Specific examples

Herein, we assume a large population (infinite sources) model where the requests for multicast connections arrive as Poisson processes.[2] In this case, the Erlang B formula can be used to implicitly determine the function α and assess the benefits of

[2]Note that aggregating demands and sharing a tree will increase the population of sources requesting access to the tree and will serve to strengthen this assumption.

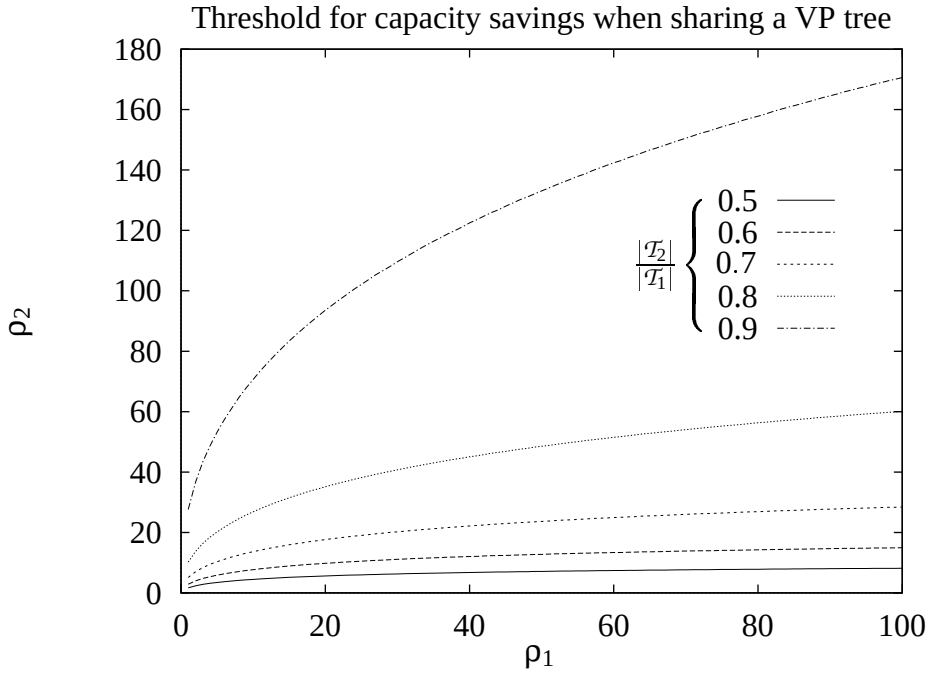


Figure 2.3: For a blocking probability of 10^{-3}, the threshold for capacity savings is plotted for values of $|\mathcal{T}_2|/|\mathcal{T}_1|$ ranging from 0.5 to 0.9. Below each line, the capacity savings is greater than zero.

call level multiplexing alone [70]. For the above setup, we must solve $E(\rho_1, C_1) = E(\rho_2, C_2) = E(\rho_1 + \rho_2, C_s) = B$ for C_1, C_2, and C_s, and then test the condition for sharing the larger tree as expressed in (2.1). Based on the Erlang function and a blocking probability of 10^{-3}, Fig. 2.3 shows the threshold for capacity savings (maximum value of ρ_2 for a given ρ_1) as $|\mathcal{T}_2|/|\mathcal{T}_1|$ is varied from 0.5 to 0.9. Below each line, the capacity savings is greater than zero. Note that, because of the sub-additivity property explained below, the threshold for $|\mathcal{T}_2|/|\mathcal{T}_1| = 1$ would be a vertical line at $\rho_1 = 0$.

In Fig. 2.4, we see that for a constant blocking probability B, the function $C = g(\rho)$, defined implicitly by Erlang's formula, is concave and sub-additive [70]. The sub-additivity property, i.e., $g(\rho_1 + \rho_2) < g(\rho_1) + g(\rho_2)$, implies that two separate links require more capacity than a single link with the same total traffic, or in other words, it assesses the benefits of call level multiplexing. Furthermore, as the desired call blocking probability is decreased, the multiplexing benefits get better.

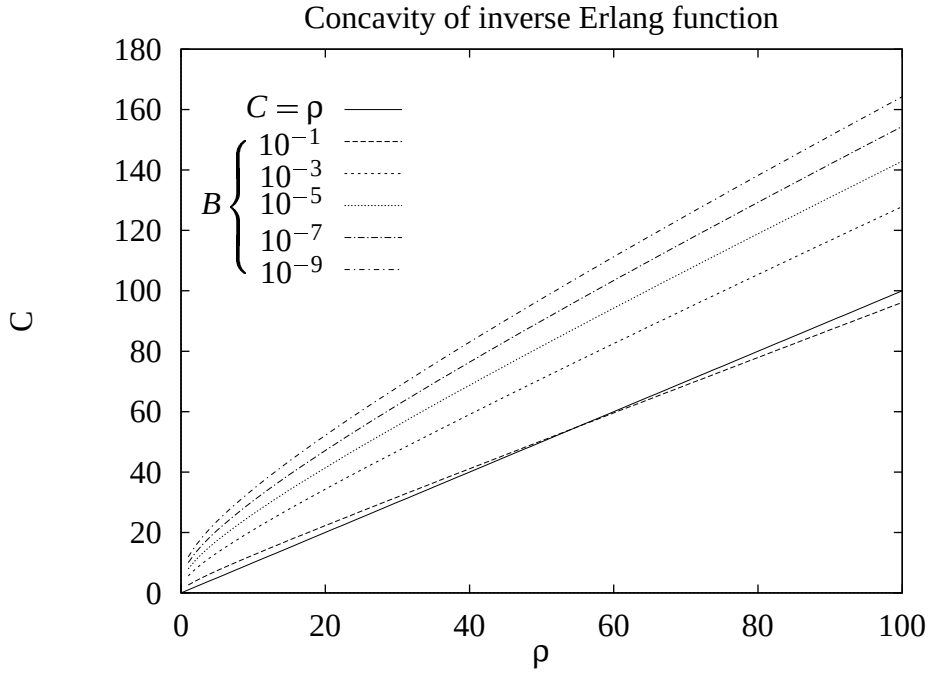


Figure 2.4: For a given demand ρ, the capacity needed to achieve blocking probabilities ranging from 10^{-1} to 10^{-9} is plotted. The line $C = \rho$ is also plotted as a comparison.

However, even for a modest blocking probability of 10^{-3}, we can achieve significant capacity savings from aggregation.

The potential capacity savings are graphed for some different scenarios in Figs. 2.5 to 2.9 for a call level blocking probability of 10^{-3}. Figs. 2.5, 2.6, and 2.7 exhibit the potential savings in capacity when aggregating 2 multicast groups. The savings are given by $100(C - C')/C$, where C and C' are the total required capacities for separate VP trees and a shared VP tree, respectively, so the values shown are relative savings percentages. Fig. 2.5 shows results for the situation shown in Fig. 2.2 where $|\mathcal{T}_1| = 5$ and $|\mathcal{T}_2| = 4$. The tree sizes are varied in the remaining figures. In Figs. 2.8 and 2.9, there are m multicast groups with $\mathcal{D}_1$ being the largest destination set, and the savings graphed in the figures are a comparison between using m separate VP trees and aggregating all m groups onto a single tree $\mathcal{T}_1$.

Figs. 2.5, 2.6, 2.7, and 2.9 show that low values of $\rho_1, \rho_2, \ldots, \rho_m$ lead to

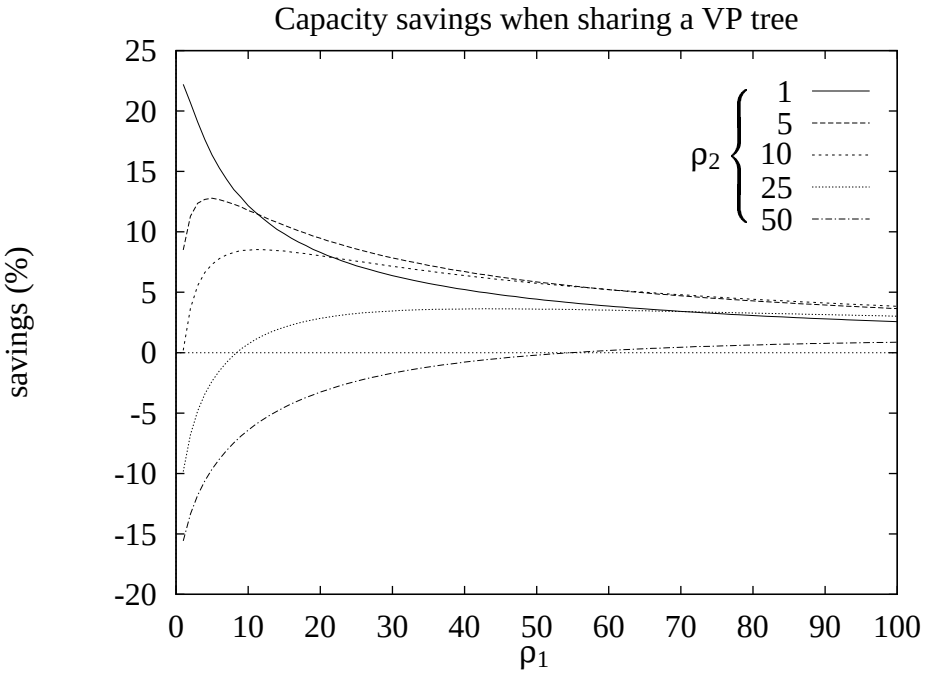


Figure 2.5: The capacity savings shown are for sharing VP tree $\mathcal{T}_1$ between 2 multicast groups with offered loads ρ_1 and ρ_2, fixed tree sizes $|\mathcal{T}_1| = 5$ and $|\mathcal{T}_2| = 4$, and a blocking probability $B = 0.001$.

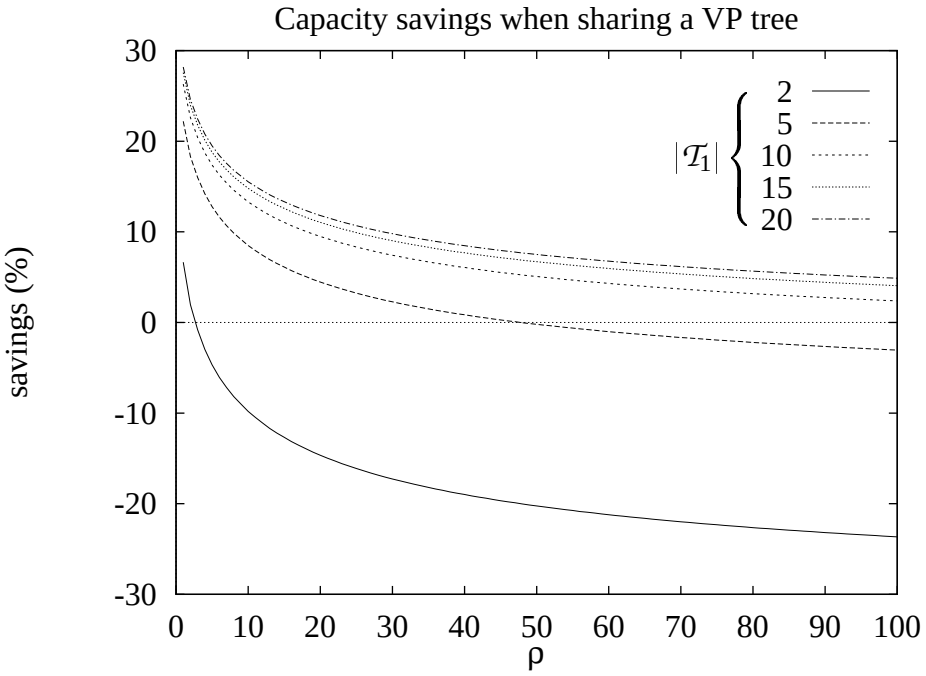


Figure 2.6: The capacity savings shown are for sharing VP tree $\mathcal{T}_1$ between 2 multicast groups with offered loads $\rho_1 = \rho_2 = \rho$, a smaller tree size $|\mathcal{T}_2| = |\mathcal{T}_1| - 1$, and a blocking probability $B = 0.001$.

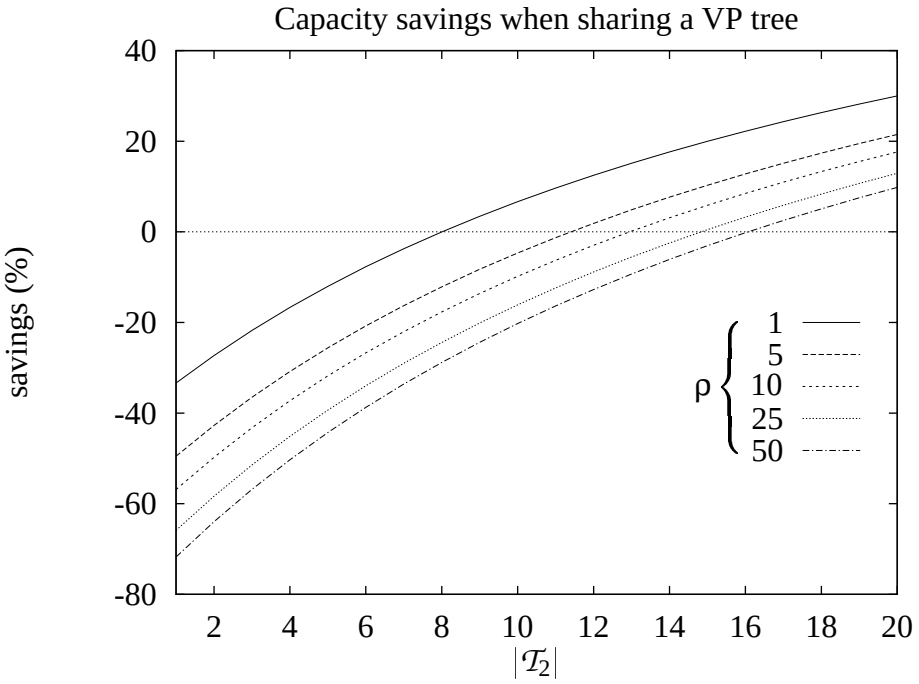


Figure 2.7: The capacity savings shown are for sharing VP tree $\mathcal{T}_1$ between 2 multicast groups with offered loads $\rho_1 = \rho_2 = \rho$, a larger tree size $|\mathcal{T}_1| = 20$, and a blocking probability $B = 0.001$.

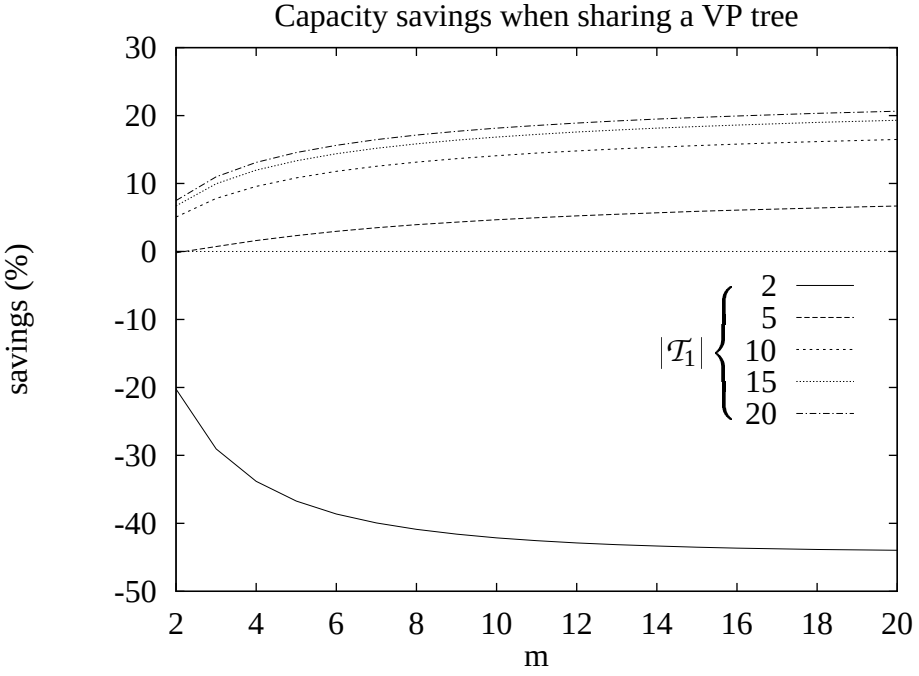


Figure 2.8: The capacity savings shown are for sharing VP tree $\mathcal{T}_1$ between m multicast groups with destination sets $\mathcal{D}_i \subseteq \mathcal{D}_1$ and tree sizes $|\mathcal{T}_i| = |\mathcal{T}_1| - 1$ for groups $i = 2, 3, \ldots, m$, offered loads $\rho_1 = \rho_2 = \cdots = \rho_m = 50$, and a blocking probability $B = 0.001$.

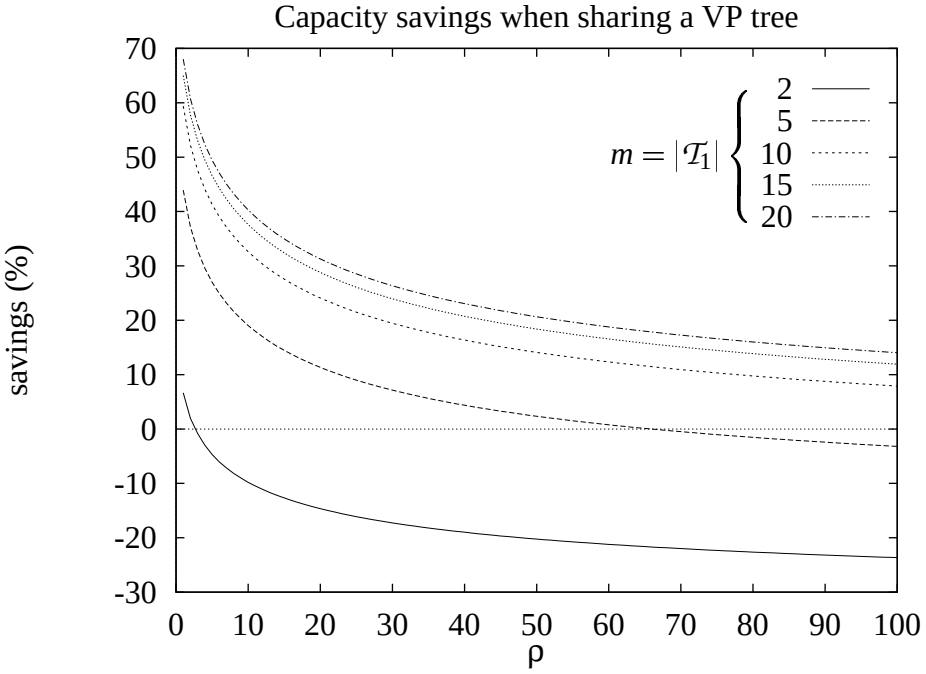


Figure 2.9: The capacity savings shown are for sharing VP tree $\mathcal{T}_1$ between m multicast groups ($m = |\mathcal{T}_1|$) with destination sets $\mathcal{D}_i \subseteq \mathcal{D}_1$ and tree sizes $|\mathcal{T}_i| = |\mathcal{T}_1| - 1$ for groups $i = 2, 3, \ldots, m$, offered loads $\rho_1 = \rho_2 = \cdots = \rho_m = \rho$, and a blocking probability $B = 0.001$.

higher savings. This is due to the fact that the savings percentage from the sub-additivity of the inverse Erlang function, $C = g(\rho)$, introduced above is greater for smaller values of ρ (see Fig. 2.4). Note that the low offered load regime is not unrealistic in practice. Indeed, currently the utilizations achieved on ATM/SONET links are as low as 10–20% due to overheads and spare capacity reserved for 1+1 failure protection [66]. This would mean that a 10 Gbps link would have around 2 Gbps of usable capacity. When this capacity is further partitioned into a logical VP layout, with say 100 VPs, then each VP would have a capacity of 20 Mbps. Each VP could, for instance, accommodate 20 video connections at 1 Mbps, or in other words, the VP capacity would be $C = 20$ circuits.

Aggregation of larger trees also leads to higher savings, as can be seen in Figs. 2.6, 2.7, 2.8, and 2.9, because the wasted bandwidth due to a call for a smaller set of destinations using the larger tree becomes less relative to the total required bandwidth. The incremental gain in savings decreases as the tree sizes grow larger because asymptotically, for fixed offered loads, the savings approaches a constant. For example, in Fig. 2.6, the savings for a given ρ and B is $(|\mathcal{T}_1|C_1 + (|\mathcal{T}_1| - 1)C_2 - |\mathcal{T}_1|C_s)/(|\mathcal{T}_1|C_1 + (|\mathcal{T}_1| - 1)C_2)$ where $C_1 = C_2 = \alpha(\rho, B)$, $C_s = \alpha(2\rho, B)$, and α is the inverse Erlang function. As $|\mathcal{T}_1| \to \infty$, the savings approaches $1 - \big(C_s/(C_1 + C_2)\big)$, i.e., the savings that arises from combining trees of the same size.

From Fig. 2.8, we see that aggregation of more and more trees (increasing m) increases savings, but it tapers off. In fact, as m approaches infinity, the savings approaches a constant for fixed offered loads, fixed tree sizes, and constant blocking probability. For the Erlang function, if the offered load and capacity are scaled proportionally, we have that, for $\rho > C$ (heavy traffic), [35]

$$E(m\rho, mC) \xrightarrow{m\to\infty} 1 - \frac{C}{\rho}. \tag{2.2}$$

Our scaling is not linear in both ρ and C because as the number of groups (m) grows, the offered load to the shared tree is scaled linearly but the capacity only grows enough to keep the blocking probability constant (see Fig. 2.4). However, for large enough m, we are indeed in the heavy traffic or overloaded regime because the blocking probability must remain greater than zero and in the critical and underloaded regimes the blocking probability goes to zero as the capacity grows large [51]. Therefore, letting $\alpha(\rho, B)$ represent the inverse Erlang function, we can use (2.2) to obtain the rough approximation $\alpha(m\rho, B) \approx m\rho(1 - B)$ for large m. Letting $\rho = \rho_i =$ constant, the savings in Fig. 2.8 is $1 - \big(|\mathcal{T}_1|\alpha(m\rho, B)\big)/\big(|\mathcal{T}_1|\alpha(\rho, B) +$

$\big(|\mathcal{T}_1|-1\big)(m-1)\alpha(\rho,B)\big)$ which asymptotically, as $m\to\infty$, becomes

$$1-\frac{|\mathcal{T}_1|\rho(1-B)}{(|\mathcal{T}_1|-1)\alpha(\rho,B)}.$$

Now suppose we combine the last two situations and scale both m and $|\mathcal{T}_1|$ as in Fig. 2.9. Using the above approximation, as $m=|\mathcal{T}_1|$ approaches infinity, the savings for fixed $\rho=\rho_i$ is

$$1-\frac{\rho(1-B)}{\alpha(\rho,B)}. \tag{2.3}$$

The second term in (2.3) can be interpreted as the inverse of the bandwidth required per connection. The bandwidth per connection increases as ρ decreases because of less multiplexing, so for smaller ρ, the inverse of the bandwidth per connection is smaller which leads to higher potential savings. As an example, for $B=0.001$ and $\rho=10$, the limit in (2.3) implies a maximum savings of 52.1%; for $\rho=100$ the maximum savings is 21.9%. These values are quite reasonable in light of Fig. 2.9, where for $m=|\mathcal{T}_1|=20$ and $\rho=10$ we have a savings of 40.3%, and for $\rho=100$ the savings is 14.0%.

Finally, we see from Fig. 2.7 that there is a threshold for the smaller tree size below which it never pays to aggregate two candidate trees.

Although the benefits of call level multiplexing are significant, even more capacity savings can be exhibited by incorporating a burst or cell level model. As a simple example, we could model the cell arrival rate of each call by a Gaussian random variable with mean λ and variance σ^2. For a bufferless link with N ongoing connections, it can be shown that the capacity requirement is roughly given by

$$\beta(N)=N\lambda+k\sqrt{N\sigma^2}, \tag{2.4}$$

where k is a QoS parameter determined by the desired cell loss probability (see e.g. [69]). For instance, a cell loss probability of 10^{-6} would require $k=4.7534$. Letting $\alpha(\rho,B)$ represent the inverse Erlang function, we need to allocate $\beta(\alpha(\rho,B))$ to satisfy the call level and cell level QoS requirements for fixed cell loss probability and traffic parameters (λ,σ^2). We will explore the impact of the Gaussian traffic model in the simulations of Sec. 2.4.

2.4 Heuristics

We now consider the more general problem of having multicast demands from a common source to m destination sets with $\mathcal{D}_i \subset \mathcal{D}_1$ for $i = 2, 3, \ldots, m$. Let $N = |\mathcal{D}_1|$. The subsets $\mathcal{D}_i$, $i = 2, 3, \ldots, m$, can have from 1 to $N-1$ elements. First, we assume that $\mathcal{D}_m \subset \mathcal{D}_{m-1} \subset \cdots \subset \mathcal{D}_1$, and later we will drop this assumption.

If each set has a known VP tree $\mathcal{T}_i$, the brute force approach to finding a grouping of sets with the least total required capacity is simply to try all possible combinations and keep the best combination. The m sets can be divided into 1 to m groups. The number of possible ways to divide them into k groups is equivalent to finding the number of ways to place m distinguishable balls into k indistinguishable cells such that no cell is empty. This is given by a Stirling number [60] of the second kind $S(m,k)$ where

$$S(m,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i\binom{k}{i}(k-i)^m.$$

For each combination, we must find the capacity needed by the k groups, so the computational complexity is proportional to $\sum_{k=1}^{m} kS(m,k)$which grows exponentially. For example, for $m = 2$, $\sum_{k=1}^{m} kS(m,k) = 3$, for $m = 4$, it is 37, and for $m = 8$, it is 17,007. Thus, for reasonably large m, heuristics with polynomial complexity would be preferable to the exponential complexity of the brute force approach.

Although we cannot guarantee optimality, it seems reasonable to try combining pairs of sets starting from the largest set $\mathcal{D}_1$, as suggested by the observation made in Sec. 2.3 that aggregation of larger trees leads to higher savings. Furthermore, we expect to get the most significant savings from combining trees which are close together in size. These ideas lead to the following (clustering) algorithm with complexity $O(m)$. We first describe the algorithm in words and then give pseudocode for the algorithm. In the pseudocode, the function $combine(i, j)$ combines the demands for $\mathcal{D}_i$ and $\mathcal{D}_j$ into a single demand for $\mathcal{D}_i$ (i.e., $\rho_i := \rho_i + \rho_j$; $\rho_j := 0$;) and returns true if the condition in (2.1) holds. Otherwise, it does nothing and returns false.

Algorithm 2.1. *Starting with the largest destination set $\mathcal{D}_1$, combine the demands for $\mathcal{D}_1$ and $\mathcal{D}_2$ into a single demand for $\mathcal{D}_1$ if the condition in* (2.1) *holds. If successful, try to combine demands for $\mathcal{D}_1$ and $\mathcal{D}_3$, and continue on until unsuccessful*

with candidates $\mathcal{D}_1$ and $\mathcal{D}_i$, $2 \leq i \leq m$. If $i < m$, repeat the procedure starting with $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$, continuing on until reaching $\mathcal{D}_m$.

```
1  begin
2     i := 1; j := 2;
3     while j ≤ m do
4           if ¬combine(i, j) then i := j; endif;
5           j := j + 1;
6     endwhile;
7  end
```

Note that each combination made would result in capacity savings, so regardless of the final grouping, the procedure would be worthwhile, but not necessarily optimal. As a comparison to Algorithm 2.1, we also propose the following algorithm which attempts to make as many combinations as possible with the current destination set before moving on, resulting in an algorithm of complexity $O(m^2)$:

Algorithm 2.2. *Starting with the largest destination set $\mathcal{D}_1$, combine the demands for $\mathcal{D}_1$ and $\mathcal{D}_2$ into a single demand for $\mathcal{D}_1$ if the condition in* (2.1) *holds. Next try to combine demands for $\mathcal{D}_1$ and $\mathcal{D}_3$, and continue on with candidates $\mathcal{D}_1$ and $\mathcal{D}_i$ until $i = m$. If a successful combination was made, repeat the procedure starting with $\mathcal{D}_1$ and $\mathcal{D}_2$ (skipping sets which have previously been absorbed). When a pass without a successful combination has been made, move from $\mathcal{D}_1$ to the next smallest set $\mathcal{D}_j$ that has not been aggregated and repeat from the beginning starting with $\mathcal{D}_j$ and $\mathcal{D}_{j+1}$.*

```
1  begin
2     for i := 1 to m − 1 step 1 do
3        if ρ_i > 0 then                    (the demand for D_i is nonzero)
4           k := 1;
5           while k > 0 do
6                 k := 0;
7                 for j := i + 1 to m step 1 do
8                    if ρ_j > 0 ∧ combine(i, j) then k := k + 1; endif;
9                 endfor;
10          endwhile;
11       endif;
```

12 **endfor**;
13 **end**

In Algorithm 2.2, we make another pass through the destination sets whenever a successful combination is made. The reason for this is that the offered load of the larger set has been increased and, as can be seen from Fig. 2.3, since the threshold is monotonically increasing, the range of offered loads for the smaller tree allowing for capacity savings when sharing has also been increased. Therefore, new combinations could potentially be made that were not allowed on a previous pass through the destination sets.

To see how close to optimal Algorithms 2.1 and 2.2 might be in practice, we ran simulations of Algorithms 2.1 and 2.2 and the brute force approach using common random numbers. There were m destination sets with $\mathcal{D}_m \subset \mathcal{D}_{m-1} \subset \cdots \subset \mathcal{D}_1$, and the tree sizes were $1, 2, \ldots, m$, respectively. The offered loads were randomly generated according to a uniform distribution between 0 and $\rho_{\max}$. For each case, the results obtained are 95% confidence intervals based on independent replications. The call blocking probability was held constant at 10^{-3}.

From the results shown in Table 2.1, we see that Algorithm 2.1 is consistently better in total capacity than Algorithm 2.2 and is quite close to the optimal. As expected, the savings percentage drops off significantly as $\rho_{\max}$ grows larger, but it improves as the size and number of the destinations sets m increases. For this scenario, Algorithm 2.1 appears to be a good compromise between complexity and performance.

Two additional statistics are shown in Table 2.2: the link capacity standard deviation and the final number of trees obtained after running the algorithms.[3] Unlike the other statistics, the link capacity standard deviation is topology-dependent, and for our current experiments we employed a topology similar to that shown in Fig. 2.10 for $m = 5$. The link capacity standard deviation gives us a measure of how well the load is balanced across the links of the network with a standard deviation of zero signifying that all links have the same capacity. As can be seen in Table 2.2, the load is indeed better balanced after running our algorithms, with Algorithms 2.1 and 2.2 both beating the brute force algorithm. The benefits are most dramatic at low loads and tail off as $\rho_{\max}$ increases. Also, the benefits increase as the number

[3]Note that if we tried to optimize in terms of the link capacity standard deviation or the number of trees instead of the total required capacity, we would always end up with one tree.

	Total Capacity				Savings Percentage		
	Original	Optimal	Alg. 2.1	Alg. 2.2	Optimal	Alg. 2.1	Alg. 2.2
$\rho_{max} = 10$							
$m = 2$	34.9 ± 5.9	34.4 ± 5.9	34.4 ± 5.9	34.4 ± 5.9	1.8 ± 1.3	1.8 ± 1.3	1.8 ± 1.3
4	110.2 ± 17.2	100.0 ± 16.4	100.0 ± 16.4	100.0 ± 16.4	9.5 ± 2.9	9.5 ± 2.9	9.5 ± 2.9
8	435.8 ± 49.8	358.7 ± 46.9	363.4 ± 46.5	364.5 ± 47.0	18.0 ± 2.0	16.8 ± 2.5	16.6 ± 2.5
12	952.7 ± 109.1	740.8 ± 96.8	747.0 ± 94.8	753.0 ± 92.0	22.5 ± 1.8	21.8 ± 1.8	21.1 ± 1.2
$\rho_{max} = 30$							
$m = 2$	71.9 ± 11.2	71.7 ± 11.3	71.7 ± 11.3	71.7 ± 11.3	0.5 ± 0.5	0.5 ± 0.5	0.5 ± 0.5
4	231.6 ± 39.3	223.7 ± 39.0	223.8 ± 39.0	223.8 ± 39.0	3.7 ± 2.1	3.6 ± 2.1	3.6 ± 2.1
8	906.6 ± 120.0	828.7 ± 119.2	829.9 ± 119.4	834.0 ± 116.2	8.9 ± 1.7	8.7 ± 1.7	8.1 ± 1.8
12	1986 ± 263.6	1747 ± 251.4	1769 ± 256.3	1783 ± 255.4	12.3 ± 1.4	11.2 ± 1.8	10.4 ± 1.9
$\rho_{max} = 50$							
$m = 2$	104.8 ± 16.4	104.7 ± 16.4	104.7 ± 16.4	104.7 ± 16.4	0.3 ± 0.3	0.3 ± 0.3	0.3 ± 0.3
4	328.8 ± 52.8	322.3 ± 52.6	322.4 ± 52.7	322.4 ± 52.7	2.2 ± 1.3	2.1 ± 1.4	2.1 ± 1.4
8	1327 ± 185.6	1252 ± 186.8	1257 ± 186.4	1258 ± 186.2	5.9 ± 1.5	5.5 ± 1.6	5.5 ± 1.6
12	2910 ± 408.1	2665 ± 398.7	2684 ± 390.5	2699 ± 387.5	8.7 ± 1.2	7.9 ± 1.3	7.3 ± 1.5

Table 2.1: Final capacities and savings percentages for simulations aggregating multicast trees with demands uniformly distributed between 0 and ρ_{max} and a blocking probability of 10^{-3}.

	Link Capacity Standard Deviation				Final Number of Trees		
	Original	Optimal	Alg. 2.1	Alg. 2.2	Optimal	Alg. 2.1	Alg. 2.2
$\rho_{max} = 10$							
$m = 2$	7.9 ± 1.6	6.0 ± 2.4	6.0 ± 2.4	6.0 ± 2.4	1.6 ± 0.2	1.6 ± 0.2	1.6 ± 0.2
4	15.3 ± 2.8	10.9 ± 3.8	10.9 ± 3.8	10.9 ± 3.8	2.4 ± 0.5	2.4 ± 0.5	2.4 ± 0.5
8	30.7 ± 3.8	19.0 ± 3.3	15.4 ± 5.3	15.2 ± 5.2	3.4 ± 0.4	3.2 ± 0.5	3.2 ± 0.5
12	46.0 ± 4.1	25.0 ± 3.5	21.5 ± 3.6	21.0 ± 3.6	3.7 ± 0.5	3.4 ± 0.4	3.3 ± 0.5
$\rho_{max} = 30$							
$m = 2$	16.3 ± 3.2	15.9 ± 3.5	15.9 ± 3.5	15.9 ± 3.5	1.9 ± 0.1	1.9 ± 0.1	1.9 ± 0.1
4	33.2 ± 6.6	27.3 ± 7.9	26.7 ± 8.5	26.7 ± 8.5	2.8 ± 0.4	2.8 ± 0.4	2.8 ± 0.4
8	64.3 ± 9.1	46.9 ± 8.5	45.8 ± 8.0	45.4 ± 8.0	4.1 ± 0.4	4.1 ± 0.4	4.1 ± 0.4
12	96.4 ± 9.8	70.9 ± 9.3	60.8 ± 9.9	57.8 ± 9.8	5.0 ± 0.3	4.3 ± 0.4	4.2 ± 0.3
$\rho_{max} = 50$							
$m = 2$	24.9 ± 5.0	24.3 ± 5.3	24.3 ± 5.3	24.3 ± 5.3	1.9 ± 0.1	1.9 ± 0.1	1.9 ± 0.1
4	47.9 ± 9.0	42.6 ± 9.6	41.4 ± 10.5	41.4 ± 10.5	3.1 ± 0.4	3.0 ± 0.4	3.0 ± 0.4
8	94.5 ± 14.1	79.1 ± 14.0	74.5 ± 12.7	74.5 ± 12.7	4.8 ± 0.5	4.6 ± 0.4	4.6 ± 0.4
12	141.7 ± 15.0	113.4 ± 12.5	105.6 ± 15.4	104.3 ± 16.1	6.0 ± 0.5	5.4 ± 0.4	5.5 ± 0.4

Table 2.2: Link capacity standard deviations and final number of trees for simulations aggregating multicast trees with demands uniformly distributed between 0 and ρ_{max} and a call blocking probability of 10^{-3}.

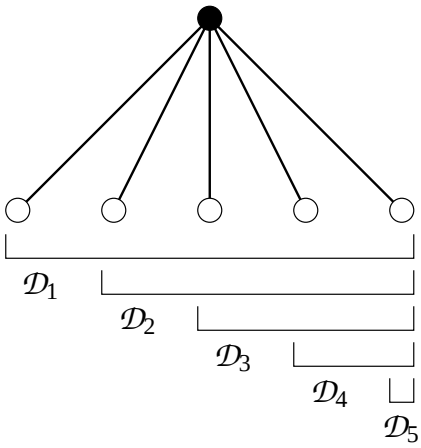


Figure 2.10: One-level tree topology, shown with $m = 5$, used to determine the link capacity standard deviation in the simulations.

of destination sets m increases. Although it is difficult to quantify, we expect that similar load-balancing benefits might be seen for other more general topologies. The results for the final number of trees are also quite encouraging. Once again, the greatest benefits occur for smaller $\rho_{\max}$, and they grow as m increases. The savings in the number of trees translates to reduced VPI usage and a significant reduction in VP setup and management loads.

We repeated the simulations with the addition of the Gaussian traffic model discussed at the end of Sec. 2.3. For each connection, the mean and variance of the cell arrival rate was given by $\lambda = 1$ and $\sigma^2 = 1$, respectively. The cell loss probability was held constant at 10^{-6} which translates to a value of 4.7534 for the QoS parameter k in (2.4). The results, shown in Tables 2.3 and 2.4, exhibit the same general trends as the previous experiments but with a large increase in the capacity savings percentages. There is also a more significant reduction in the link capacity standard deviation and the final number of trees. Although the Gaussian model is certainly not the best cell level model, it does effectively demonstrate the potential benefits if rate multiplexing is taken into account when aggregating multicast trees.

We shall now drop the assumption that $\mathcal{D}_m \subset \mathcal{D}_{m-1} \subset \cdots \subset \mathcal{D}_1$. We still keep the less restrictive assumption that $\mathcal{D}_i \subset \mathcal{D}_1$ for $i = 2, 3, \ldots, m$, so with $N = |\mathcal{D}_1|$, the subsets $\mathcal{D}_i$, $i = 2, 3, \ldots, m$, can have from 1 to $N-1$ elements, and we can form equivalence classes based on how many elements are in each set. We can also construct relationships based on which destination sets are subsets of other

	Total Capacity				Savings Percentage		
	Original	Optimal	Alg. 2.1	Alg. 2.2	Optimal	Alg. 2.1	Alg. 2.2
$\rho_{max} = 10$							
$m = 2$	86.3 ± 11.5	79.4 ± 11.6	79.4 ± 11.6	79.4 ± 11.6	8.3 ± 3.0	8.3 ± 3.0	8.3 ± 3.0
4	265.2 ± 33.3	210.5 ± 29.3	210.6 ± 29.3	210.6 ± 29.3	20.7 ± 4.0	20.7 ± 4.1	20.7 ± 4.1
8	1019 ± 85.5	661.3 ± 72.7	663.6 ± 75.1	664.0 ± 75.4	35.3 ± 2.4	35.1 ± 2.8	35.1 ± 2.8
12	2220 ± 186.7	1301 ± 137.6	1326 ± 146.2	1330 ± 150.0	41.6 ± 1.6	40.5 ± 2.2	40.3 ± 2.3
$\rho_{max} = 30$							
$m = 2$	139.0 ± 23.0	134.6 ± 23.1	134.6 ± 23.1	134.6 ± 23.1	3.4 ± 2.0	3.4 ± 2.0	3.4 ± 2.0
4	447.2 ± 69.5	389.0 ± 66.5	389.0 ± 66.5	389.0 ± 66.5	13.3 ± 3.7	13.3 ± 3.7	13.3 ± 3.7
8	1740 ± 180.3	1338 ± 165.3	1345 ± 168.5	1350 ± 171.0	23.4 ± 2.3	23.0 ± 2.6	22.8 ± 2.7
12	3800 ± 394.7	2703 ± 325.8	2744 ± 330.2	2771 ± 340.2	29.1 ± 1.6	28.0 ± 1.9	27.3 ± 1.8
$\rho_{max} = 50$							
$m = 2$	187.0 ± 31.0	183.7 ± 31.1	183.7 ± 31.1	183.7 ± 31.1	1.9 ± 1.4	1.9 ± 1.4	1.9 ± 1.4
4	595.7 ± 101.1	536.3 ± 97.2	536.3 ± 97.2	536.3 ± 97.2	10.3 ± 3.2	10.3 ± 3.2	10.3 ± 3.2
8	2333 ± 263.0	1903 ± 247.0	1929 ± 249.8	1936 ± 252.3	18.7 ± 2.0	17.5 ± 2.5	17.3 ± 2.5
12	5098 ± 576.4	3918 ± 504.9	3955 ± 505.2	3981 ± 487.0	23.4 ± 1.7	22.6 ± 1.8	22.1 ± 1.2

Table 2.3: Final capacities and savings percentages for simulations aggregating multicast trees with bufferless links, demands uniformly distributed between 0 and ρ_{max}, a call blocking probability of 10^{-3}, a cell loss probability of 10^{-6}, and a Gaussian cell arrival rate with mean 1 and variance 1.

	Link Capacity Standard Deviation				Final Number of Trees		
	Original	Optimal	Alg. 2.1	Alg. 2.2	Optimal	Alg. 2.1	Alg. 2.2
$\rho_{max} = 10$							
$m = 2$	19.0 ± 3.8	3.3 ± 4.9	3.3 ± 4.9	3.3 ± 4.9	1.1 ± 0.2	1.1 ± 0.2	1.1 ± 0.2
4	35.9 ± 5.5	18.3 ± 6.7	16.0 ± 4.9	16.0 ± 4.9	1.9 ± 0.2	1.9 ± 0.2	1.9 ± 0.2
8	70.8 ± 6.6	22.9 ± 8.4	20.5 ± 5.6	20.4 ± 5.6	2.2 ± 0.3	2.1 ± 0.2	2.1 ± 0.2
12	105.9 ± 7.1	35.7 ± 6.2	18.5 ± 5.2	18.0 ± 5.2	2.7 ± 0.4	2.3 ± 0.4	2.3 ± 0.4
$\rho_{max} = 30$							
$m = 2$	30.9 ± 6.1	16.8 ± 10.4	16.8 ± 10.4	16.8 ± 10.4	1.4 ± 0.2	1.4 ± 0.2	1.4 ± 0.2
4	61.3 ± 11.3	40.7 ± 16.6	40.7 ± 16.6	40.7 ± 16.6	2.3 ± 0.5	2.3 ± 0.5	2.3 ± 0.5
8	121.9 ± 13.8	58.7 ± 19.3	48.0 ± 17.6	47.4 ± 17.0	3.0 ± 0.6	2.7 ± 0.5	2.7 ± 0.5
12	182.6 ± 14.8	82.5 ± 13.9	65.1 ± 11.5	61.0 ± 11.1	3.1 ± 0.5	2.9 ± 0.4	2.8 ± 0.5
$\rho_{max} = 50$							
$m = 2$	42.2 ± 8.2	31.9 ± 12.9	31.9 ± 12.9	31.9 ± 12.9	1.6 ± 0.2	1.6 ± 0.2	1.6 ± 0.2
4	82.2 ± 16.4	58.9 ± 22.3	58.9 ± 22.3	58.9 ± 22.3	2.4 ± 0.5	2.4 ± 0.5	2.4 ± 0.5
8	164.0 ± 20.1	101.1 ± 17.3	78.9 ± 27.3	78.2 ± 26.7	3.4 ± 0.4	3.1 ± 0.4	3.1 ± 0.4
12	245.8 ± 21.6	132.7 ± 18.3	110.8 ± 17.0	108.7 ± 17.0	3.7 ± 0.5	3.2 ± 0.5	3.2 ± 0.5

Table 2.4: Link capacity standard deviations and final number of trees for simulations aggregating multicast trees with bufferless links, demands uniformly distributed between 0 and ρ_{max}, a call blocking probability of 10^{-3}, a cell loss probability of 10^{-6}, and a Gaussian cell arrival rate with mean 1 and variance 1.

destination sets. Accordingly, we present a modified version of Algorithm 2.1 that proceeds down the hierarchy by equivalence class.

Algorithm 2.3. *Start with the largest destination set $\mathcal{D}_1$ and the equivalence class directly below $\mathcal{D}_1$ with $N-1$ elements per destination set. For each $\mathcal{D}_i$ in that equivalence class (taken in any order), combine the demands for $\mathcal{D}_1$ and $\mathcal{D}_i$ into a single demand for $\mathcal{D}_1$ if the condition in* (2.1) *holds. If successful in combining all members in that equivalence class with $\mathcal{D}_1$ (or if the equivalence class is empty), try to combine demands for $\mathcal{D}_1$ and the members of the equivalence class with $N-2$ elements. Continue on until reaching an equivalence class in which all members are not combined with $\mathcal{D}_1$. Now repeat the procedure starting with each destination set of that class (in any order) and restricting combinations to destination sets which are subsets of the current destination set. Continue on recursively until all destination sets have either been absorbed or have served as the primary candidate to which other destination sets may be combined.*

```
1  begin
2    for n := N to 2 step -1 do
3      foreach i ∈ {1,2,...,m} such that |D_i| = n ∧ ρ_i > 0 do
4        q := n-1; k := 0;
5        while q ≥ 1 ∧ k = 0 do
6          foreach j ∈ {1,2,...,m} such that |D_j| = q ∧ ρ_j > 0 do
7            if D_j ⊂ D_i ∧ ¬combine(i,j) then k := k+1; endif;
8          endfor;
9          q := q-1;
10       endwhile;
11     endfor;
12   endfor;
13 end
```

For completeness, we also define a modified version of Algorithm 2.2.

Algorithm 2.4. *Start with the largest destination set $\mathcal{D}_1$ and the equivalence class directly below $\mathcal{D}_1$ with $N-1$ elements per destination set. For each $\mathcal{D}_i$ in that equivalence class (taken in any order), combine the demands for $\mathcal{D}_1$ and $\mathcal{D}_i$ into a single demand for $\mathcal{D}_1$ if the condition in* (2.1) *holds. Next try to combine demands*

for $\mathcal{D}_1$ and the members of the equivalence class with $N-2$ elements, and continue on until reaching the equivalence class with the smallest number of elements per destination set. If a successful combination was made, repeat the procedure starting with $\mathcal{D}_1$ and the equivalence class with $N-1$ elements per destination set (skipping sets which have previously been absorbed). When a pass without a successful combination has been made, move from $\mathcal{D}_1$ to the equivalence class with the largest number of elements per destination set that has members which have not been aggregated, and repeat from the beginning starting with each destination set of that class (in any order) and restricting combinations to destination sets which are subsets of the current destination set. Continue on recursively until all destination sets have either been absorbed or have served as the primary candidate to which other destination sets may be combined.

```
1  begin
2    for n := N to 2 step −1 do
3      foreach i ∈ {1,2,...,m} such that |D_i| = n ∧ ρ_i > 0 do
4        k := 1;
5        while k > 0 do
6              k := 0;
7              for q := n−1 to 1 step −1 do
8                foreach j ∈ {1,2,...,m} such that |D_j| = q ∧ ρ_j > 0 do
9                  if D_j ⊂ D_i ∧ combine(i, j) then k := k+1; endif;
10               endfor;
11             endfor;
12       endwhile;
13     endfor;
14   endfor;
15 end
```

To simulate Algorithms 2.3 and 2.4, we began with a destination set of size m and from the 2^m-2 proper subsets (excluding the empty set) chose $m-1$ other destination sets at random at the beginning of each replication. This means that the majority of the destination sets chosen will have close to $m/2$ members. As before, the tree sizes were equal to the destination set sizes, the offered loads were randomly generated according to a uniform distribution between 0 and $\rho_{\max}$, and the results obtained are 95% confidence intervals based on independent replications.

To compute the link capacity standard deviation, we used a one-level tree topology, similar to that shown in Fig. 2.10, but now the destination sets are chosen at random from all possible combinations. We performed simulations with and without the Gaussian traffic model using the same traffic parameters and blocking probabilities as before.

The results for these experiments are given in Tables 2.5 through 2.8. We do not repeat the results for $m = 2$ as they are the same as before. Algorithm 2.3 is close to the optimal in total capacity, but, in contrast to before, it is no longer consistently better than Algorithm 2.4. The savings percentages have been reduced significantly, but they are still quite good when the Gaussian traffic model is included. The results for the link capacity standard deviation and the final number of trees generally exhibit the same trends as before except that the values for Algorithm 2.3 tend to be a bit higher than the other two algorithms. Overall, the impact of including the Gaussian traffic model is more significant than before, and Algorithm 2.3 still appears to be a good compromise between complexity and performance.

2.5 Role of topology and pre-processing

In this section, we restrict ourselves to the case where the VP multicast layout has not yet been determined, and our procedure is a pre-processing step. As in Sec. 2.1, we consider the situation with a common source and two destination sets $\mathcal{D}_1$ and $\mathcal{D}_2$ such that $\mathcal{D}_2 \subseteq \mathcal{D}_1$. Suppose that we do not know the actual network topology, but we do know the distances from the source to all destinations in the larger set $\mathcal{D}_1$. The exact ratio $|\mathcal{T}_2|/|\mathcal{T}_1|$ is then unknown, but we can bound it for all possible trees $\mathcal{T}_1$ and $\mathcal{T}_2$ with the specified distances to each destination. In the following, $h(d)$ is the number of hops to destination d.

Lemma 2.1. *Let* $u(\mathcal{D}) = \sum_{d\in\mathcal{D}} h(d)$ *and* $l(\mathcal{D}) = |\mathcal{D}| + \sum_{j=1}^{M-1} 1(j \neq h(d)\ \forall\, d \in \mathcal{D})$, *where* $M = \max_{d\in\mathcal{D}} h(d)$ *and* $1(\cdot)$ *is the indicator function. Then, for any trees* $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *satisfying the distance constraints* $h(d)\ \forall\, d \in \mathcal{D}_1$ *or* $\mathcal{D}_2$, *respectively, and connecting a common source to destination sets* $\mathcal{D}_1$ *and* $\mathcal{D}_2$, *respectively, with* $\mathcal{D}_2 \subseteq \mathcal{D}_1$, *we have*

$$\frac{l(\mathcal{D}_2)}{u(\mathcal{D}_1)} \leq \frac{|\mathcal{T}_2|}{|\mathcal{T}_1|} \leq \frac{u(\mathcal{D}_2)}{l(\mathcal{D}_1)}. \tag{2.5}$$

	Total Capacity				Savings Percentage		
	Original	Optimal	Alg. 2.3	Alg. 2.4	Optimal	Alg. 2.3	Alg. 2.4
$\rho_{max} = 10$							
$m = 4$	121.5 ± 16.2	112.1 ± 14.4	112.5 ± 14.5	112.3 ± 14.5	7.4 ± 2.7	7.1 ± 2.8	7.3 ± 2.8
8	457.5 ± 45.7	404.1 ± 38.0	405.2 ± 37.7	405.7 ± 37.4	11.4 ± 2.6	11.2 ± 2.7	11.0 ± 2.8
12	961.5 ± 123.1	843.9 ± 95.9	865.7 ± 92.7	847.3 ± 97.8	11.9 ± 3.2	9.5 ± 4.1	11.6 ± 3.6
$\rho_{max} = 30$							
$m = 4$	252.4 ± 37.9	245.8 ± 37.2	246.1 ± 37.4	245.8 ± 37.2	2.6 ± 1.5	2.6 ± 1.6	2.6 ± 1.5
8	956.3 ± 108.0	916.4 ± 102.1	917.8 ± 101.6	918.5 ± 100.9	4.1 ± 1.4	4.0 ± 1.4	3.9 ± 1.4
12	2004 ± 288.5	1939 ± 264.5	1957 ± 264.8	1939 ± 264.5	3.1 ± 1.4	2.2 ± 1.4	3.1 ± 1.4
$\rho_{max} = 50$							
$m = 4$	372.0 ± 55.1	366.7 ± 54.7	366.7 ± 54.7	366.7 ± 54.7	1.5 ± 1.0	1.5 ± 1.0	1.5 ± 1.0
8	1381 ± 160.7	1351 ± 157.1	1352 ± 156.8	1353 ± 155.6	2.2 ± 0.9	2.1 ± 0.9	1.9 ± 0.9
12	2937 ± 442.1	2902 ± 428.3	2912 ± 439.7	2903 ± 428.8	1.1 ± 0.7	0.9 ± 0.5	1.1 ± 0.7

Table 2.5: Final capacities and savings percentages for simulations aggregating multicast trees with random destination sets, demands uniformly distributed between 0 and ρ_{max}, and a call blocking probability of 10^{-3}.

	Link Capacity Standard Deviation				Final Number of Trees		
	Original	Optimal	Alg. 2.3	Alg. 2.4	Optimal	Alg. 2.3	Alg. 2.4
$\rho_{max} = 10$							
$m = 4$	11.3 ± 2.2	9.2 ± 2.6	9.6 ± 2.2	8.8 ± 2.7	2.3 ± 0.4	2.6 ± 0.4	2.3 ± 0.4
8	17.5 ± 3.4	13.3 ± 3.0	14.0 ± 3.2	12.3 ± 2.8	3.8 ± 0.6	4.1 ± 0.7	3.7 ± 0.6
12	21.7 ± 3.1	17.2 ± 2.5	17.8 ± 3.2	15.2 ± 2.9	4.7 ± 0.8	7.2 ± 2.2	5.1 ± 1.9
$\rho_{max} = 30$							
$m = 4$	24.2 ± 4.8	22.0 ± 4.6	22.7 ± 4.6	22.0 ± 4.6	2.9 ± 0.4	3.0 ± 0.4	2.9 ± 0.4
8	37.1 ± 7.4	33.6 ± 5.6	33.7 ± 5.5	32.7 ± 6.1	5.4 ± 0.7	5.5 ± 0.7	5.3 ± 0.8
12	46.7 ± 6.7	44.5 ± 6.4	45.8 ± 6.2	44.5 ± 6.4	7.8 ± 1.6	9.8 ± 1.4	7.8 ± 1.6
$\rho_{max} = 50$							
$m = 4$	35.9 ± 6.8	33.3 ± 7.3	33.3 ± 7.3	33.3 ± 7.3	3.3 ± 0.4	3.3 ± 0.4	3.3 ± 0.4
8	54.0 ± 10.3	51.5 ± 9.2	51.9 ± 9.5	51.4 ± 9.3	6.3 ± 0.7	6.5 ± 0.7	6.3 ± 0.6
12	69.4 ± 10.0	68.8 ± 9.5	68.9 ± 9.5	68.8 ± 9.5	9.8 ± 1.2	10.7 ± 0.8	9.8 ± 1.2

Table 2.6: Link capacity standard deviations and final number of trees for simulations aggregating multicast trees with random destination sets, demands uniformly distributed between 0 and ρ_{max}, and a call blocking probability of 10^{-3}.

	Total Capacity				Savings Percentage		
	Original	Optimal	Alg. 2.3	Alg. 2.4	Optimal	Alg. 2.3	Alg. 2.4
$\rho_{max} = 10$							
$m = 4$	285.1 ± 30.1	226.0 ± 21.5	226.1 ± 21.5	226.0 ± 21.5	20.0 ± 4.0	20.0 ± 4.1	20.0 ± 4.0
8	1061 ± 90.9	728.0 ± 53.2	728.0 ± 53.2	728.3 ± 53.3	31.2 ± 2.5	31.2 ± 2.5	31.1 ± 2.6
12	2242 ± 228.8	1413 ± 124.1	1416 ± 127.3	1416 ± 127.3	36.8 ± 2.4	36.6 ± 2.5	36.6 ± 2.5
$\rho_{max} = 30$							
$m = 4$	485.9 ± 59.8	429.3 ± 50.8	431.9 ± 50.9	431.4 ± 50.9	11.2 ± 3.3	10.7 ± 3.5	10.8 ± 3.5
8	1823 ± 183.5	1505 ± 136.0	1512 ± 133.2	1508 ± 134.7	17.1 ± 3.1	16.6 ± 3.4	17.0 ± 3.1
12	3836 ± 455.8	3050 ± 319.4	3100 ± 334.5	3050 ± 319.4	20.3 ± 2.8	18.8 ± 5.2	20.3 ± 2.8
$\rho_{max} = 50$							
$m = 4$	650.6 ± 85.6	596.7 ± 75.6	599.1 ± 76.1	597.7 ± 76.3	7.9 ± 2.8	7.6 ± 2.8	7.8 ± 2.8
8	2448 ± 242.2	2143 ± 197.3	2149 ± 195.3	2150 ± 194.3	12.2 ± 2.7	11.9 ± 2.8	11.9 ± 2.8
12	5145 ± 652.2	4461 ± 502.3	4548 ± 499.4	4482 ± 510.9	13.0 ± 3.2	11.2 ± 4.5	12.6 ± 3.8

Table 2.7: Final capacities and savings percentages for simulations aggregating multicast trees with random destination sets, bufferless links, demands uniformly distributed between 0 and ρ_{max}, a call blocking probability of 10^{-3}, a cell loss probability of 10^{-6}, and a Gaussian cell arrival rate with mean 1 and variance 1.

	Link Capacity Standard Deviation				Final Number of Trees		
	Original	Optimal	Alg. 2.3	Alg. 2.4	Optimal	Alg. 2.3	Alg. 2.4
$\rho_{max} = 10$							
$m = 4$	25.2 ± 4.8	7.1 ± 4.6	8.4 ± 5.0	7.1 ± 4.6	1.6 ± 0.4	1.6 ± 0.4	1.6 ± 0.4
8	40.7 ± 7.9	8.4 ± 5.9	8.4 ± 5.9	8.3 ± 5.8	1.5 ± 0.4	1.5 ± 0.4	1.5 ± 0.4
12	48.1 ± 7.1	13.4 ± 8.8	10.6 ± 8.2	10.6 ± 8.2	2.0 ± 0.9	1.8 ± 0.7	1.8 ± 0.7
$\rho_{max} = 30$							
$m = 4$	44.4 ± 8.5	33.5 ± 10.9	28.5 ± 9.4	28.8 ± 9.5	2.2 ± 0.4	2.1 ± 0.4	2.1 ± 0.4
8	70.8 ± 14.0	49.6 ± 9.1	49.4 ± 10.4	48.0 ± 10.4	3.0 ± 0.4	3.5 ± 0.7	3.1 ± 0.4
12	84.9 ± 12.4	34.8 ± 18.6	43.9 ± 17.7	34.8 ± 18.6	2.7 ± 1.1	4.1 ± 2.3	2.7 ± 1.1
$\rho_{max} = 50$							
$m = 4$	60.4 ± 11.7	49.0 ± 13.9	51.0 ± 11.7	46.7 ± 14.2	2.3 ± 0.4	2.6 ± 0.4	2.3 ± 0.4
8	93.6 ± 18.2	73.1 ± 16.1	76.6 ± 16.6	67.8 ± 14.9	3.7 ± 0.6	4.0 ± 0.6	3.6 ± 0.6
12	115.7 ± 16.7	83.4 ± 9.6	89.4 ± 19.1	77.4 ± 11.9	4.1 ± 0.6	6.5 ± 2.2	4.8 ± 1.9

Table 2.8: Link capacity standard deviations and final number of trees for simulations aggregating multicast trees with random destination sets, bufferless links, demands uniformly distributed between 0 and ρ_{max}, a call blocking probability of 10^{-3}, a cell loss probability of 10^{-6}, and a Gaussian cell arrival rate with mean 1 and variance 1.

Proof: To prove the lemma, we show that $l(\mathcal{D}) \leq |\mathcal{T}| \leq u(\mathcal{D})$ for any tree $\mathcal{T}$ connecting a source to destination set $\mathcal{D}$. The largest number of links occurs when the tree $\mathcal{T}$ has disjoint paths from the source to every destination, i.e., the root of the tree has $|\mathcal{D}|$ branches and no links are shared by two or more source-destination paths. Therefore, $|\mathcal{T}| \leq \sum_{d\in\mathcal{D}} h(d) = u(\mathcal{D})$. To establish the lower bound, we assume that no two destinations are collocated. (If not, simply combine the demands for the collocated destinations into a demand for a single combined destination.) First, suppose that $\{h(d)|d \in \mathcal{D}\} = \{1,2,\ldots,M\}$. Then the tree $\mathcal{T}$ with the smallest number of links occurs when the destinations are connected in a straight line. Thus, $|\mathcal{T}| = M = |\mathcal{D}|$. If another destination is added at a duplicate distance $1 \leq j \leq M$, then another branch must be added originating from a node at distance $j-1$. If a destination is removed at distance $1 \leq j < M$ such that now $j \notin \{h(d)|d \in \mathcal{D}\}$, then a branch cannot be removed because the tree would become disconnected. Thus, by construction, $|\mathcal{T}| \geq |\mathcal{D}| + \sum_{j=1}^{M-1} 1(j \neq h(d)\ \forall\, d \in \mathcal{D}) = l(\mathcal{D})$. ■

The bounds in Lemma 2.1 are not very tight. For example, define $\Gamma(\mathcal{D}) = \{h(d)|d \in \mathcal{D}\}$, and suppose that $\Gamma(\mathcal{D}_2) = \{3,5\}$ and $\Gamma(\mathcal{D}_1) = \{2,3,5\}$. Then we have $\frac{5}{10} \leq \frac{|\mathcal{T}_2|}{|\mathcal{T}_1|} \leq \frac{8}{5}$.

Now consider a similar setup with the additional requirement that $\mathcal{T}_2 \subseteq \mathcal{T}_1$. Tighter bounds than before can be obtained as stated in the following lemma. We use $\mathcal{D}_1 - \mathcal{D}_2$ to designate the set of elements in $\mathcal{D}_1$ that are not in $\mathcal{D}_2$.

Lemma 2.2. *Let $u(\mathcal{D}) = \sum_{d\in\mathcal{D}} h(d)$ and $l(\mathcal{D}) = |\mathcal{D}| + \sum_{j=1}^{M-1} 1(j \neq h(d)\ \forall\, d \in \mathcal{D})$, where $M = \max_{d\in\mathcal{D}} h(d)$ and $1(\cdot)$ is the indicator function. Then, for any trees $\mathcal{T}_1$ and $\mathcal{T}_2$ satisfying the distance constraints $h(d)\ \forall\, d \in \mathcal{D}_1$ or $\mathcal{D}_2$, respectively, and connecting a common source to destination sets $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively, with $\mathcal{D}_2 \subseteq \mathcal{D}_1$ and $\mathcal{T}_2 \subseteq \mathcal{T}_1$, we have*

$$\frac{l(\mathcal{D}_2)}{l(\mathcal{D}_2) + u(\mathcal{D}_1) - u(\mathcal{D}_2)} \leq \frac{|\mathcal{T}_2|}{|\mathcal{T}_1|} \leq \min\left\{\frac{u(\mathcal{D}_2)}{l(\mathcal{D}_1)}, 1\right\}. \tag{2.6}$$

Proof: Since $\mathcal{T}_2 \subseteq \mathcal{T}_1$, $|\mathcal{T}_2| \leq |\mathcal{T}_1|$, and so $|\mathcal{T}_2|/|\mathcal{T}_1| \leq 1$. From Lemma 2.1, we also know that $|\mathcal{T}_2|/|\mathcal{T}_1| \leq u(\mathcal{D}_2)/l(\mathcal{D}_1)$, thus establishing the upper bound. For the lower bound, let $\mathcal{T}_{12}$ be a separate tree for $\mathcal{D}_1 - \mathcal{D}_2$. From Lemma 2.1, we know that $|\mathcal{T}_{12}| \leq u(\mathcal{D}_1 - \mathcal{D}_2) = u(\mathcal{D}_1) - u(\mathcal{D}_2)$. Since $\mathcal{T}_2 \subseteq \mathcal{T}_1$, $|\mathcal{T}_1| \leq |\mathcal{T}_2| + u(\mathcal{D}_1) - u(\mathcal{D}_2)$. So we have

$$\frac{|\mathcal{T}_2|}{|\mathcal{T}_2| + u(\mathcal{D}_1) - u(\mathcal{D}_2)} \leq \frac{|\mathcal{T}_2|}{|\mathcal{T}_1|}. \tag{2.7}$$

To find the lower bound over all trees $\mathcal{T}_2$, we differentiate the left-hand side of (2.7) with respect to $|\mathcal{T}_2|$. The derivative, $(u(\mathcal{D}_1) - u(\mathcal{D}_2))/(|\mathcal{T}_2| + u(\mathcal{D}_1) - u(\mathcal{D}_2))^2$, is always positive because $u(\mathcal{D}_1) - u(\mathcal{D}_2) \geq 0$. Therefore, the left-hand side of (2.7) is an increasing function of $|\mathcal{T}_2|$. Hence, the smallest possible value of $|\mathcal{T}_2|$, which is $l(\mathcal{D}_2)$, is substituted for $|\mathcal{T}_2|$ to establish the lower bound. ■

Using Lemma 2.2, the bounds for our previous example are $\frac{5}{7} \leq \frac{|\mathcal{T}_2|}{|\mathcal{T}_1|} \leq 1$, a significant improvement.

By substituting the lower bound of Lemma 2.1 or preferably Lemma 2.2 for $|\mathcal{T}_2|/|\mathcal{T}_1|$ in (2.1), we can conservatively decide whether or not to combine the demands for $\mathcal{D}_1$ and $\mathcal{D}_2$ into a single demand for the larger set $\mathcal{D}_1$ without knowing the full network topology. As confirmed by Lemma 2.2 and Fig. 2.3, large destination sets ($l(\mathcal{D}_2)$ large) with small differences between them ($u(\mathcal{D}_1) - u(\mathcal{D}_2)$ small) will produce a lower bound for $|\mathcal{T}_2|/|\mathcal{T}_1|$ close to 1 (the maximum value) and improve the likelihood of achieving positive capacity savings by combining the demands. Without knowing the distances to the relevant destinations, as we have assumed in this section, it is not feasible to bound $|\mathcal{T}_2|/|\mathcal{T}_1|$ and make any decisions as part of a pre-processing step.

2.6 Additional remarks

In conclusion, we comment on the practicality of the common source assumption for large numbers of destination sets. Instead of an end system, the source could very well be a "core" node inside the network from which core-based trees are established for multicast routing, an architecture being proposed for the Internet [5, 6]. A multicast route would consist of a Virtual Path Connection (VPC) or SVC from the source to the core node followed by the established VP tree. Also, the destinations may be gateway nodes instead of end systems, in which case the VP trees would be entirely contained within the backbone of the network.

It is also worth noting that after aggregation, it may be desirable to perform trunk reservation within a VP tree because of the varying revenues generated by incoming connections [37]. For example, with two types of connections, it is optimal (in terms of total revenue generated) to reserve a certain amount of capacity for the connections which generate more revenue [25].

Chapter 3

Adaptive Resource Allocation for Aggregated Flows

3.1 Introduction

While the flexibility of ATM networks to support a variety of service classes and other value added services is likely to materialize, network operators will be faced with increasingly complex operational constraints and tradeoffs among overall network throughput, processing loads on the signaling system, and QoS requirements at both the call level, e.g., blocking probabilities, and cell level, e.g., cell loss or delay characteristics. One way to control the daunting complexity of the system is to aggregate flows, i.e., use Virtual Path Connections (VPCs) or configure virtual networks, and make *a priori* fixed or adaptive resource allocations. With flow aggregation, a group of individual flows is bundled together and jointly managed and switched inside a network or subnetwork. This type of aggregation can reduce the size of routing tables or the signaling/processing loads (e.g., Connection Admission Control) at the switches, reduce call setup times, and/or reduce the number of connection labels needed inside a network.

After noting the related work in this area in Sec. 3.2, the first concern of this chapter is to provide an overview and crude evaluation in Sec. 3.3 of the increasing signaling loads that large-scale, connection-oriented networks will need to support. Sec. 3.4 considers the role that configuring an overlay VP network and traffic management algorithms can play in controlling what we currently perceive as a possible

lack of scalability for ATM networks. We present some specific VP capacity allocation schemes for both VP-only and mixed VP/VC switching environments. Finally, in Sec. 3.5, we propose and evaluate algorithms for migrating from one VP layout to another. Sec. 3.6 concludes with a chapter summary.

3.2 Related work

The performance of SVC signaling in ATM networks has only received limited attention in the literature. One notable attempt at quantifying the signaling overhead through an empirical study is found in [55]. In contrast, much work has been aimed at VP capacity allocation in a VP-only switching environment, e.g. [19, 50, 57, 61]. The material in Sec. 3.4.1 is based primarily on the work in [19]. Additional background material on implied costs can be found in [36]. Very little research has addressed adaptive VP capacity allocation in a VP/VC switching environment where call processing capabilities are a limiting factor. The only other research in this particular setting that we are aware of is found in [4]. In [41], a global tradeoff between call processing costs and multiplexing gain is considered while jointly designing the VP layout, capacity allocation, and routing. Our particular formulation in Sec. 3.4.2 based on signaling flow constraints and implied costs appears to be new. To our knowledge, the performance of the general VP capacity migration problem which we address in Sec. 3.5 has not been previously studied. Part of the material in that section has been previously presented [54].

3.3 SVC signaling: loads, complexity, and technology

Herein we will focus on the signaling overheads and processing required to establish Switched Virtual Circuits (SVCs) in high-speed networks. There are of course a variety of other signaling tasks that will load up the network resources and will eventually be of concern. For example, a recent study of the Private Network-Network Interface (PNNI) signaling overheads required in advertising available capacity in the network suggested it would totally overload the channel [62].

Consider a 155 Mbps link. It could in principle support at least 2500 voice

Signal	Line rate	Call processing rates
STS-1	51.84 Mbps	3 cps
STS-3	155.52 Mbps	10 cps
STS-12	622.08 Mbps	43 cps
STS-48	2.488 Gbps	172 cps
STS-192	9.953 Gbps	691 cps

Table 3.1: Call processing rate requirements vs. bandwidth growth.

connections.[1] Assuming the average connection duration is 3 minutes, then the maximum average call processing rate that might be required for such a link would be roughly 14 calls per second (cps). For higher line rates, with the same link utilization, the necessary call processing throughput would eventually scale proportionally with bandwidth.[2] Thus, for example, Table 3.1 shows the average signaling rates that would need to be sustained for various links given the following assumptions: they are operated at 80% utilization, voice connections require 64 Kbps each, and each connection has a mean holding time of 3 minutes. These numbers give a rough idea of the manner in which call processing rates would grow with bandwidth for links carrying homogeneous uncompressed continuous bit rate voice traffic.

Current ATM switches typically support call processing using a shared general purpose processor. High-end systems can achieve throughputs on the order of 100–215 cps. Thus a switch supporting 16 OC-12 links, subject to the above loading assumption, could find that the required call processing rate, 688 cps, rather than bandwidth is the system bottleneck. This bottleneck can be overcome with dedicated special-purpose call processors per line card, but the downside is the increased difficulty in making upgrades and the possible increased cost compared to a software-only solution.

Several factors affect the call processing loads that switches will see. In the next sections we consider the role of various types of applications, the manner in

[1] Some of todays ATM core switches support voice using various compression techniques and silence detection to extract the benefits of statistical multiplexing and substantially increase the number of connections that can be supported.

[2] Statistical multiplexing would increase the number of concurrent connections that can be admitted more than linearly as the line rates increase, but as they achieve truly high capacities, the growth would eventually become linear.

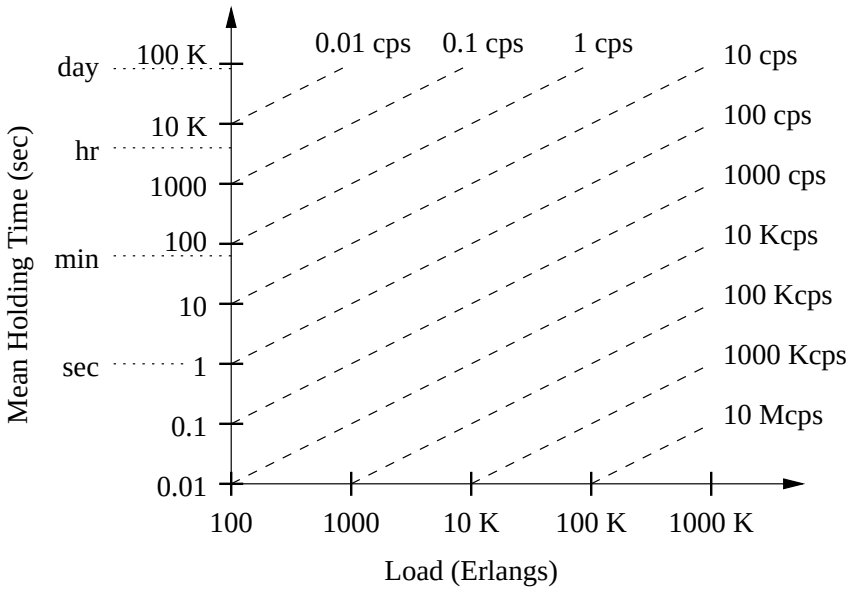


Figure 3.1: Call processing rates versus demand and mean holding times.

which the technology and signaling requirements will scale in a full-blown ATM network, and the issue of congestion in the signaling network.

3.3.1 What are the SVC hungry applications?

Different applications will place markedly varying burdens on the system. The graph shown in Fig. 3.1 gives an idea of how various loads will impact the system. It exhibits the call processing load in cps versus the demand, measured in Erlangs, and the mean holding times of connections.[3] For our purposes here, we shall assume that there are sufficient network resources to meet most of the demand. In other words, we ignore blocked calls which would in fact only further increase the call processing loads on the system. Consider, for example, video applications having a mean holding time of about 1 hour. If the overall system sustains an average load of about 10,000 connections, then the call processing load would not exceed 3 cps. In contrast, consider data applications such as web browsing with mean holding times

[3]Note the demand, measured in Erlangs, i.e., average connection requests per mean connection holding time, is also a rough estimate for the average number of concurrent connections in the system.

not exceeding a minute and a total average load of 100,000 connections. In this case the resulting call processing loads could easily exceed 1000 cps. Even shorter holding times would be associated with the transfer of say a single IP packet, and such traffic would far exceed the setup rates that are currently feasible. Below we comment on these examples in somewhat more detail.

Video-on-demand and other video applications are considered to be among the possible "killer" application for broadband ATM networks. Because the holding times of video connections are relatively long, the call setup loads would be relatively small. Thus, only if database video applications, requesting say video clips or still images, or smaller sized video programs become popular, will the signaling load become significant. Note however that if adaptive resource allocation was to be used on a per connection basis, as proposed in the RCBR protocol [27], this conclusion would change dramatically.

Voice calls are estimated to have a load of 0.1 Erlangs per line with a typical voice connection having a mean holding time of 3–5 minutes, which means an average of 0.00042 cps per line. Assuming that roughly 100,000 lines are supported at a central office switch, this translates to 42 cps. Core switches will however see far greater rates since they support much higher multiplexed rates. Videoconferencing calls should also be put in this category since they have similar holding times. These types of calls could significantly increase the call processing loads because they are typically multicast to several participants.

Data is perhaps the most "SVC hungry" application. Supporting LAN Emulation (LANE) or Multiple Protocol over ATM (MPOA) to carry IP traffic over ATM would result in significant signaling loads. In the worst case, a user clicking on an icon translates to the underlying ATM layer setting up an SVC to the destination router or server. This is the so-called SVC cut-through approach. Since the demand per user is likely to be quite high, these solutions to IP-over-ATM are unlikely to scale up in the public WAN environment without taking appropriate management steps to reduce complexity. A possible solution is a topology-driven approach such as Cisco's tag switching in which data is sent on SVCs already set up along routes computed by the Internet protocols.

In summary, the average demands for call processing will depend on the overall demand, the connection holding time, and the capacity of the system. Of particular concern today would be connections having short holding times, on the order of seconds or minutes, and high demand, such as voice or transaction processing loads that future business applications might require.

3.3.2 Dimensioning large networks

Consider the following simplified scenario. Suppose that the overall load on the system is increasing linearly, i.e., as $N\rho$. In dealing with increasing loads, network service providers can essentially use two extremes in developing their infrastructure. At one extreme, the provider could simply increase the capacity and/or number of ports of its switches so as to meet the demand. In this case the total signaling load seen by these large core switches grows linearly since each switch will see a fraction of the total load $N\rho$ on the network. With this solution, if the call processing resources do not increase in speed linearly, they will eventually become the bottleneck of the system.

At the other extreme, one can build increasingly complex meshes by adding more switches to the network, say linearly in N, so as to increase its carrying capacity. Assuming an optimal hierarchical configuration, the number of switches traversed by a connection would grow logarithmically, i.e., as $\log N$. Assuming that the load is spread uniformly on the network, then the total load on a given switch would be proportional to

$$N\rho \times \frac{\log N}{N} = \rho \log N.$$

With this approach, a switch's call processing capacity would only need to grow logarithmically with the load. However, given the increasingly complex network, there is no guarantee that the processing requirements would not increase with the number of switches in the system. Indeed, in [55], based on an empirical study, the authors suggest that the bulk of the processing delay occurs in a small body of code for which the processing time depends on the size of the network. That is, the processing rates are likely to depend on the network size, further limiting the capacity of the switching system.

The call processing requirements of a real system might then scale somewhere between the desirable logarithmic growth and the worst case linear growth.

This brings us to the following question: what can technology today achieve?

3.3.3 Technology

Whether ATM switches will be able to support a linear increase in speed for their call processing capacity is at this point difficult to predict. The type of processing required at connection setup can in fact be significant and may span several switching systems and require additional updates during the lifetime of a connection. Tasks spanning the gamut from call admission and assessment and negotiation of QoS requirements, to routing, will be increasingly complex in multiservice networks, and thus one would expect them to place increasingly higher loads on call processing resources. Furthermore significant processing loads would be associated with performance and usage monitoring as well as billing records.

There is little experience operating ATM networks today, and since the performance characteristics of signaling channels depend on a complex interaction between software and hardware resources in a distributed environment, they are difficult to assess [55]. Nevertheless, some relevant data points give a starting point. Current workgroup switches, such as the ForeRunnerLE 155, advertise a connection set up time of 10 msec, that is 100 cps. Core switches range from 215 cps for the ForeRunner ASX-200BX to 3000 cps for the Ascend CBX 500 where the call processors are implemented in hardware with a separate processor card per port. Note however that a switch might need to support full accounting capabilities, such as the generating of billing records on a per connection basis, therefore it is difficult to extrapolate what throughput might be achieved.

While it is possible to achieve a call setup rate on the order of thousands of cps per switch with the help of call processors implemented in hardware, it is hard to scale these speeds even further without running into the high costs that such systems represent. Moreover, even if the desired speed could be achieved on general purpose workstations, it is questionable that supporting the full flexibility of SVC setup in software will be worthwhile given the increased software and maintenance costs. Indeed with the advent of multiservice broadband networks, the complexity and maintenance cost of call processing software is likely to increase, and in order to control costs it may be advantageous to limit the amount of signaling and call processing performed in the network.

3.3.4 Dealing with congestion in signaling networks

Looking at the average throughput that network elements can sustain is appropriate as a first cut. However, since call requests arrive as stochastic processes, the variability in the rate of requests is likely to lead to queuing and setup delays, and congestion will eventually lead to lost calls. Thus one eventually needs to take appropriate action in controlling demand and signaling flows in the network so as to reduce focused congestion and allocate resources fairly.

As a reference case, current Intelligent Network (IN) services typically have much greater processing requirements than Plain Old Telephone Service (POTS). Indeed, IN services might require a database lookup and authentication in addition to routing and call admission. With the advent of multiservice networks, the requirements at call setup, in terms of call admission, authentication, QoS negotiation, billing, etc., will far exceed those of current systems.

A significant amount of effort has yet to be invested in optimizing protocols and implementations to enhance performance. Some studies suggest that much work will need to be carried out before definitive performance comparisons and numbers can be collected [32, 55].

3.4 Controlling processing and complexity costs via VP networks

By configuring VPCs, the network manager can reduce the processing requirements placed on intermediate switching nodes in the backbone by deferring call processing to a VPC's end nodes [20]. The cost of this approach is a possible loss of efficiency since capacity would now be segregated per VPC. Thus the service provider needs to find a compromise between limiting the processing load on network nodes and maximizing the large-scale network's throughput/revenue.

In fact one can view the virtual path layer as an intermediate resource allocation layer where allocation decisions are made on a slow time scale but coordinated over the network [19]. We contend that by allowing adaptive resource allocation for VP networks based on demand estimates and/or on-line estimates, possible losses in efficiency resulting from limiting the resources available for multiplexing SVC connections may be overcome. The configuration of VP networks is likely to become

Switching	VP	VP/VC
VPs	single- or multiple-link	end-to-end
Algorithm	centralized (periodic)	decentralized (incremental)

Table 3.2: A taxonomy of approaches to VP allocation.

an effective strategy for not only limiting signaling requirements in the network core, but also, and in some cases more importantly, it will enable network operators to handle the increasing complexity of multiservice networks, e.g., by simplifying connection admission control, routing, and provisioning of QoS requirements, and by aiding in addressing network reliability concerns as well as a variety of traffic management issues.

A variety of approaches are available for allocating VPs. In Table 3.2, the approaches are classified according to the type of switching available in the network, the extent of the VPs, and the structure of the allocation algorithm, resulting in eight distinct combinations. In Sec. 3.4.1, we discuss algorithms proposed for a VP switching environment, and in Sec. 3.4.2, we present our original algorithms for a VP/VC switching environment.

3.4.1 VP switching

In this section, we consider a core network that only switches VPs, and we present a method to periodically adjust the VP capacities. We will first treat the case of single- or multiple-link VPs and a centralized algorithm that is run periodically. This setup has been previously considered in [19]. We assume the routing is fixed or quasi-static with load sharing occurring between routes connecting a particular source/destination pair. The physical network consists of a set $\mathcal{J}$ of links with capacities C_j for $j \in \mathcal{J}$. The logical network is defined by a set $\mathcal{V}$ of virtual paths. The matrix $V = (V_{jv}, j \in \mathcal{J}, v \in \mathcal{V})$ is a 0–1 matrix with $V_{jv} = 1$ if VP v passes through link j and $V_{jv} = 0$ otherwise. By assuming the existence of $\mathcal{V}$, we have bypassed the VP layout problem, a difficult problem in its own right that has been addressed elsewhere, e.g. [12, 23]. We are assuming that the layout would change infrequently. For simplicity, we start with a single-service loss network model, i.e.,

all calls require unit bandwidth, call holding times are independent (of all earlier arrival times and holding times) and identically distributed with unit mean, and blocked calls are lost.[4]

Each VP can be considered to be a logical link from the point of view of routing, and in fact, we define a route (VPC) $r \in \mathcal{R}$ to be a collection of VPs from $\mathcal{V}$. The matrix $A = (A_{vr}, v \in \mathcal{V}, r \in \mathcal{R})$ is a 0–1 matrix with $A_{vr} = 1$ if route r passes through VP v and $A_{vr} = 0$ otherwise. Define the vector $x = (x_v, v \in \mathcal{V})$ to be the capacities allocated to each VP. The vector $\nu = (\nu_r, r \in \mathcal{R})$ denotes the rates of independent Poisson arrival processes for each route. We use w_r to denote the revenue generated by accepting a connection on route r, and L_r is the blocking probability on route r.

Our goal is to maximize the rate of network revenue, so the optimization problem can be stated as

$$\begin{aligned} \text{maximize} \quad & W(\nu; x) = \sum_{r \in \mathcal{R}} w_r \nu_r (1 - L_r) && (3.1) \\ \text{subject to} \quad & Vx \leq C && (3.2) \\ \text{over} \quad & x \geq 0. \end{aligned}$$

The constraint (3.2) is a physical capacity constraint on the VPs.

To calculate the revenue sensitivities, we must first find the blocking probability L_r for each route r, an important performance measure in its own right. Steady-state blocking probabilities can be obtained through the invariant distribution of the number of calls in progress on each route. However, the normalization constant for this distribution can be difficult to compute, especially for large networks. Therefore, the blocking probabilities are usually estimated using the Erlang fixed point approximation [26, 38].

Let $B = (B_v, v \in \mathcal{V})$ be the solution to the equations

$$B_v = E(\rho_v, x_v), \quad v \in \mathcal{V}, \tag{3.3}$$

[4]One realistic example of a single-service environment is a single-class embedded network. Alternatively, our model is roughly equivalent to a network with very high bandwidth links where the real resource constraint is that of labels (e.g., virtual path or virtual circuit identifiers) for connections on links. The unit bandwidth requirement per call can be considered to be an *effective bandwidth* [16, 40].

where

$$\rho_v = \sum_{r \in \mathcal{R}} A_{vr} \nu_r \prod_{u \in r - \{v\}} (1 - B_u) \tag{3.4}$$

and the function E is the Erlang B formula [8]

$$E(\rho_v, x_v) = \frac{\rho_v^{x_v}}{x_v!} \left[\sum_{n=0}^{x_v} \frac{\rho_v^n}{n!} \right]^{-1}. \tag{3.5}$$

The vector B is called the Erlang fixed point; its existence follows from the Brouwer fixed point theorem and uniqueness was proved in [35]. Using B, an approximation for the blocking probability on route r is

$$L_r \approx 1 - \prod_{v \in r} (1 - B_v). \tag{3.6}$$

The idea behind the approximation is as follows. Each Poisson stream of rate ν_r that passes through VP v is thinned by a factor $1 - B_u$ at each VP $u \in r - \{v\}$ before being offered to v. Assuming these thinnings are independent both from VP to VP and over all routes, then the traffic offered to VP v is Poisson with rate ρ_v as given in (3.4), the blocking probability at VP v is B_v as given in (3.3), and the loss probability on route r is exactly L_r as given in (3.6).

Starting from the Erlang fixed point approximation and by extending the definition of the Erlang B formula (3.5) to non-integral values of x_v via linear interpolation,[5] the sensitivity of the rate of revenue with respect to the capacity of VP v has been derived by Kelly [36] and is given by

$$\frac{\partial}{\partial x_v} W(\nu; x) = c_v \tag{3.7}$$

where the implied costs c are the (unique) solution to the equations

$$c_v = \eta_v (1 - B_v)^{-1} \sum_{r \in \mathcal{R}} A_{vr} \nu_r (1 - L_r)(w_r - \sum_{u \in \mathcal{V}} A_{ur} c_u + c_v), \quad v \in \mathcal{V}. \tag{3.8}$$

In these equations, $\eta_v = E(\rho_v, \lceil x_v - 1 \rceil) - E(\rho_v, \lceil x_v \rceil)$; B_v, ρ_v, and L_r are obtained from the Erlang fixed point approximation. Note that, based on (3.7), the implied costs c also have an interpretation as shadow prices.

[5] At integer values of x_v, define the derivative of $E(\rho_v, x_v)$ with respect to x_v to be the left derivative.

Using the partial derivatives of the rate of network revenue (3.7), the optimization problem can be solved (at periodic intervals) using a standard gradient-based hill-climbing procedure over a convex feasibility region defined by linear inequalities. An execution of this centralized algorithm could be triggered by a combination of a time limit and a window for the revenue function $W(\nu; x)$. If the current rate of revenue reaches the upper or lower limits of the window or the time limit is reached, then a new optimization would be executed. Subsequently, a new window would be computed based on the current rate of revenue. Upper and lower bounds for $W(\nu; x)$ can be computed [19, 52] giving an idea of how we are doing. These bounds are only valid for the particular VP layout being used; a more difficult problem is to find tight bounds over all possible VP layouts. Bounds of this type would help to gauge when the VP layout should be adjusted. An alternative would be to adjust the layout on a daily or hourly basis with the VP capacities adjusted more frequently for fine-tuning using the algorithm proposed here. We note that, in general, $W(\nu; x)$ is not concave. However, Kelly has shown that it is asymptotically linear as ν and x are increased in proportion [36], and it is possible that the stochastic fluctuations in the offered traffic may allow the optimization procedure to escape a nonoptimal local maximum.

If the VPs are constrained to be end-to-end, then each route traverses a single logical link. Since we are allocating end-to-end VPs, the blocking probability $L_r = E(\nu_r, x_v)$ for the VP v such that $A_{vr} = 1$. Therefore, $\frac{\partial}{\partial x_v} L_r = A_{vr}(E(\nu_r, \lceil x_v \rceil) - E(\nu_r, \lceil x_v - 1 \rceil)) = -A_{vr}\eta_v$, and the revenue sensitivity is

$$\frac{\partial}{\partial x_v} W(\nu; x) = \eta_v \sum_{r \in \mathcal{R}} A_{vr} w_r \nu_r. \tag{3.9}$$

Compared to (3.7), this sensitivity can be computed independently of the other VPs.

In a decentralized algorithm, each VP would be responsible for computing its own sensitivity. In the single- or multiple-link VP case, this would involve exchanging values with other VPs in an iterative fashion. Once computed, the sensitivities would be used to guide VP capacity allocation and deallocation requests made to link controllers. The link controllers would enforce the link capacity constraints and grant or deny requests as appropriate based on the revenue sensitivities.

3.4.2 VP/VC switching

In this section, we address the problem of a network that can simultaneously switch VPs and VCs where the constraint on call processing capacity is a limiting factor. As argued above, this situation may or may not materialize in the future. If it does, the underlying tradeoff is between network efficiency, which we represent via the overall network revenue or throughput, and the switch call processing capacity. Based on models for circuit-switched networks, we formulate this problem as one of maximizing network revenue subject to call processing constraints. There is at least one similar formulation [4], where they use an M/M/1 model to approximate call setup delays. However, our goal is to design an adaptive mechanism that, based on on-line measurements of the network loads, slowly dimensions and configures VPs to enhance network performance.

Our framework in this section consists of a set $\mathcal{J}$ of links with capacities C_j for $j \in \mathcal{J}$, a set $\mathcal{R}$ of routes which are used for switched virtual circuits, and a set $\mathcal{V}$ of routes corresponding to virtual paths. As before, we start with a single-service loss network model. The initial algorithm that we present will be for end-to-end VPs in a decentralized setting.

Define the vector $x = (x_v, v \in \mathcal{V})$ to be the capacities allocated to each VPC. The matrix $A = (A_{jr}, j \in \mathcal{J}, r \in \mathcal{R})$ is a 0–1 matrix with $A_{jr} = 1$ if route r passes through link j and $A_{jr} = 0$ otherwise. The matrix $V = (V_{jv}, j \in \mathcal{J}, v \in \mathcal{V})$ is the analogous 0–1 matrix for the VPCs. The residual capacities left to the SVCs are given by the vector $C^s = C - Vx$. The vector $\nu = (\nu_r, r \in \mathcal{R})$ denotes the rates of independent Poisson arrival processes for each route. The vector $\xi = (\xi_v, v \in \mathcal{V})$ is the analogous demand vector for the VPCs. We use w_r (w_v) to denote the revenue generated by accepting a connection on route r (VPC v), and L_r (L_v) is the blocking probability on route r (VPC v). There may be multiple VPCs and routes available for SVCs between each origin/destination pair, so to retain the Poisson assumption, we will assume that load sharing occurs between the choices available for each origin/destination pair. The percentage routed to the available VPCs is determined separately from our algorithm for adjusting the VPC capacities. Note that this type of routing coupled with the loss network assumption provides an upper bound on the amount of blocking that would occur with a more realistic alternate routing scheme (in which the VPCs are attempted first). Hence, our estimate for the rate of revenue generated is conservative.

To model the signaling constraint, we define a vector $\mu = (\mu_j, j \in \mathcal{J})$ representing the service rate available for setting up SVCs on a per link (or, equivalently, per port) basis. The signaling capacity of each switch is in proportion to the number of input (or output) links, so we simply take the average available to each link. We will define the signaling constraint as a flow constraint μ on the amount of traffic desiring SVCs.

For a given ν and ξ (and a uniform amount of revenue per connection), the rate of revenue for the network will be maximized when no capacity is allocated to VPCs because allocating capacity for an end-to-end VPC reduces the amount of capacity available (and thus increases blocking) for SVCs using those links. However, we will be forced to allocate capacity to a VPC if the signaling constraint is violated at one or more of the switches. Our optimization problem can be stated as

$$\begin{aligned}
\text{maximize} \quad & W(\nu;\, \xi;\, x;\, C) = \sum_{v \in \mathcal{V}} w_v \xi_v (1 - L_v) + \sum_{r \in \mathcal{R}} w_r \nu_r (1 - L_r) && (3.10)\\
\text{subject to} \quad & Vx \leq C && (3.11)\\
\text{and} \quad & A\nu \leq \mu && (3.12)\\
\text{over} \quad & x \geq 0.
\end{aligned}$$

The second constraint (3.12) is the signaling constraint for SVCs. We have conservatively chosen to constrain the offered load (without thinning) to be less than the signaling capacity. If this constraint is violated, then the delays for call setup are becoming unacceptable at the affected switches. We note that, in general, $W(\nu;\, \xi;\, x;\, C)$ is not concave with respect to the VPC capacities x because of the second term in (3.10).

The revenue sensitivity with respect to the VPC capacity x_v can be derived as follows. In the first term of (3.10), only L_v depends on x_v. Since we are allocating end-to-end VPCs, the blocking probability $L_v = E(\xi_v, x_v)$ where the function E is the Erlang B formula. As before, we extend the Erlang B formula to non-integral values of x_v via linear interpolation, and, at integer values of x_v, we define the derivative of $E(\xi_v, x_v)$ with respect to x_v to be the left derivative. Therefore, $\frac{\partial}{\partial x_v} L_v = E(\xi_v, \lceil x_v \rceil) - E(\xi_v, \lceil x_v - 1 \rceil) = -\eta_v$.

To treat the second term of (3.10), we note that the residual capacities C^s together with the demand vector ν comprise a loss network in which we can compute

implied costs according to [36]

$$c_j^s = \eta_j^s (1 - B_j^s)^{-1} \sum_{r \in \mathcal{R}} A_{jr} \nu_r (1 - L_r)(w_r - \sum_{k \in \mathcal{J}} A_{kr} c_k^s + c_j^s), \quad j \in \mathcal{J}, \tag{3.13}$$

where $\eta_j^s = E(\rho_j, C_j^s - 1) - E(\rho_j, C_j^s)$, and ρ_j and B_j^s are the offered load and blocking probability for SVCs at link j, respectively. B_j^s, ρ_j, and L_r are obtained from either the Erlang fixed point approximation or on-line measurement.

In [36], for a revenue function $W^s(\nu;\, C^s) = \sum_{r \in \mathcal{R}} w_r \nu_r (1 - L_r)$, Kelly derived the sensitivity of the rate of revenue with respect to C_j^s to be c_j^s. Noting that $\frac{\partial}{\partial x_v} W^s(\nu;\, C^s) = \sum_{j \in \mathcal{J}} \frac{\partial}{\partial x_v} C_j^s \frac{\partial}{\partial C_j^s} W^s(\nu;\, C^s)$ and combining this with our result for the first term, we have

$$\frac{\partial}{\partial x_v} W(\nu;\, \xi;\, x;\, C) = w_v \xi_v \eta_v - \sum_{j \in \mathcal{J}} V_{jv} c_j^s. \tag{3.14}$$

The first term in (3.14) represents the increase in revenue that would result from additional calls accepted on VPC v after increasing x_v, while the second term quantifies the loss in revenue that would occur from decreasing the residual capacity along path v available to SVCs.

Our on-line algorithm for adjusting the VPC capacities can be described intuitively as follows. Each switch (port) monitors the processing loads generated by each route. If the aggregate offered load is seen to reach a link's signaling constraint, then a VPC is initiated. To determine which flow of traffic to aggregate, the switches coordinate a calculation of the revenue sensitivities with respect to allocating capacity to the possible VPCs flowing through the overloaded link.[6] A VPC is configured on the route with the highest revenue sensitivity, i.e., likely to cause the least loss of network revenue. This procedure is carried out slowly and adaptively to meet changing demands. As the call processing loads on the network relax, it becomes advantageous to eliminate VPCs allowing for better usage of network resources. In this case a lower threshold on measured loads would initiate the removal of VPCs. We propose a mechanism incorporating a hysteresis loop with two thresholds in order to avoid making excessive changes due to fluctuations of the network processing loads. In addition, the actual measurements of the processing loads can be damped by exponential weighted averaging.

[6]Note that this requires coordination because the revenue sensitivities depend on the network topology, routes, and loads at that moment in time.

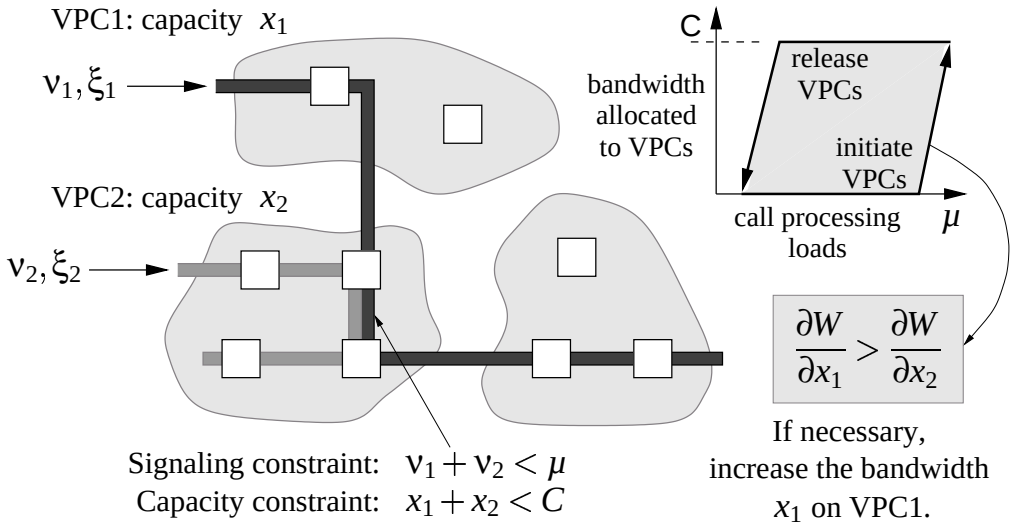


Figure 3.2: VP allocation scenario with two VPCs subject to call processing constraints.

A simple illustration of this process is shown in Fig. 3.2 for a link with two possible VPCs. In this particular case, the revenue sensitivity for VPC 1 is higher, so it would be chosen first for capacity allocation as the call processing loads at the link increase.

We now define more formally a decentralized protocol for adaptation. We assume that each node in the network has complete topology and route information and that a decentralized computation of the implied costs for the residual SVC network is carried out at appropriate intervals. Initially, $x = 0$, i.e., no capacity is allocated to VPCs. For each link, there is a link controller (or port monitor) which is responsible for checking the signaling constraint. If the constraint is violated for a particular link, then the controller computes the revenue sensitivities for all VPCs which are alternate routes for routes in $\mathcal{R}$ passing through this link. Increase the capacity by a unit amount (subject to the capacity constraint) of the VPC with the highest sensitivity subject to not passing through a bottleneck link j with $\sum_{v \in \mathcal{V}} V_{jv} x_v = C_j$. After allowing the load vectors ν and ξ to readjust, repeat as necessary until the signaling constraint at that link is not violated (assuming this is feasible). Capacity should be removed in unit decrements from a VPC v when $\sum_{r \in \mathcal{R}} A_{jr} \nu_r < \mu_j - \varepsilon_j$ for all $j \in v$ where $\mu_j - \varepsilon_j$ is a predetermined threshold.

If the VPs are not constrained to be end-to-end, then routes can use a mixture

of VCs and VPs where multiple-link VPs are treated as single logical links. Implied costs can be computed for all "links" whether they be VPs or actual links with a residual capacity $C_j^s = C_j - \sum_{v \in \mathcal{V}} V_{jv} x_v$. The revenue sensitivity with respect to the capacity of VP v would be

$$\frac{\partial}{\partial x_v} W(\nu;\, x;\, C) = c_v - \sum_{j \in \mathcal{J}} V_{jv} c_j^s. \tag{3.15}$$

It is unclear whether the algorithm to adjust the VP capacities would work as well as the end-to-end VPC case in limiting the call processing loads for SVCs. Increasing the capacity of a VP may not increase the capacity available to routes using that VP because of bottlenecks elsewhere. For this reason, it may be wise to restrict the candidate VPs for an increase to those passing directly through the link with the signaling constraint violation. This would increase the probability of a VP capacity increase actually decreasing the SVC setup traffic through that link.

Another alternative is to run a centralized algorithm at periodic intervals. For instance, given future demand estimates and average holding times, the average cps per link can be constrained below a certain value by limiting the capacity available to SVCs. The remaining capacity in the network can be allocated to VPs using the method proposed in Sec. 3.4.1 for a VP-only environment. The algorithm could then be repeated periodically to adapt to changing demands.

3.4.3 Additional remarks

A multiservice environment could accommodate traffic from several different bandwidth classes. The implied costs would now be indexed by (logical) link and service type, and with the choice of an appropriate blocking function at each link, the implied cost equations can be derived by extending the approach of Kelly to the multirate case [19, 52]. The formulation in [19] is particularly relevant here since they consider a situation similar to that in Sec. 3.4.1. We will have more to say about the multirate case in Chapter 4 when we discuss network aggregation.

Our goal here was to present a framework for adapting VP capacities in a variety of settings. Establishing an optimal VP layout and set of capacities is a hard combinatorial problem, and no comprehensive solution has been presented to date. We have tried to decouple the problem as much as possible from the effects of the routing algorithm, but in fact, there are subtle interactions between the routing and

VP capacity allocation algorithms that affect the ultimate success of the methods proposed here.

3.5 VP capacity migration

If we run a centralized algorithm to update the VP capacities, then we are faced with the following problem: given two sets of VP capacities, how do we migrate from one set to another? This may involve changing the capacities of the current VP layout or changing to an entirely different layout.[7] As discussed previously, such changes may be triggered by a centralized optimization that is run periodically for the purpose of allocating resources more efficiently [50]. They may also be carried out in response to a failure in the network, in which case, enough spare (unallocated) capacity must be available to handle the reconfiguration. Call priority levels might be used to determine which calls are reconnected if there is not enough room to accommodate all calls which were dropped due to the failure. For a non-failure related reconfiguration, it is reasonable to assume that ongoing calls cannot be preempted. As we will see in the sequel, this assumption admits the possibility of migration taking a significant amount of time.

We can divide the VPs that must change capacities into two groups, $\mathcal{V}_d$ and $\mathcal{V}_i$, consisting of those that must decrease their capacities and those that must increase their capacities, respectively. For a given VP v, let x_v denote its current capacity, x_v^t its target capacity, and y_v its current utilization. For each $v \in \mathcal{V}_d$, we assume that ongoing calls cannot be preempted and that no new calls are admitted until the decrease in capacity is accomplished. For $v \in \mathcal{V}_d$ such that $y_v > x_v^t$, the VP must wait for enough calls to complete to make $y_v \leq x_v^t$ before requesting the decrease in capacity. (It could also make changes incrementally.) VPs in $\mathcal{V}_i$ can still admit calls as usual.

In the migration algorithms that we will present, it is assumed that links and VPs have associated "intelligences" that enable communication between the links

[7] We will say that we are changing layouts if at least one VP is being created or destroyed. This distinction is, in reality, arbitrary as the same algorithm is used for migration whether or not the layout is being changed. We make the distinction because changing VP layouts will, in general, take longer than incremental capacity adjustments, and in practice, it is expected that the VP layout will be changed less frequently than the VP capacities, as the creation or destruction of VPs causes additional overhead.

and VPs. In addition, for a centralized optimization, we must have the capability to broadcast the new capacities to existing VPs and arrange for the creation of new VPs.

3.5.1 Basic algorithm

A simplistic algorithm for accomplishing the migration is to first broadcast the new capacities to VPs in $\mathcal{V}_d$, wait for all of them to decrease, and then broadcast the new capacities to VPs in $\mathcal{V}_i$. After presenting pseudocode for this algorithm, we will show by way of example that it can be extremely inefficient.

Algorithm 3.1.

```
1  begin
2     foreach v ∈ V_d do x_v := max(y_v, x_v^t); endfor;
3     while { v ∈ V_d | x_v > x_v^t } ≠ ∅ do
4           wait for a call completion;
5           r := route on which call completed;
6           foreach v ∈ r such that x_v > x_v^t do
7              x_v := x_v − 1;
8           endfor;
9     endwhile;
10    foreach v ∈ V_i do x_v := x_v^t; endfor;
11 end
```

3.5.2 Migration example

Consider a homogeneous situation in which all calls require unit bandwidth and holding times are independent and exponentially distributed with mean μ^{-1}. Suppose n of the VPs in $\mathcal{V}_d$ have $y_v - x_v^t = 1$ while the remaining VPs in $\mathcal{V}_d$ have $y_v \leq x_v^t$. For each VP, the time until one call completes is exponentially distributed with mean $(y_v\mu)^{-1}$. For simplicity, assume $y_v = y$ for all VPs with $y_v > x_v^t$. Let T be a random variable denoting the time until all VPs in $\mathcal{V}_d$ can decrease capacity to

x_v^t. Then,

$$\begin{aligned}\mathbb{E}[T] &= \frac{1}{ny\mu} + \frac{1}{(n-1)y\mu} + \cdots + \frac{1}{y\mu} \\ &= \frac{1}{y\mu}(\frac{1}{n} + \frac{1}{n-1} + \cdots + 1) \\ &\approx \frac{1}{y\mu}(\log n + 0.5772) \quad \text{for large } n,\end{aligned}$$

where 0.5772 is Euler's constant [64]. However, the tail of the distribution of T is given by

$$\begin{aligned}\mathbb{P}(T > t) &= 1 - \mathbb{P}(T \leq t) \\ &= 1 - (1 - e^{-y\mu t})^n \quad \text{by independence} \\ &\approx ne^{-y\mu t} \quad \text{for large } t.\end{aligned}$$

Therefore, for large n, the mean migration time grows logarithmically in n, but the tail probabilities, for a large, fixed time t, grow linearly in n. For holding time distributions other than exponential, the tail can potentially grow faster or slower. For example, it grows faster for a Pareto distribution. These growth rates and limiting distributions are studied more carefully in [21].

The primary observation here is that in large networks where significant adjustments are being made, we might on occasion have very poor "performance." By that, we mean a significant number of connections which were blocked during migration could have been admitted under a migration process more sophisticated than Algorithm 3.1. The migration process itself is a transient occurrence, yet if the duration is lengthy and/or updates on VP layouts are made quite often, the overall impact on performance may be significant. Roughly speaking, we are likely to have significant blocking during migration when holding times are long, e.g., there are lots of video connections, and/or there are lots of VPs being decreased to a small capacity, i.e., when n is large and $y\mu$ is small. As we will see in the experiments below, the situation can potentially be very bad if we are making dramatic changes in the VP layout.

3.5.3 Refinements to the basic algorithm

Improved algorithms require more coordination between VPs and links as they attempt to increase the capacity of VPs in $\mathcal{V}_i$ before all VPs in $\mathcal{V}_d$ have decreased

to their target levels. The hope is that this will make a difference as the expected time until the first call completes is much less than $\mathbb{E}[T]$. In the example above with the n VPs and $y_v - x_v^t = 1$, the average time until the first call completion is $1/ny\mu$, which is much less than $\mathbb{E}[T]$ for large n. The performance metric of interest is how many extra connections are admitted that would have otherwise been blocked. Note that our improved algorithms will have the same migration time as Algorithm 3.1 because (ignoring propagation and processing delays) the length of the migration process is determined solely by the time it takes for enough ongoing calls to complete to bring the utilization levels of all VPs in $\mathcal{V}_d$ below their target capacities.

We now briefly outline two enhanced algorithms that can be implemented in a decentralized fashion. In the following, let $x_{v;\,j}$ denote the capacity allocated to VP v at link j, and let $\mathcal{V}_{i;\,j}$ be the set of VPs in $\mathcal{V}_i$ that pass through link j. Note that that the actual capacity of VP v is $x_v = \min_{j \in v} x_{v;\,j}$. Suppose that, at link j, as VPs in $\mathcal{V}_d$ release capacity, available bandwidth units are allocated one at a time to the VP in $\mathcal{V}_{i;\,j}$ with the *maximum remaining increase* to its target capacity, i.e., $\max_{v \in \mathcal{V}_{i;\,j}} (x_v^t - x_{v;\,j})$. Ties are broken in an arbitrary fashion. We have omitted the breaking of ties in the following description and have assumed that calls require unit bandwidth.

Algorithm 3.2.

```
1  begin
2    foreach v ∈ V_d do
3      x_v := max(y_v, x_v^t);
4      foreach j ∈ v do x_{v;j} := x_v; endfor;
5    endfor;
6    foreach j ∈ J do
7      c := spare capacity at link j;
8      while c > 0 do
9            z := arg max_{v ∈ V_{i;j}} (x_v^t − x_{v;j});
10           if x_z^t > x_{z;j} then x_{z;j} := x_{z;j} + 1; x_z := min_{k∈z} x_{z;k}; endif;
11           c := c − 1;
12     endwhile;
13   endfor;
14   while { v ∈ V_d | x_v > x_v^t } ≠ ∅ do
```

```
15          wait for a call completion;
16          r := route on which call completed;
17          foreach v ∈ r such that x_v > x_v^t do
18            x_v := x_v − 1;
19            foreach j ∈ v do
20              x_{v;j} := x_v;
21              z := arg max_{w∈V'_{i;j}} (x_w^t − x_{w;j});
22              if x_z^t > x_{z;j} then x_{z;j} := x_{z;j} + 1; x_z := min_{k∈z} x_{z;k}; endif;
23            endfor;
24          endfor;
25    endwhile;
26    foreach v ∈ V'_i do x_v := x_v^t; endfor;
27  end
```

We can define a more refined algorithm that uses an extra piece of information, the current utilization levels, to try to anticipate which VPs in $\mathcal{V}'_i$ are likely to block soon. In this algorithm, when allocating a unit of bandwidth at link j, we choose the VP in $\mathcal{V}'_{i;j}$ with the minimum available capacity, i.e., $\min_{v\in\mathcal{V}'_{i;j}}(x_{v;j} - y_v)$, given that $x_v^t - x_{v;j} > 0$. Ties are broken by choosing the VP with the maximum remaining increase as in Algorithm 3.2, with further ties broken arbitrarily. The assumption that links know the utilization levels of VPs is a bad one because it breaks the interface abstraction between VPs and links. Nonetheless, it is instructive to see how well this algorithm performs compared to Algorithm 3.2. Once again, we have omitted the breaking of ties in the following description, and instead of repeating most of Algorithm 3.2, we only specify what is different.

Algorithm 3.3. *Substitute the following for lines 9–10 and for lines 21–22 in Algorithm 3.2.*

```
1  z := arg min_{ w∈V'_{i;j} | x_w^t > x_{w;j} } (x_{w;j} − y_w);
2  if z is valid then x_{z;j} := x_{z;j} + 1; x_z := min_{k∈z} x_{z;k}; endif;
```

Various other algorithms can be defined based on whether or not link j is a bottleneck for a particular VP, i.e., whether or not $x_v = x_{v;j}$. The idea would be to only allocate capacity to VPs for which link j is a bottleneck. In our experiments, these algorithms did not perform significantly better than Algorithms 3.2 and 3.3, so we have not included their descriptions here.

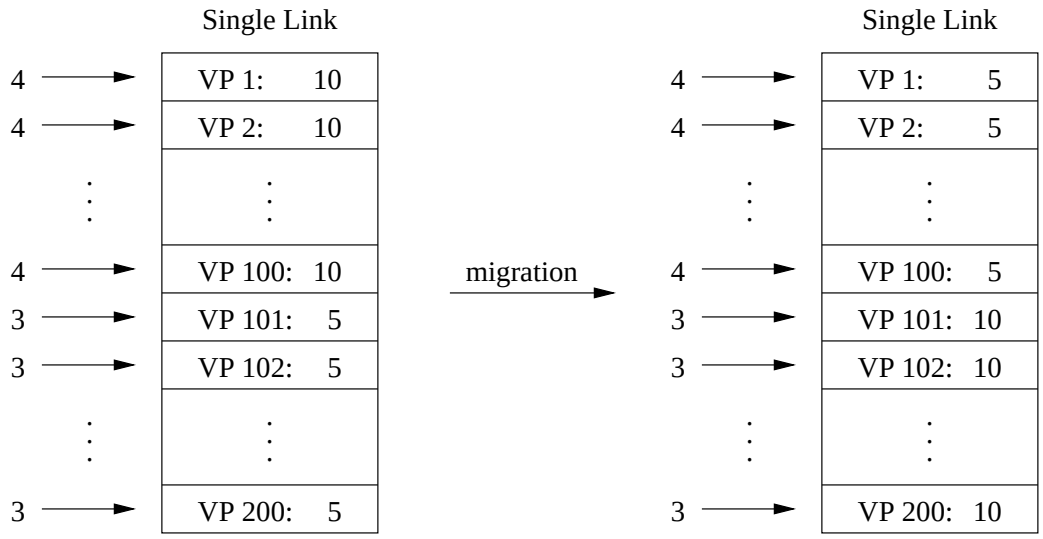


Figure 3.3: An illustration of a single link divided into VPs migrating from capacity 10 to 5 and 5 to 10. Arrivals are Poisson with rates 4 and 3, respectively.

3.5.4 Simulation results

To evaluate the algorithms, we conducted several simulations. We began with a situation resembling the example above. Suppose we have a single link of capacity 1500 supporting a total of 200 VPs, where 100 VPs have capacity 10 and the remaining 100 have capacity 5. The VPs establish a logical (or overlay) network upon which routes are defined using the VPs as logical links. For this single physical link, suppose there are 200 routes, one per VP, with Poisson call arrivals at rate 4 for the VPs with capacity 10 and at rate 3 for the VPs with capacity 5. As before, all calls require unit bandwidth and holding times are independent and exponentially distributed with mean $\mu^{-1} = 1$. During migration, the VPs simply swap capacities, i.e., all VPs with capacity 10 move to 5 and vice versa. This situation is illustrated in Fig. 3.3.

The results quoted below are 95% confidence intervals based on independent replications. Sufficient warmup time was allowed for each replication and common random numbers were used when comparing the various algorithms. For this experiment, which we refer to as Link-Incr, the average migration time was 0.90 ± 0.05 time units, and the total number of offered calls during migration was 631 ± 34 on average. For Algorithm 3.1, an average of 132 ± 8 calls were blocked

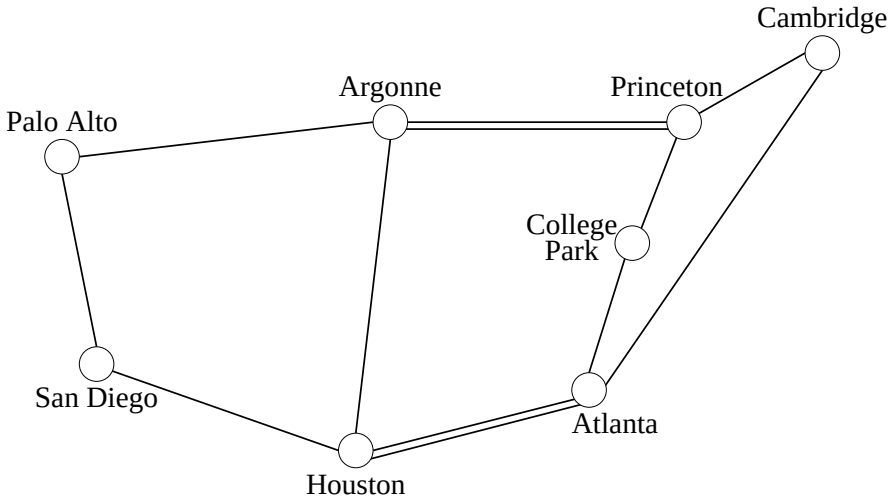


Figure 3.4: Hypothetical core network. The double lines indicate 90 Mbps links. All other links are 45 Mbps.

during migration. For both Algorithms 3.2 and 3.3, 103 ± 7 calls were blocked during migration.

Now consider a similar setup based on changing VP layouts in which the link capacity is 500, 100 of the VPs are decreased from 5 to 0, and the other 100 VPs are increased from 0 to 5. One hundred routes with an offered load of 2.5 each are defined for the VPs currently at capacity 5, and they are immediately shifted to the other 100 VPs when migration commences. For this experiment, which we refer to as Link-Layout, the average migration time jumped to 6.10 ± 0.14 time units. The average number of offered calls during migration was 1628 ± 38, all of which were blocked when using Algorithm 3.1. The number of extra connections admitted was quite large in this example with 198 ± 4 calls being blocked for Algorithm 3.2 and 181 ± 4 calls being blocked for Algorithm 3.3.

We experimented with several larger, multi-link networks, one example of which is shown in Fig. 3.4. This represents a hypothetical core network and is borrowed from [52]. For each pair of endpoints, we defined an end-to-end VP along the shortest path. We also defined a single-link VP for each physical link. We specified three routes for each pair of endpoints: the end-to-end VP and two alternate routes using the single-link VPs. The arrival rates for the routes were inversely proportional to the number of links traversed with shorter routes receiving

greater offered loads. In all, there were 38 VPs and 84 routes. Using this setup, we performed two experiments: the Mesh-Incr experiment which corresponds to a plausible incremental change in VP capacities, and the Mesh-Layout experiment which, in addition to the Mesh-Incr changes, simulates the replacement of the 45 Mbps link between Princeton and College Park with a separate 90 Mbps link. The results of the link and mesh experiments are summarized in Table 3.3.

The performance difference between Algorithm 3.1 and Algorithms 3.2 and 3.3 is especially dramatic when changing layouts. In a sense, the Link-Layout experiment simulates a worst-case scenario since all existing VPs are replaced during migration. The migration time for the Mesh-Incr experiment is so short that hardly any calls are blocked. If the link capacities and number of VPs were increased by an order of magnitude, the number of blocked calls would become significant similar to the Link-Incr experiment. In all cases, Algorithm 3.3 is not significantly better than Algorithm 3.2. We note that, in general, more performance can be gained by our improved algorithms when VPs do not traverse many physical links because to increase its capacity, a VP has to wait for each link in its path to allocate additional capacity to it.

We also performed experiments with holding time distributions having more variance and larger tails than the exponential such as two-stage hyperexponential [49] and Pareto distributions [33]. As expected, migration times can potentially be much longer, especially when changing VP layouts, making the case for implementing improved algorithms even stronger. For example, we repeated the Link-Incr and Link-Layout experiments with a mean holding time of 3 for three different distributions: a standard exponential, a two-stage hyperexponential with a standard deviation that was twice the mean, and a Pareto distribution. (We chose a mean of 3 because the Pareto distribution cannot have a mean of 1.) The offered loads were divided by 3 to keep the blocking probabilities roughly the same as before. The results are summarized in Table 3.4.

It is interesting to note that both the fastest migration in the Link-Incr experiment and the slowest migration (by far) in the Link-Layout experiment occurs for the Pareto distribution. This makes sense because the Pareto distribution typically has a greater number of both short calls and very long calls when compared to the other distributions.

We conclude that for small incremental changes, VP capacity migration is not much of a problem, but when dramatically changing VP layouts, it would be

	Migration Time	Offered Calls	Blocked Calls		
			Algorithm 3.1	Algorithm 3.2	Algorithm 3.3
Link-Incr	0.90 ± 0.05	631 ± 34	132 ± 8	103 ± 7	103 ± 7
Link-Layout	6.10 ± 0.14	1628 ± 38	1628 ± 38	198 ± 4	181 ± 4
Mesh-Incr	0.13 ± 0.02	23 ± 4	4.1 ± 0.7	2.6 ± 0.5	2.6 ± 0.5
Mesh-Layout	3.85 ± 0.29	723 ± 54	211 ± 14	68 ± 5	67 ± 4

Table 3.3: Simulation results for single link and mesh topologies with both incremental and layout changes during migration. Holding times are independent and exponentially distributed with mean 1.

	Migration Time	Offered Calls	Blocked Calls		
			Algorithm 3.1	Algorithm 3.2	Algorithm 3.3
Link-Incr					
Exponential	2.67 ± 0.17	619 ± 44	132 ± 8	103 ± 6	103 ± 6
Hyperexp	4.62 ± 0.69	1080 ± 159	219 ± 29	168 ± 22	168 ± 22
Pareto	2.03 ± 0.14	474 ± 29	81 ± 5	64 ± 4	64 ± 4
Link-Layout					
Exponential	18.01 ± 0.79	1604 ± 66	1604 ± 66	196 ± 5	179 ± 5
Hyperexp	70.18 ± 2.25	5954 ± 186	5954 ± 186	488 ± 11	448 ± 12
Pareto	670.48 ± 124.94	55976 ± 10420	55976 ± 10420	3707 ± 730	3682 ± 730

Table 3.4: Simulation results for our single link experiments with different holding time distributions. Holding times are i.i.d. with mean 3.

wise to implement a simple algorithm such as Algorithm 3.2. Algorithm 3.3 and other "smarter" decentralized algorithms do not exhibit enough performance gains over Algorithm 3.2 to warrant their additional complexity.

3.6 Chapter summary

Whereas Chapter 2 was concerned with flow aggregation in a static environment, the theme of this chapter has been adaptive resource allocation for aggregated flows in the face of time-varying demands and finite signaling resources. We argued that signaling resources may not be sufficient for future demands on ATM networks, and we presented a framework based on implied costs for adapting VP capacities to handle the increased processing and complexity costs.

In some cases it is desirable to modify the manner in which flows are aggregated (the VP layout). In this context we investigated algorithms to migrate from one layout to another, and we found that for incremental changes, the potential for performance losses during migration in terms of call blocking is minimal. However, when dramatic changes in the VP layout are warranted, it is desirable to enhance performance by implementing Algorithm 3.2, a simple decentralized algorithm that we have proposed.

Chapter 4

Network Aggregation: Hierarchical Source Routing Using Implied Costs

4.1 Introduction

In order to provide guaranteed QoS, communication systems are increasingly drawing on "connection-oriented" techniques. ATM networks are connection-oriented by design, allowing one to properly provision for QoS. Similarly, QoS extensions to the Internet, such as RSVP [11, 31, 77], make such networks akin to connection-oriented technologies. Indeed, the underlying idea is to reserve resources for packet flows, but to do it in a flexible manner using "soft state" which allows flows to be rerouted (or "connections" repacked [38]). Similar comments apply to an IP over ATM switching environment, where IP flows are mapped to ATM virtual circuits. In light of the above trends and the push toward global communication, our focus in this work is on how to make routing effective and manageable in a large-scale, connection-oriented network by using network aggregation. We shall first introduce hierarchical source routing, explain the basics of our routing algorithm, and give an example of the complexity reduction that it can achieve.

4.1.1 Hierarchical source routing: motivation and example

In a large-scale network, there are typically multiple paths connecting a given source/destination pair, and it is the job of the routing algorithm to split the demand among the available paths. The routing algorithm which we introduce in this

chapter fits nicely into the ATM Private Network-Network Interface (PNNI) framework [71], but it can also be thought of as a candidate for replacing the Border Gateway Protocol (BGP) [31] in the Internet that would split flows in "IP/RSVP" routing. Central to our algorithm is the *implied cost* [36] for a connection along a given path which measures the opportunity cost or expected loss of revenue resulting from accepting a connection. Using implied costs takes into account the possibility of "knock-on" effects (due to blocking and subsequent alternate routing) [36] and is geared towards achieving a *network optimal* routing algorithm.

To make good decisions and provide acceptable QoS, it is desirable to have a global view of the network at the source when making routing decisions for new connections. Thus, source routing, where the source specifies the entire path for the connection, is an attractive routing method. It has the additional advantage that, in contrast to hop-by-hop routing, there is no need to run a standardized routing algorithm to avoid loops and policy issues such as provider selection are easily accommodated. Propagating information for each link throughout the network quickly becomes unmanageable as the size of the network increases, so a hierarchical structure is needed, such as that proposed in the ATM PNNI specification [71]. Groups of switches are organized into *peer groups* (also referred to as *clouds*), and peer group leaders are chosen to coordinate the representation of each group's state. These collections of switches then form peer groups at the next level of the hierarchy and so on. Nodes keep detailed information for elements within their peer group. For other peer groups, they only have an approximate view for the current state, and this view can become coarser as the "distance" to remote areas of the network increases. We refer to the formation of peer groups as *network aggregation*. Besides reducing the amount of exchanged information, a hierarchical structure also makes addressing feasible in a large-scale network, as demonstrated by the network addressing of IP, and it permits the use of different routing schemes at different levels of the hierarchy. Prior work in the area of routing in networks with aggregated, and thus inaccurate information, can be found in [29, 48].

By combining a hierarchical network with (loose[1]) source routing, we have a form of routing referred to as *hierarchical source routing*. As an illustration, Fig. 4.1 shows a fragment of a larger network (Network 0) in which Peer Group

[1]In *loose* source routing, only the high-level path is specified by the source. The detailed path through a remote peer group is determined by a border switch of that peer group.

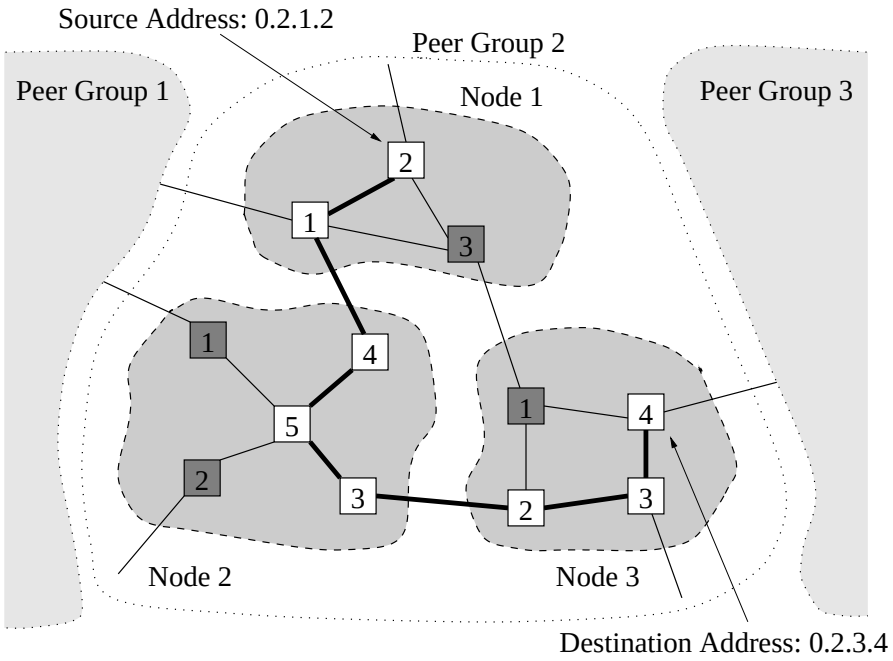


Figure 4.1: Illustration of hierarchical addressing and source routing.

2 contains Nodes 1, 2, and 3.[2] These nodes contain 3, 5, and 4 switches, respectively. To specify, for example, the source at Switch 2 of Node 1 of Peer Group 2 in Network 0, we use the 4-tuple 0.2.1.2. The example in Fig. 4.1 shows a source at 0.2.1.2 and destination at 0.2.3.4. The source 0.2.1.2 has specific information about its peer switches 0.2.1.1 and 0.2.1.3, but only aggregated information about nodes 0.2.2 and 0.2.3. The result of performing source routing is a tentative hierarchical path to reach the destination, e.g., $0.2.1.2 \rightarrow 0.2.1.1 \rightarrow 0.2.2 \rightarrow 0.2.3$ which specifies the exact path locally ($0.2.1.2 \rightarrow 0.2.1.1$) then the sequence of remote nodes to reach the destination ($\rightarrow 0.2.2 \rightarrow 0.2.3$). Upon initiating the connection request, the specified path is fleshed out, and, if successful, a (virtual circuit) connection satisfying prespecified end-to-end QoS requirements is set up. In this case, the border switches 0.2.2.4 and 0.2.3.2 in Nodes 2 and 3, respectively, are responsible for determining the detailed path to follow within their respective group. Furthermore, each switch will have a local Connection Admission Control (CAC) algorithm which it uses to determine whether new connection requests can in fact be

[2]These nodes are peer groups in their own right, but we use the term "node" here to avoid confusion with the peer groups at the next level of the hierarchy.

admitted without degraded performance. If the attempt fails, *crankback* occurs, and new attempts are made at routing the request. (Our model will ignore crankback.)

4.1.2 Explicit vs. implicit representations of available capacity

To do routing in this hierarchical framework, we must decide how to represent the "available" capacity of a peer group, either explicitly or implicitly. The explicit representation takes the physical topology and state of a peer group and represents it with a logical topology plus a metric denoting available capacity that is associated with each logical link. There may also be other metrics such as the average delay associated with logical links.

Typically, the first step in forming the explicit representation is to find the maximum available bandwidth path between each pair of border nodes, i.e., nodes directly connected to a link that goes outside the peer group. If we then create a logical link between each pair of border nodes and assign it this bandwidth parameter, we have taken the *full-mesh* approach [45]. If we collapse the entire peer group into a single point and advertise only one parameter value (usually the "worst case" parameter), we have taken the *symmetric-point* approach [45]. Most proposed solutions lie somewhere between these two extremes.

In the ATM PNNI specification [71], the baseline representation is a star in which each spoke has the same parameter value associated with it. More complex representations are permitted in which *exceptions* have a different associated parameter value than the default. These exceptions can be a spoke of the star or an additional logical link that connects a pair of border nodes.

Another alternative is to start with the full-mesh approach and encode the mesh in a maximum weight spanning tree [45]. External nodes can recover the full-mesh representation from the spanning tree if they desire. Whereas the symmetric star topology approximates the "capacity region" of the peer group by a hyper-cube region, the spanning tree approximates it with a hyper-rectangle. A simple example will help clarify the meaning of the term "capacity region." Suppose we have the three-link peer group shown in Fig. 4.2 with available link capacities C_1, C_2, and C_3 and routes r_1 and r_2. Let f_1 (f_2) be the current amount of capacity in use by connections on routes r_1 (r_2). Then we have three link constraints: $f_1 \leq C_1$, $f_2 \leq C_2$, and $f_1 + f_2 \leq C_3$, plus the requirement that $f_1 \geq 0$ and $f_2 \geq 0$. These constraints define the capacity region as shown in Fig. 4.3. The symmetric star topology approximates

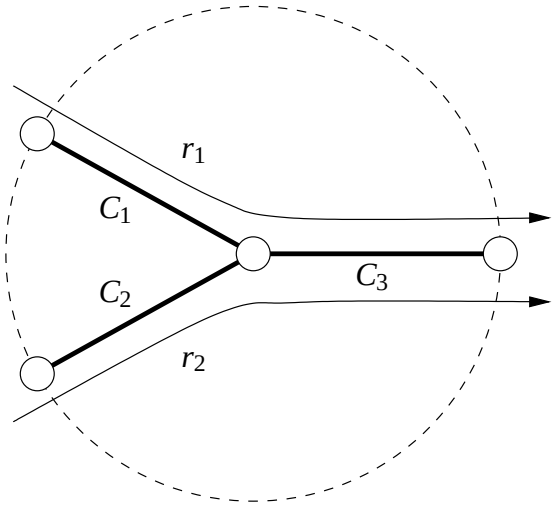


Figure 4.2: Peer group with three links and two routes.

the capacity region with a square defined by $f_1 \leq \min[C_1, C_2]$, $f_2 \leq \min[C_1, C_2]$, $f_1 \geq 0$, and $f_2 \geq 0$. The spanning tree approximates it with a rectangle given by $f_1 \leq C_1$, $f_2 \leq C_2$, $f_1 \geq 0$, and $f_2 \geq 0$. It should be clear from Fig. 4.4 that neither of these approaches captures the sharing of capacity by routes r_1 and r_2 on link 3, leading to a somewhat optimistic advertised capacity.

A third approach is to approximate the capacity region with a hyperplane [76]. For the example shown in Fig. 4.3, one possible choice would be the triangle given by $f_1 + f_2 \leq C_3$, $f_1 \geq 0$, and $f_2 \geq 0$ (see Fig. 4.4). When coupled with prediction of offered loads, the hyperplane approach has the potential to provide a more accurate picture of the available capacity than the star or the spanning tree.

None of the explicit representations, however, are without problems. For example, as mentioned earlier, the maximum available bandwidth paths between different pairs of border nodes may overlap, causing the advertised capacity to be too optimistic. Another questionable area is scalability to larger networks with more levels of hierarchy.

4.1.3 QoS routing based on implied costs

A more important problem is how the representation couples with routing. Can we really devise an accurate representation that is independent of the choice of routing

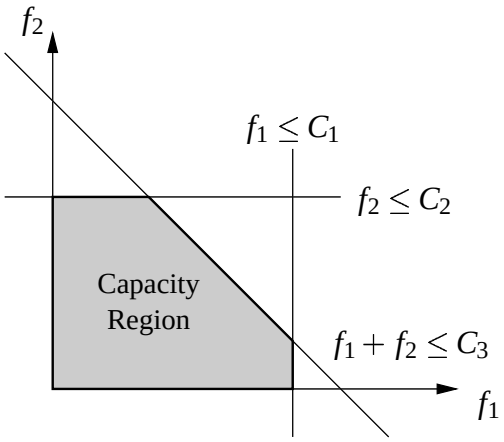
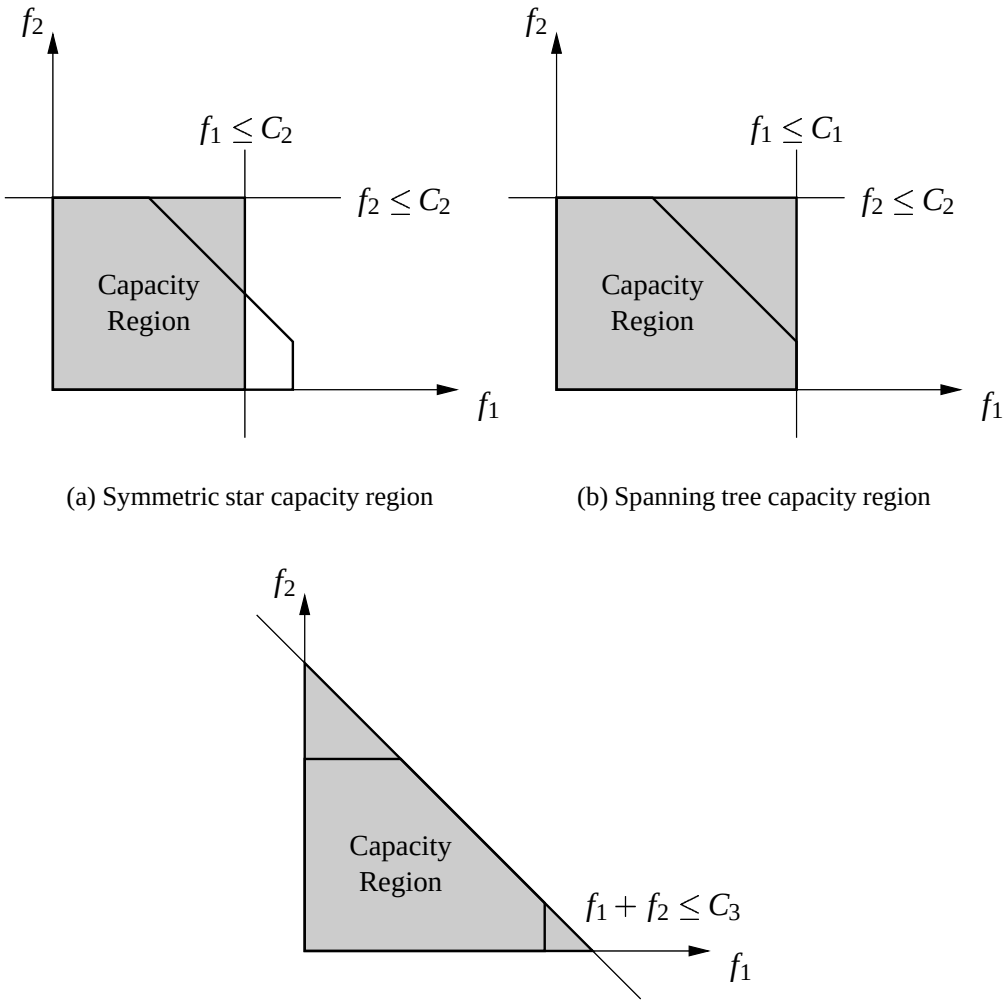


Figure 4.3: The capacity region based on the link constraints imposed on the flows along the two routes.

algorithm? None of the explicit representations address the effect that accepting a call would have on the congestion level both within the peer group and in other parts of the network due to interdependencies among traffic streams. For this reason, we introduce an implicit representation based on the average implied cost to go through or into a peer group that directly addresses this issue and is an integral part of the adaptive hierarchical source routing algorithm that we propose.

Such implied costs reflect the congestion in peer groups as well as the interdependencies among traffic streams in the network, and, independent of their use in a routing algorithm, they may be useful to network operators for the purpose of assessing current congestion levels. A rough motivation behind using the average is that, in a large network with diverse routing, a connection coming into a peer group can be thought of as taking a random path through that group, and hence the expected cost that a call would incur would simply be the average over all transit routes through that group. We will develop two closely related approximations: one in which the computed average implied costs are never used for the local portion of a route, and a more aggressive approximation in which the average implied cost is used locally as well as remotely for transit routes traversing more than one peer group. This second approximation will enable us to guarantee convergence of the implied cost computation under any traffic conditions, not just under light loads. With this approach, a route transiting through a peer group can be thought of as consuming an amount of bandwidth on each link in that peer group that is propor-



(a) Symmetric star capacity region

(b) Spanning tree capacity region

(c) Hyperplane capacity region

Figure 4.4: Various approximations to the capacity region in Fig. 4.3.

tional to the fraction of actual transit traffic in that peer group which passes through that link.

In order for our scheme to succeed, we need a hierarchical computation of the implied costs and a complementary routing algorithm to select among various hierarchical paths. The path selection will be done through adaptive (sometimes called quasi-static) routing, i.e., slowly varying how demand is split between transit routes that traverse more than one peer group, with the goal of maximizing the rate of revenue generated by the network. After eliminating routes which do not satisfy the QoS constraints, e.g., end-to-end delay,[3] the demand for transit routes connecting a given source/destination pair can be split based on the revenue sensitivities which are calculated using the implied costs. Within peer groups, we feel that dynamic routing should be used because of the availability of accurate local routing information.

By using an adaptive algorithm based on implied costs, we take the point of view that first it is of essence to design an algorithm that does the right thing on the "average," or say in terms of orienting the high-level flows in the system toward a desirable steady state. In order to make the routing scheme robust to fluctuations, appropriate actions would need to be taken upon blocking/crankback to ensure good, equitable performance in scenarios with temporary heavy loads.

4.1.4 Using hierarchy to reduce complexity

We now give an example of the complexity reduction achievable with our algorithm. Consider a network consisting solely of Peer Group 2 in Fig. 4.1. As will be explained in Sec. 4.4, the implied costs are computed via a distributed, iterative computation. At each iteration, the links must exchange their current values. Making the assumption that Nodes 1, 2, and 3 are connected locally using a broadcast medium, this would require 81 messages per iteration if we did not employ averaging. With our algorithm for computing the implied costs, only 41 messages per iteration would be needed, a savings of 49%. The memory savings would be commensurate with these numbers, and the computational complexity of the two algorithms is roughly the same. This reduction is significant because information update in an algorithm such as PNNI is a real problem, as it can easily overload the

[3] In our model, *effective bandwidth* [16, 40] allocation is used to control queueing delays which translates to a limit on hop counts plus propagation delay in order to satisfy a given delay bound.

network elements [62].

4.1.5 Chapter roadmap

The rest of this chapter is organized as follows. Sec. 4.2 summarizes the prior work directly relevant to the material in this chapter. Sec. 4.3 explains our model and notation. The theoretical basis of our adaptive routing scheme and its relation to Kelly's work is given in Sec. 4.4. An alternative approximation of the implied costs that works under any traffic conditions is developed in Sec. 4.5. Sec. 4.6 presents some computational results which attempt to quantify routing "errors" due to inaccuracies caused by aggregation. In Sec. 4.7, we discuss on-line measurements of some necessary parameters, and Sec. 4.8 briefly outlines extensions to a multiservice environment. Finally, Sec. 4.9 concludes with a chapter summary.

4.2 Related work

Hierarchical routing has been widely studied and used in both telephone and data networks [14, 26, 34, 43, 68]. Generally, only simple routing metrics such as hop count have been used to select appropriate paths. With the current trend toward integrated broadband networks, interest in QoS-sensitive routing algorithms has been increasing [47, 59, 74]. In addition, the desire for large-scale networking has made a combination of the above, hierarchical QoS-sensitive routing algorithms, an important area of study [29, 48, 56, 71]. For the specific case of routing in ATM networks, which supports QoS and makes use of hierarchy and is consequently quite complex, a good overview can be found in [2]. As an aside, we note that QoS routing problems such as the constrained shortest path problem are typically NP-complete [22, 74].

As part of the research on hierarchical QoS-sensitive routing, the explicit representation of available subnetwork capacity has been studied in detail [45, 46, 71, 76]. However, our implicit representation based on implied costs is new. Here we have extended the work of Kelly and others on the computation of implied costs and their use in adaptive routing schemes in single-service and multiservice flat networks [19, 36, 52]. Our proposed routing algorithm lies in the class of network optimal algorithms as it attempts to maximize the rate of revenue for the network instead of greedily trying to individually maximize each user's benefit. Network

versus user optimization and the possible effects on stability in QoS-sensitive routing is an issue worthy of further study. An earlier version of the material in this chapter can be found in [53].

4.3 Model and notation

Our model is that of a loss network serving a *single* type of traffic,[4] i.e., all calls require unit bandwidth, call holding times are independent (of all earlier arrival times and holding times) and identically distributed with unit mean, and blocked calls are lost. The unit bandwidth requirement per call can be considered to be an *effective bandwidth* [16, 40] which captures the traffic behavior. The capacity of each link $j \in \mathcal{J}$ is C_j units, and there are a total of J links in the network. Each link j is an element of a single node $n(j) \in \mathcal{N}$, where an aggregated node n is defined as a collection of links that form a peer group or that connect two peer groups.[5] We define E_{jn} to be an indicator function for the event that link j is an element of node n, and P_{jk} is an indicator function for the event that link j is a peer of link k (i.e., in the same node). A route is considered to be a collection of links in $\mathcal{J}$; route $r \in \mathcal{R}$ uses A_{jr} circuits on link $j \in \mathcal{J}$, where $A_{jr} \in \{0,1\}$.[6] A *transit route* is defined as a route that contains links in more than one node, and T_{nr} is an indicator function for the event that transit route r passes through node n. A call requesting route r is accepted if there are at least A_{jr} circuits available on every link j. If accepted, the call simultaneously holds A_{jr} circuits from link j for the holding time of the call. Otherwise, the call is blocked and lost. Calls requesting route r arrive as an independent Poisson process of rate ν_r. For convenience, definitions of the symbols we will be using are collected in Table 4.1. Where appropriate, all values referred to in this chapter are steady-state quantities.

For simplicity, we only consider a network with one level of aggregation like that shown in Fig. 4.5. This network has three peer groups, consisting of 3, 5, and 4 switches, respectively. The logical view of the network from a given peer group's perspective consists of complete information for all links within the peer group but only aggregated information for links between peer groups and in other peer groups.

[4] Extensions to multiservice networks will be presented in Sec. 4.8.

[5] There may be multiple links connecting the border switches of two peer groups. This set of one or more interconnecting links is considered to be a separate aggregated node in our model.

[6] In general, these routes might include multicast routes.

Symbol	Description
$J\ (\mathcal{J})$	Number (set) of links in the network.
C_j	Capacity of link j in circuits.
$R\ (\mathcal{R})$	Number (set) of routes defined in the network.
$N\ (\mathcal{N})$	Number (set) of nodes where a node is defined as a collection of links that form a peer group or that connect two peer groups.
$n(j)$	Link j is an element of node $n(j)$.
A_{jr}	Number of circuits (or units of capacity) used by route r on link j.
E_{jn}	Indicator function for the event that link j is an element of node n.
T_{nr}	Indicator function for the event that transit route r passes through node n.
P_{jk}	Indicator function for the event that link j is a peer of link k (i.e., in the same node).
ν_r	Rate of independent Poisson arrival process for route r.
L_r	Blocking probability for route r.
λ_r	Throughput achieved on route r.
B_j	Blocking probability at link j.
ρ_j	Reduced load at link j from thinned Poisson streams which pass through j.
θ_j	Throughput achieved through link j.
η_j	The expected increase in blocking probability at link j from removing a single circuit.
δ_j	The expected number of calls blocked at link j as a result of removing a single circuit for unit time.
w_r	Revenue generated by accepting a connection on route r.
$W(\nu; C)$	Rate of revenue for the network.
c_j	Implied cost to later calls which are blocked due to accepting a connection through link j.
c_r^n	Sum of implied costs for links in route r that lie in node n.
$\overline{c_n}$	Average implied cost of transiting through node n.
s_r	Surplus value (revenue minus costs) of an additional connection on route r.
$s_{r;\,j}$	Surplus value of an additional connection on route r from the perspective of link $j \in r$ (in the hierarchical framework).
$\mathcal{H}_n$	Set of hierarchical paths from the point of view of node n.
H_{jh}	Number of circuits used explicitly by hierarchical path h on link j.
L_h	Blocking probability for hierarchical path h.
λ_h	Throughput achieved on hierarchical path h.
$s_{h;\,j}$	Surplus value of an additional connection on hierarchical path h of which j is an explicit member.

Table 4.1: Definition of symbols for single-service model.

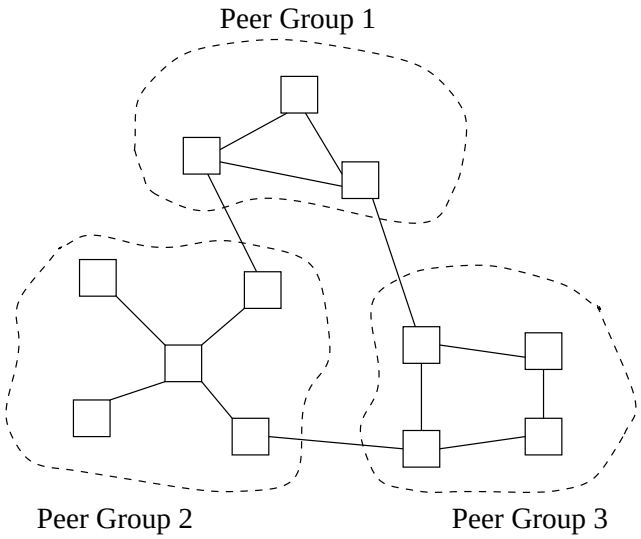


Figure 4.5: Example network with a single level of aggregation.

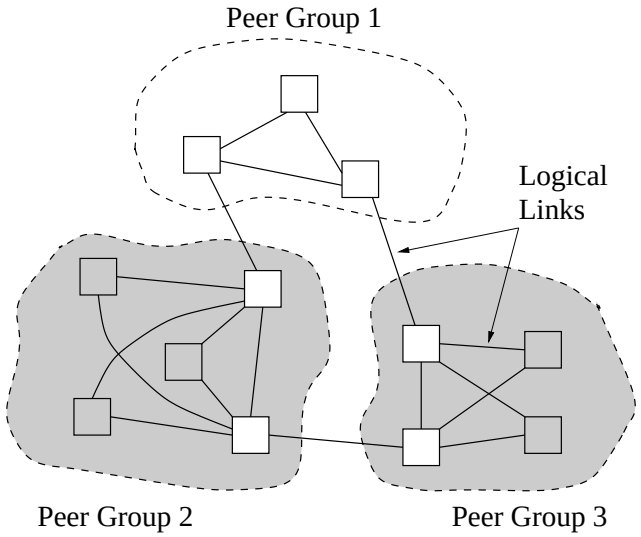


Figure 4.6: Logical view of the network from the perspective of peer group 1. The set of links connecting two peer groups is also considered to be an aggregated node in our model.

The other peer groups conceptually have logical links which connect each pair of border switches and connect each border switch to each internal destination. These logical links have an associated *implied cost*, i.e., marginal cost of using this logical resource, which is approximated from the real link implied costs. Currently, we calculate an average implied cost for any transit route that passes through or into a node, i.e., all of the logical links in a node have the same implied cost, and this value is then advertised to other peer groups. Fig. 4.6 shows the logical view of the example network from the perspective of peer group 1.

4.4 Approximations to revenue sensitivity

To calculate the revenue sensitivities, we must first find the blocking probability for each route, an important performance measure in its own right. Steady-state blocking probabilities can be obtained through the invariant distribution of the number of calls in progress on each route. However, the normalization constant for this distribution can be difficult to compute, especially for large networks. Therefore, the blocking probabilities are usually estimated using the Erlang fixed point approximation [26, 38]. For ease of reference, we repeat the presentation of the Erlang fixed point approximation already given in Sec. 3.4.1.

Let $B = (B_j, j \in \mathcal{J})$ be the solution to the equations

$$B_j = E(\rho_j, C_j), \quad j \in \mathcal{J}, \tag{4.1}$$

where

$$\rho_j = \sum_{r \in \mathcal{R}} A_{jr} \nu_r \prod_{k \in r - \{j\}} (1 - B_k) \tag{4.2}$$

and the function E is the Erlang B formula [8]

$$E(\rho_j, C_j) = \frac{\rho_j^{C_j}}{C_j!} \left[\sum_{n=0}^{C_j} \frac{\rho_j^n}{n!} \right]^{-1}. \tag{4.3}$$

The vector B is called the Erlang fixed point; its existence follows from the Brouwer fixed point theorem and uniqueness was proved in [35]. Using B, an approximation for the blocking probability on route r is

$$L_r \approx 1 - \prod_{k \in r} (1 - B_k). \tag{4.4}$$

The idea behind the approximation is as follows. Each Poisson stream of rate ν_r that passes through link j is thinned by a factor $1-B_k$ at each link $k \in r-\{j\}$ before being offered to j. Assuming these thinnings are independent both from link to link and over all routes, then the traffic offered to link j is Poisson with rate ρ_j as given in (4.2), the blocking probability at link j is B_j as given in (4.1), and the loss probability on route r is exactly L_r as given in (4.4).

Alternatively, instead of using the Erlang fixed point to approximate the blocking probabilities, it may be more accurate and efficient to measure the relevant quantities. Specifically, L_r, λ_r (the throughput achieved on route r), and $\theta_j = \sum_{r\in\mathcal{R}} A_{jr}\lambda_r$ (the total throughput through link j) can be obtained based on moving-average estimates. This will in turn allow us to compute the associated implied costs and hence the approximate revenue sensitivities. We will discuss the subject of on-line measurements more fully in Sec. 4.7.

Assuming that a call accepted on route r generates an expected revenue w_r, the rate of revenue for the network is

$$W(\nu; C) = \sum_{r\in\mathcal{R}} w_r\lambda_r. \tag{4.5}$$

Starting from the Erlang fixed point approximation and by extending the definition of the Erlang B formula (4.3) to non-integral values of C_j via linear interpolation,[7] the sensitivity of the rate of revenue with respect to the offered loads has been derived by Kelly [36] and is given by

$$\frac{\partial}{\partial\nu_r}W(\nu; C) = (1-L_r)s_r \tag{4.6}$$

where

$$s_r = w_r - \sum_{k\in\mathcal{J}} A_{kr}c_k \tag{4.7}$$

is the surplus value of an additional connection on route r, and the link implied costs are the (unique) solution to the equations

$$c_j = \eta_j(1-B_j)^{-1}\sum_{r\in\mathcal{R}} A_{jr}\lambda_r(s_r+c_j), \quad j\in\mathcal{J}, \tag{4.8}$$

[7] At integer values of C_j, define the derivative of $E(\rho_j, C_j)$ with respect to C_j to be the left derivative.

where $\eta_j = E(\rho_j, C_j - 1) - E(\rho_j, C_j)$. B_j, ρ_j, and L_r are obtained from the Erlang fixed point approximation, and $\lambda_r = \nu_r(1 - L_r)$.

Remark. In a flat network, the offered load for a given source/destination pair should be split among the available routes based on the revenue sensitivities in (4.6). An additional call offered to route r will be accepted with probability $1 - L_r$. If accepted, it will generate revenue w_r, but at a cost of c_j for each $j \in r$. The implied costs c quantify the potential knock-on effects or expected loss in revenue due to accepting a call. The goal of the routing algorithm is to maximize the rate of network revenue $W(\nu; C)$ by adaptively adjusting the splitting for each source/destination pair over time in response to changing traffic conditions. The splitting for a source/destination pair should favor routes for which $(1 - L_r)s_r$ has a positive value since increasing the offered traffic on these routes will increase the rate of revenue. Routes for which $(1 - L_r)s_r$ is negative should be avoided, with all adjustments of the splitting made gradually to guard against sudden congestion. We note that, in general, $W(\nu; C)$ is not concave, so there may exist nonoptimal local maxima. However, Kelly has shown that it is asymptotically linear as ν and C are increased in proportion [36]. Furthermore, even though the routing algorithm could potentially reach a nonoptimal local maximum of the revenue function, the stochastic fluctuations in the offered traffic may allow it to escape that particular region.

To perform aggregation by peer group, we first define the quantity $\bar{c}_n$ as the weighted average of the implied costs associated with pieces of transit routes that pass through or enter node n (or, equivalently, over the links in n visited by such routes) where, in the following, $c_r^n = \sum_{j \in \mathcal{J}} A_{jr} E_{jn} c_j$:

$$\begin{aligned} \bar{c}_n &= \frac{\sum_{r \in \mathcal{R}} T_{nr} \lambda_r c_r^n}{\sum_{r \in \mathcal{R}} T_{nr} \lambda_r} \\ &= \frac{\sum_{j \in \mathcal{J}} E_{jn} (\sum_{r \in \mathcal{R}} T_{nr} A_{jr} \lambda_r) c_j}{\sum_{r \in \mathcal{R}} T_{nr} \lambda_r}. \end{aligned} \tag{4.9}$$

This averaging is illustrated in Fig. 4.7. We redefine the surplus value for a route as a function of the local link implied costs and the remote nodal implied costs, *from the perspective of link* $j \in r$ (see Fig. 4.8):

$$s_{r;\,j} = w_r - \sum_{k \in \mathcal{J}} A_{kr} P_{kj} c_k - \sum_{n \neq n(j)} T_{nr}\, \bar{c}_n. \tag{4.10}$$

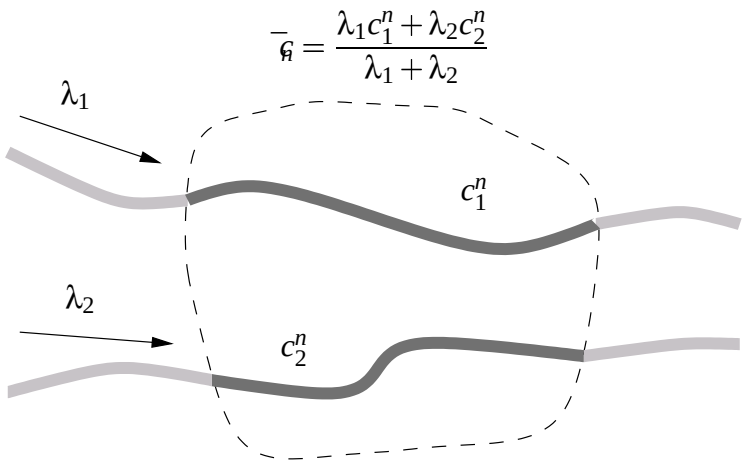


Figure 4.7: Computation of $\bar{c}_n$ for an aggregated node n with two transit routes.

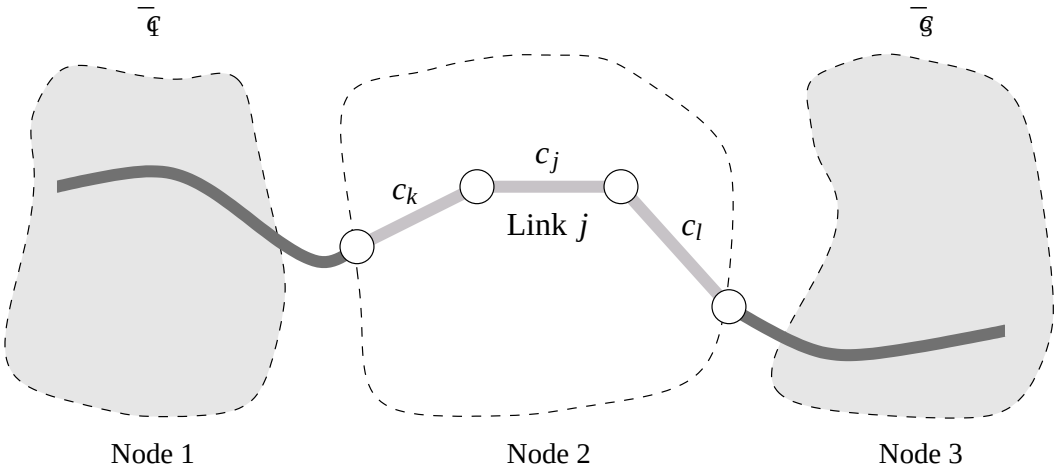


Figure 4.8: Implied costs for a route from the perspective of link j.

The link implied costs are now calculated as

$$c_j = \eta_j(1-B_j)^{-1}\sum_{r\in\mathcal{R}} A_{jr}\lambda_r(s_{r;\,j}+c_j), \quad j\in\mathcal{J}. \tag{4.11}$$

In the sequel, we will address the following issues: the existence of a unique solution to these equations, convergence to that solution, and the accuracy relative to Kelly's implied costs.

Eq. (4.11) can be solved iteratively in a distributed fashion via successive substitution. If we define a linear mapping $f:\mathbb{R}^J\to\mathbb{R}^J$ by $f=(f_1,f_2,\ldots,f_J)$,

$$f_j(x) = \eta_j(1-B_j)^{-1}\sum_{r\in\mathcal{R}} A_{jr}\lambda_r(w_r - \sum_{k\neq j} A_{kr}P_{kj}x_k - \sum_{n\neq n(j)} T_{nr}\,\overline{x}_n), \tag{4.12}$$

then successive substitution corresponds to calculating the sequence $f^i(x), i=1,2,\ldots$, where $f^i(x)$ is the result of iterating the linear mapping i times.

Define a norm on $\mathbb{R}^J$ by

$$\|x\|_M = \max_{j,r}\{A_{jr}(\sum_{k\neq j} A_{kr}P_{kj}|x_k| + \sum_{n\neq n(j)} T_{nr}\,\overline{|x|}_n)\} \tag{4.13}$$

where

$$\overline{|x|}_n = \frac{\sum_{j\in\mathcal{J}} E_{jn}(\sum_{r\in\mathcal{R}} T_{nr}A_{jr}\lambda_r)|x_j|}{\sum_{r\in\mathcal{R}} T_{nr}\lambda_r}.$$

For any positive vector α, we define the weighted maximum norm on $\mathbb{R}^J$ by $\|x\|_\infty^\alpha = \max_j|\frac{x_j}{\alpha_j}|$, where we suppress the index α if $\alpha_j=1$ for all j. Also, let $\delta=(\delta_1,\delta_2,\ldots,\delta_J)$, where $\delta_j=\eta_j\rho_j$ denotes Erlang's improvement formula.

Theorem 4.1. *Suppose that $\|\delta\|_M<1$. Then the mapping $f:\mathbb{R}^J\to\mathbb{R}^J$ is a contraction mapping under the norm $\|\cdot\|_M$, and the sequence $f^i(x), i=1,2,\ldots$, converges to c', the unique solution of* (4.11)*, for any $x\in\mathbb{R}^J$.*

Proof: Choose $x,x'\in\mathbb{R}^J$. Then, $\forall\, j\in\mathcal{J}$,

$$f_j(x)-f_j(x') = -\eta_j(1-B_j)^{-1}\sum_{r\in\mathcal{R}} A_{jr}\lambda_r\big(\sum_{k\neq j} A_{kr}P_{kj}(x_k-x'_k) + \sum_{n\neq n(j)} T_{nr}(\overline{x}_n-\overline{x'}_n)\big).$$

Therefore

$$
\begin{aligned}
|f_j(x) - f_j(x')| &\le \eta_j (1 - B_j)^{-1} \sum_{r \in \mathcal{R}} A_{jr} \lambda_r \Big(\sum_{k \neq j} A_{kr} P_{kj} |x_k - x'_k| + \sum_{n \neq n(j)} T_{nr} |\bar{x}_n - \bar{x}'_n| \Big) \\
&\le \eta_j (1 - B_j)^{-1} \sum_{r \in \mathcal{R}} A_{jr} \lambda_r \|x - x'\|_M \\
&= \eta_j \rho_j \|x - x'\|_M .
\end{aligned}
$$

Taking the norm on both sides, we have

$$
\|f(x) - f(x')\|_M \le \|\delta\|_M \|x - x'\|_M .
$$

So $f(\cdot)$ is a contraction mapping if $\|\delta\|_M < 1$. Using the definition of a contraction mapping and the properties of norms, one can easily show that the sequence $f^i(x), i = 1, 2, \ldots$, converges to c', the unique solution of (4.11), for any $x \in \mathbb{R}^J$.

■

Remark. The product $\eta_j \rho_j$ increases to 1 as ρ_j, the offered load at link j, increases [36]. So $\|\delta\|_M < 1$ can be referred to as a light load condition. If the network has long routes and/or heavily loaded links, this constraint may be violated, but at moderate utilization levels, we expect that it will hold. As an example, consider a loss network in which all links have capacity $C = 150$ and the reduced load at each link from thinned Poisson streams is $\rho = 100$. Furthermore, for simplicity, assume that each transit route across a node has the same length. Then $\delta = 3.3 \times 10^{-5}$ for each link, and the condition $\|\delta\|_M < 1$ requires the maximum route length to be at most 30,717 links. The blocking probability for a route of maximum length is approximately 2% (under the link independence assumption). If ρ is increased to 120 for each link, the maximum route length is 33 links with a blocking probability of approximately 3% along such a route. At $\rho = 140$, the maximum route length is 3 links with a blocking probability of approximately 8%. For this example, link utilizations up to about 80% are certainly feasible under our "light load" condition. As the capacities of the links increase (relative to bandwidth requests), even higher utilizations are possible before the maximum route length becomes too small and/or blocking becomes prohibitive.

The convergence proved in Thm. 4.1 assumes iterates are computed synchronously. In a large-scale network, synchronous computation may be infeasible,

so we will show that our light load condition is sufficient for convergence of an asynchronous computation in the following sense [9]:

Assumption 4.1. (Total Asynchronism) Each link performs updates infinitely often, and given any time t_1, there exists a time $t_2 > t_1$ such that for all $t \geq t_2$, no component values (link and average implied costs) used in updates occurring at time t were computed before t_1.

Note that, under this assumption, old information is eventually purged from the computation, but the amount of time by which the variables are outdated can become unbounded as t increases.

Theorem 4.2. *Suppose that* $\|\delta\|_M < 1$ *and* $\delta > 0$. *Then, under Assumption 4.1 (total asynchronism), the sequence* $f^i(x), i = 1,2,\ldots$, *converges to* c', *the unique solution of* (4.11), *for any* $x \in \mathbb{R}^J$.

Proof: Rewrite (4.11) in matrix form as $f(x) = Gx + b$. The goal is to show that G corresponds to a weighted maximum norm contraction. For, in that case, we can satisfy the conditions of the Asynchronous Convergence Theorem in [9] (see Sec. 6.2 and 6.3, pp. 431–435), which guarantees asynchronous convergence to the unique fixed point c'. In the following, we use δ as the weight vector for the weighted maximum norm; in order to do so, we require the condition $\delta \geq 0$. (We are guaranteed that $\delta \geq 0$, but in all practical cases $\delta > 0$ as we have assumed).

Choose $x, x' \in \mathbb{R}^J$. Then, $\forall j \in \mathcal{J}$,

$$|f_j(x) - f_j(x')| \leq \eta_j (1 - B_j)^{-1} \sum_{r \in \mathcal{R}} A_{jr} \lambda_r \Big(\sum_{k \neq j} A_{kr} P_{kj} |x_k - x'_k| + \sum_{n \neq n(j)} T_{nr} |\bar{x}_n - \bar{x}'_n| \Big).$$

Therefore

$$\begin{aligned}
\left|\frac{f_j(x)-f_j(x')}{\delta_j}\right| &\le \frac{\eta_j(1-B_j)^{-1}}{\delta_j}\sum_{r\in\mathcal{R}} A_{jr}\lambda_r\left(\sum_{k\neq j} A_{kr}P_{kj}\delta_k\left|\frac{x_k-x'_k}{\delta_k}\right|\right.\\
&\qquad\left.+\sum_{n\neq n(j)} T_{nr}\,\frac{\sum_{l\in\mathcal{J}} E_{ln}(\sum_{q\in\mathcal{R}} T_{nq}A_{lq}\lambda_q)\delta_l\left|\frac{x_l-x'_l}{\delta_l}\right|}{\sum_{q\in\mathcal{R}} T_{nq}\lambda_q}\right)\\
&\le \frac{\eta_j(1-B_j)^{-1}}{\delta_j}\sum_{r\in\mathcal{R}} A_{jr}\lambda_r\left(\sum_{k\neq j} A_{kr}P_{kj}\delta_k\right.\\
&\qquad\left.+\sum_{n\neq n(j)} T_{nr}\,\frac{\sum_{l\in\mathcal{J}} E_{ln}(\sum_{q\in\mathcal{R}} T_{nq}A_{lq}\lambda_q)\delta_l}{\sum_{q\in\mathcal{R}} T_{nq}\lambda_q}\right)\|x-x'\|^{\delta}_{\infty}
\end{aligned}$$

since the weighted maximum norm $\|x\|^{\delta}_{\infty} = \max_{j\in\mathcal{J}}|\frac{x_j}{\delta_j}|$. Taking the norm on both sides, we have

$$\|f(x)-f(x')\|^{\delta}_{\infty} \le \|G\|^{\delta}_{\infty}\|x-x'\|^{\delta}_{\infty}$$

where the induced matrix norm $\|G\|^{\delta}_{\infty} = \max_{j\in\mathcal{J}}\{\frac{1}{\delta_j}\sum_{k\in\mathcal{J}}|g_{jk}|\delta_k\}$ [9]. So G corresponds to a weighted maximum norm contraction if $\|G\|^{\delta}_{\infty} < 1$. This follows from $\|\delta\|_M < 1$ because

$$\begin{aligned}
\|G\|^{\delta}_{\infty} &= \max_{j\in\mathcal{J}}\frac{\eta_j(1-B_j)^{-1}}{\delta_j}\sum_{r\in\mathcal{R}} A_{jr}\lambda_r\left(\sum_{k\neq j} A_{kr}P_{kj}\delta_k\right.\\
&\qquad\left.+\sum_{n\neq n(j)} T_{nr}\,\frac{\sum_{l\in\mathcal{J}} E_{ln}(\sum_{q\in\mathcal{R}} T_{nq}A_{lq}\lambda_q)\delta_l}{\sum_{q\in\mathcal{R}} T_{nq}\lambda_q}\right)\\
&\le \max_{j\in\mathcal{J}}\frac{\eta_j(1-B_j)^{-1}}{\delta_j}\sum_{r\in\mathcal{R}} A_{jr}\lambda_r\|\delta\|_M\\
&= \|\delta\|_M
\end{aligned}$$

since $\rho_j = (1-B_j)^{-1}\sum_{r\in\mathcal{R}} A_{jr}\lambda_r$ and $\delta_j = \eta_j\rho_j$. ■

Remark. With the additional restriction of bounded communication delays, the convergence rate of an asynchronous iteration satisfying the conditions of Thm. 4.2 is geometric and can actually be faster than the corresponding synchronous version which has to wait for all values from the previous iteration to be distributed before performing the next update. See [9, pp. 441–443] for the details of a situation analogous to ours which has "fast" local communication (within peer groups)

and "slower" remote communication (between peer groups) and where the asynchronous convergence rate is faster if there is a "strong coupling" among the local variables (i.e., the local implied costs), a condition which should typically hold true in a hierarchical network if the amount of local traffic dominates the amount of remote traffic in each peer group.

Theorem 4.3. *Suppose that* $\|\delta\|_M < 1$ *and denote* c *and* c' *as the solutions to* (4.8) *and* (4.11)*, respectively. Define* $\Delta = \max_{n,r}\{T_{nr}\sum_{m\neq n} T_{mr}|c_r^m - \bar{c}_n^m|\}$ *where* $c_r^m = \sum_{j\in\mathcal{J}} A_{jr}E_{jm}c_j$ *and* $\bar{c}_n^m$ *is defined by* (4.9)*. Then we have*

$$\|s - s'\|_\infty \leq \frac{\Delta\|\delta + 1\|_\infty}{1 - \|\delta\|_M} \tag{4.14}$$

where by $\|s - s'\|_\infty$ *we mean* $\max_{j,r:j\in r}|s_r - s'_{r;\,j}|$.

Proof: We have, $\forall\, j \in \mathcal{J}$,

$$c'_j - c_j = \eta_j(1 - B_j)^{-1}\sum_{r\in\mathcal{R}} A_{jr}\lambda_r\Big(\sum_{k\neq j} A_{kr}P_{kj}(c_k - c'_k) + \sum_{n\neq n(j)} T_{nr}(c_r^n - \bar{c}_n^r)\Big).$$

Hence

$$\begin{aligned}|c'_j - c_j| &\leq \eta_j(1 - B_j)^{-1}\sum_{r\in\mathcal{R}} A_{jr}\lambda_r\Big(\sum_{k\neq j} A_{kr}P_{kj}|c_k - c'_k| \\ &\qquad + \sum_{n\neq n(j)} T_{nr}|c_r^n - \bar{c}_n^r + \bar{c}_n^r - \bar{c}_n^r|\Big) \\ &\leq \eta_j\rho_j(\|c' - c\|_M + \Delta).\end{aligned} \tag{4.15}$$

Taking the M-norm on both sides and rearranging, we have

$$\|c' - c\|_M \leq \frac{\Delta\|\delta\|_M}{1 - \|\delta\|_M}. \tag{4.16}$$

We also have, $\forall\, j, r$ such that $j \in r$,

$$s_r - s'_{r;\,j} = \sum_{k\in\mathcal{J}} A_{kr}P_{kj}(c'_k - c_k) + \sum_{n\neq n(j)} T_{nr}(\bar{c}_n^r - c_r^n).$$

Hence

$$
\begin{aligned}
|s_r - s'_{r;\,j}| &\le \sum_{k \in \mathcal{J}} A_{kr} P_{kj} |c'_k - c_k| + \sum_{n \neq n(j)} T_{nr} |\overline{c}_n - \overline{c}_n + \overline{c}_n - c^n_r| \\
&\le |c'_j - c_j| + \|c' - c\|_M + \Delta && \text{since } A_{jr} = 1 \\
&\le \eta_j \rho_j (\|c' - c\|_M + \Delta) + \|c' - c\|_M + \Delta && \text{using (4.15)} \\
&= (\delta_j + 1)(\|c' - c\|_M + \Delta) \\
&\le (\delta_j + 1) \frac{\Delta}{1 - \|\delta\|_M} && \text{using (4.16).}
\end{aligned}
$$

Taking the maximum norm on both sides, the result follows. ■

Remark. The error between our modified implied costs and Kelly's implied costs will be minimized under light loads ($\|\delta\|_M \ll 1$) and if the difference between transit route costs and the average for each node is small (Δ close to 0). We use the maximum norm of $s - s'$ as a comparison because it directly affects the difference in the revenue sensitivity in (4.6) using the flat and hierarchical frameworks. The measured value of L_r used in (4.6) may also be different from that in a flat network because it is potentially averaged over several routes with the same hierarchical path from a given node's point of view. When making adaptive routing decisions, we are really only concerned with the relative values of $\frac{\partial}{\partial \nu_r} W(\nu; C)$ among routes sharing a common source/destination pair. It is unclear in what situations our approximation might affect this ordering.

4.5 An alternative approximation

In this section, we consider a more aggressive averaging mechanism. In the previous approach, we used exact information for resources within a peer group and aggregated metrics to represent its remote peers. By contrast, herein we also perform local averaging among routes transiting through or into a local peer group. We will show that this alternative approximation has a similar structure to the previous case, although the relation to the exact implied costs is further "removed." The key advantage of this approach is that, subject to sufficient damping, one can show convergence to new approximate implied costs under any traffic conditions and route topology. In fact, the required damping within a peer group depends only on local

information, the number of links within the peer group, and aggregated global information, the total number of peer groups. Thus, the damping factor within a peer group only requires information that is consistent with its hierarchically aggregated view of the network, and the nonlocal knowledge required, namely the total number of peer groups, is not detrimental to the decentralized nature of the computation.

Define the matrix $\bar{A}$ with elements $\bar{A}_{jr} \in [0,1]$ such that

$$\bar{A}_{jr} = \begin{cases} \frac{\sum_{q \in \mathcal{R}} T_{n(j)q} A_{jq} \lambda_q}{\sum_{q \in \mathcal{R}} T_{n(j)q} \lambda_q} & \text{if } T_{n(j)r} = 1, \\ A_{jr} & \text{if } T_{n(j)r} = 0. \end{cases} \tag{4.17}$$

Local routes remain unchanged: they take a single circuit on each link that they traverse. However, transit routes can be thought of as consuming a fraction of a circuit on every link in each node that they traverse. This fraction is equal to the fraction $\bar{A}_{jr}$ of transit traffic in node $n(j)$ which passes through that link. Note that the offered load ρ_j at link j remains the same whether it is computed based on the flat network's routing matrix A or the aggregated routing matrix $\bar{A}$. Indeed, for fixed λ_r, we have $\rho_j = (1-B_j)^{-1} \sum_{r \in \mathcal{R}} A_{jr} \lambda_r = (1-B_j)^{-1} \sum_{r \in \mathcal{R}} \bar{A}_{jr} \lambda_r$.

By substituting $\bar{A}$ for A in (4.8), we have the following implied cost equations:

$$c_j = \eta_j (1-B_j)^{-1} \sum_{r \in \mathcal{R}} \bar{A}_{jr} \lambda_r (w_r - \sum_{k \neq j} \bar{A}_{kr} c_k), \quad j \in \mathcal{J}. \tag{4.18}$$

We can rewrite these equations in various ways to bring out the connections with both our first aggregation method (4.9) and the original implied cost equations (4.8) for a flat network. First, we note that for a given link j and route r such that $T_{n(j)r} = 1$, we have $\sum_{k \in \mathcal{J}} \bar{A}_{kr} P_{kj} c_k = \bar{c}_{n(j)}$, which illuminates the role of $\bar{A}_{jr}$ in performing additional averaging of implied costs at the local level; compare this with (4.9). Second, we can rewrite (4.18) as

$$c_j = \eta_j (1-B_j)^{-1} \sum_{r \in \mathcal{R}} \bar{A}_{jr} \lambda_r (w_r - \sum_{k \neq j} \bar{A}_{kr} P_{kj} c_k - \sum_{n \neq n(j)} T_{nr} \bar{c}_n) \tag{4.19}$$

$$= \eta_j (1-B_j)^{-1} \sum_{r \in \mathcal{R}} \big[(1 - T_{n(j)r}) A_{jr} \lambda_r (w_r - \sum_{k \in \mathcal{J}} A_{kr} c_k + c_j) + T_{n(j)r} \bar{A}_{jr} \lambda_r (w_r - \sum_{n \in \mathcal{N}} T_{nr} \bar{c}_n + \bar{A}_{jr} c_j) \big]. \tag{4.20}$$

Eq. (4.19) indicates the connection with our previous equations (4.11) for a hierarchical network, the only difference being the use of the $\bar{A}$ matrix locally. In (4.20), we see that the equation for c_j is a combination of the original equation (4.8) for routes not transiting through node $n(j)$ and an equation based on "averaged" surplus values $s_r = w_r - \sum_{n\in\mathcal{N}} T_{nr}\bar{c}_n$ for routes transiting through node $n(j)$ plus the use of $\bar{A}_{jr}$ instead of A_{jr}.

Based on the above, we define a new linear mapping $\tilde{f}:\mathbb{R}^J \to \mathbb{R}^J$ by $\tilde{f} = (\tilde{f}_1, \tilde{f}_2, \ldots, \tilde{f}_J)$,

$$\tilde{f}_j(x) = \eta_j(1-B_j)^{-1}\sum_{r\in\mathcal{R}} \bar{A}_{jr}\lambda_r(w_r - \sum_{k\neq j}\bar{A}_{kr}x_k), \tag{4.21}$$

where $\tilde{f}^i(x)$ is the result of iterating the linear mapping i times. Define $\tilde{f}_{(\gamma)}:\mathbb{R}^J \to \mathbb{R}^J$ to be a damped version of the iteration $\tilde{f}(\cdot)$ for $\gamma = \mathrm{diag}(\gamma_j)_j$ where $\gamma_j \in (0,1)\ \forall\ j \in \mathcal{J}$:

$$\tilde{f}_{(\gamma)}(x) = (I-\gamma)x + \gamma\tilde{f}(x). \tag{4.22}$$

If we define a norm on $\mathbb{R}^J$ by

$$\|x\|_{\tilde{M}} = \max_{j,r}\{1(\bar{A}_{jr}>0)\sum_{k\neq j}\bar{A}_{kr}|x_k|\}, \tag{4.23}$$

then Thms. 4.1 and 4.2 can be shown to hold for $\tilde{f}(x)$ under the condition $\|\delta\|_{\tilde{M}} < 1$. However, our main interest here lies in proving convergence of the damped iteration $\tilde{f}_{(\gamma)}(x)$ without requiring $\|\delta\|_{\tilde{M}}$ to be less than one.

The proofs of the following two theorems closely resemble the development in [36, Sec. 4]. Note that all vectors are considered to be column vectors.

Theorem 4.4. *The equations* (4.18) *have a unique solution* $\tilde{c}$.

Proof: Rewrite (4.18) in the equivalent form

$$c_j = \eta_j(1-\tilde{\delta}_j)^{-1}(1-B_j)^{-1}\sum_{r\in\mathcal{R}}\bar{A}_{jr}\lambda_r(w_r - \sum_{k\in\mathcal{J}}\bar{A}_{kr}c_k), \tag{4.24}$$

where $\tilde{\delta}_j = \eta_j(1-B_j)^{-1}\sum_{r\in\mathcal{R}}\bar{A}_{jr}^2\lambda_r \leq \eta_j\rho_j$. Let $\tilde{g} = (\tilde{g}_1, \tilde{g}_2, \ldots, \tilde{g}_J)$ where $\tilde{g}_j(c)$ denotes the right-hand side of (4.24). Also, define the positive diagonal matrices

$$\lambda = \mathrm{diag}(\lambda_r)_r, \quad \zeta = \mathrm{diag}(\eta_j(1-\tilde{\delta}_j)^{-1}(1-B_j)^{-1})_j.$$

Then we can write (4.24) in matrix form as

$$c = \tilde{g}(c) = \zeta\bar{A}\lambda(w - \bar{A}^T c).$$

Define the positive diagonal matrices $\zeta^{\frac{1}{2}}$, $\zeta^{-\frac{1}{2}}$ componentwise. The equation $c = \tilde{g}(c)$ is equivalent to $(I + \zeta\bar{A}\lambda\bar{A}^T)c = \zeta\bar{A}\lambda w$. Multiplying both sides of this equation on the left by $\zeta^{-\frac{1}{2}}$, we have

$$(I + \zeta^{\frac{1}{2}}\bar{A}\lambda\bar{A}^T\zeta^{\frac{1}{2}})\zeta^{-\frac{1}{2}}c = \zeta^{\frac{1}{2}}\bar{A}\lambda w.$$

The symmetric matrix $(I + \zeta^{\frac{1}{2}}\bar{A}\lambda\bar{A}^T\zeta^{\frac{1}{2}})$ is positive definite and hence invertible. Thus the equation $c = \tilde{g}(c)$ has a unique solution

$$\tilde{c} = \zeta^{\frac{1}{2}}(I + \zeta^{\frac{1}{2}}\bar{A}\lambda\bar{A}^T\zeta^{\frac{1}{2}})^{-1}\zeta^{\frac{1}{2}}\bar{A}\lambda w, \tag{4.25}$$

which is also the unique solution to (4.18). ■

In the following, let J_n denote the number of links in node n, and recall that N denotes the total number of aggregated nodes in the network.

Theorem 4.5. *If $\gamma_j \leq (NJ_{n(j)})^{-1}\ \forall\ j \in \mathcal{J}$, then the sequence $\tilde{f}^i_{(\gamma)}(x), i = 1,2,\ldots,$ converges to $\tilde{c}$, the unique solution of* (4.18)*, for any $x \in \mathbb{R}^J$.*

Proof: It is enough to establish that the sequence $\tilde{f}^i_{(\gamma)}(x), i = 1,2,\ldots,$ converges since the limit vector must solve (4.18) by the continuity of $\tilde{f}_{(\gamma)}(\cdot)$. Define the diagonal matrices

$$\lambda = \mathrm{diag}(\lambda_r)_r, \quad \beta = \mathrm{diag}(1 - B_j)_j, \quad \eta = \mathrm{diag}(\eta_j)_j, \quad \tilde{\delta} = \mathrm{diag}(\eta_j\tilde{\rho}_j)_j,$$

where $\tilde{\rho}_j = (1 - B_j)^{-1}\sum_{r\in\mathcal{R}}\bar{A}^2_{jr}\lambda_r$. Then in matrix form

$$\tilde{f}(x) = \eta\beta^{-1}\bar{A}\lambda w - \eta\beta^{-1}\bar{A}\lambda\bar{A}^T x + \tilde{\delta}x,$$

and so

$$\tilde{f}_{(\gamma)}(x) = [I - \gamma(I - \tilde{\delta}) - \gamma\eta\beta^{-1}\bar{A}\lambda\bar{A}^T]x + \gamma\eta\beta^{-1}\bar{A}\lambda w.$$

The sequence $\tilde{f}^i_{(\gamma)}(x), i = 1,2,\ldots$, will converge provided the eigenvalues of the iteration matrix $[I-\gamma(I-\tilde{\delta})-\gamma\eta\beta^{-1}\bar{A}\lambda\bar{A}^T]$ lie in the interval $(-1,1)$. The eigenvalues of this matrix coincide with the eigenvalues of the matrix

$$\begin{aligned}\gamma^{-\frac{1}{2}}\eta^{-\frac{1}{2}}\beta^{\frac{1}{2}}[I-\gamma(I-\tilde{\delta})-\gamma\eta\beta^{-1}\bar{A}\lambda\bar{A}^T]\beta^{-\frac{1}{2}}\eta^{\frac{1}{2}}\gamma^{\frac{1}{2}} \\ &= I-\gamma(I-\tilde{\delta})-\gamma^{\frac{1}{2}}\eta^{\frac{1}{2}}\beta^{-\frac{1}{2}}\bar{A}\lambda\bar{A}^T\beta^{-\frac{1}{2}}\eta^{\frac{1}{2}}\gamma^{\frac{1}{2}} \\ &= I-\gamma(I-\tilde{\delta})-(\lambda^{\frac{1}{2}}\bar{A}^T\beta^{-\frac{1}{2}}\eta^{\frac{1}{2}}\gamma^{\frac{1}{2}})^T(\lambda^{\frac{1}{2}}\bar{A}^T\beta^{-\frac{1}{2}}\eta^{\frac{1}{2}}\gamma^{\frac{1}{2}})\end{aligned}$$

which is of the form $I-(D+M)$ where D is a diagonal matrix and M is a symmetric, positive semi-definite matrix of the form $M = Y^TY$. The eigenvalues of D are equal to its diagonal terms $d_j, j \in \mathcal{J}$. Let $\rho(D)$ denote the spectral radius of D, i.e., the maximum of the magnitudes of its eigenvalues. Since $0 \leq \tilde{\delta}_j \leq \eta_j\rho_j < 1$ and $\gamma_j \in (0,1)$ for all $j \in \mathcal{J}$, we have that $0 < d_j < 1, j \in \mathcal{J}$, and thus $\rho(D) < 1$.

Next, we determine a bound on the spectral radius of M. Let $\|\cdot\|_2$ denote the Euclidean norm, and define the induced matrix norm $\|M\|_2$ as $\max_{\|x\|_2=1}\|Mx\|_2$. Since M is symmetric, it can be shown that $\rho(M) = \|M\|_2 = \max_{\|x\|_2=1}|x^TMx| = \max_{\|x\|_2=1}|x^TY^TYx| = \max_{\|x\|_2=1}\|Yx\|_2^2$. We will show that $\rho(M)$ is guaranteed to be less than one if we choose $\gamma_j \leq (NJ_{n(j)})^{-1}\ \forall\ j \in \mathcal{J}$. We have the following:

$$\begin{aligned}\|Yx\|_2 = \|\lambda^{\frac{1}{2}}\bar{A}^T\beta^{-\frac{1}{2}}\eta^{\frac{1}{2}}\gamma^{\frac{1}{2}}x\|_2 &= \Big[\sum_{r\in\mathcal{R}}\Big(\sum_{j\in\mathcal{J}}\lambda_r^{\frac{1}{2}}\bar{A}_{jr}\beta_j^{-\frac{1}{2}}\eta_j^{\frac{1}{2}}\gamma_j^{\frac{1}{2}}x_j\Big)^2\Big]^{\frac{1}{2}} \\ &\leq \Big[\sum_{r\in\mathcal{R}}\Big(\sum_{j\in\mathcal{J}}\lambda_r\bar{A}_{jr}^2\beta_j^{-1}\eta_j\gamma_j\Big)\|x\|_2^2\Big]^{\frac{1}{2}} \quad \text{by the Cauchy-Schwarz inequality} \\ &= \Big(\sum_{j\in\mathcal{J}}\tilde{\delta}_j\gamma_j\Big)^{\frac{1}{2}}\|x\|_2 \leq \Big(\sum_{j\in\mathcal{J}}\gamma_j\Big)^{\frac{1}{2}}\|x\|_2 \leq \Big(\sum_{j\in\mathcal{J}}\frac{1}{NJ_{n(j)}}\Big)^{\frac{1}{2}}\|x\|_2 \\ &= \Big(\sum_{n\in\mathcal{N}}\sum_{j\in\mathcal{J}}\frac{E_{jn}}{NJ_n}\Big)^{\frac{1}{2}}\|x\|_2 = \Big(\sum_{n\in\mathcal{N}}\frac{1}{N}\Big)^{\frac{1}{2}}\|x\|_2 = \|x\|_2.\end{aligned}$$

Therefore, $\rho(M) = \|M\|_2 = \max_{\|x\|_2=1}\|Yx\|_2^2 \leq 1$.

Since D is a diagonal matrix with positive terms, $D+M$ is symmetric and positive definite. Therefore, its eigenvalues are strictly positive, and it can be written in the form SKS^{-1} where K is a diagonal matrix with the same eigenvalues. The

maximum eigenvalue of $D+M$ is strictly less than 2 because

$$\rho(D+M) = \|D+M\|_2 \leq \|D\|_2 + \|M\|_2 = \rho(D) + \rho(M) < 2.$$

Hence, the terms of K lie in the interval $(0,2)$, and so the terms of $I-K$ lie in the interval $(-1,1)$. But these terms are the eigenvalues of $I-(D+M)$ because

$$S(I-K)S^{-1} = I - SKS^{-1} = I - (D+M).$$

Thus, the eigenvalues of the original iteration matrix lie in the interval $(-1,1)$, and the sequence $\tilde{f}^i_{(\gamma)}(x), i = 1,2,\ldots$, converges to $\tilde{c}$. ■

Remark. The convergence proved in Thm. 4.5 is based on synchronous iterations. To prove totally asynchronous convergence of the damped computation, it is sufficient to show that the iteration matrix $G = [I - \gamma(I - \tilde{\delta}) - \gamma\eta\beta^{-1}\bar{A}\lambda\bar{A}^T]$ corresponds to a weighted maximum norm contraction, or equivalently, that $\rho(|G|) < 1$, where $\rho(|G|)$ is the spectral radius of the matrix $|G|$ having as elements the absolute values $|g_{jk}|$ of the elements of G. The proof of Thm. 4.5 showed that with $\gamma_j \leq (NJ_{n(j)})^{-1}\ \forall\ j \in \mathcal{J}$, we have $\rho(G) < 1$. However, the off-diagonal entries of G are nonpositive, and its structure is such that no matter how small we make $\gamma > 0$, we cannot guarantee that $\rho(|G|) < 1$ without requiring the light load condition $\|\delta\|_{\tilde{M}} < 1$. Our conjecture is that under a partially asynchronous model [9], i.e., there is a fixed bound D on the amount of time by which the information used at a link can become outdated, the algorithm will converge if we use a small enough stepsize γ. As the asynchronism measure D or the number of links J increases, we would have to decrease γ to mitigate the effects of asynchronism.

4.6 Computational results

In this section, we explore the computation of the implied costs at one point in time for a given set of offered loads. We use the Erlang fixed point equations to obtain the route blocking probabilities, and then input the results to the implied cost calculations. Let c, c', and $\tilde{c}$ denote the solutions to (4.8), (4.11), and (4.18), respectively. The surplus values s and s' are computed according to (4.7) and (4.10), respectively. For our alternative approximation, we compute $\tilde{s}_r = w_r - \sum_{k\in\mathcal{J}} \bar{A}_{kr}\tilde{c}_k$. Because we use the same route blocking probabilities L in computing the revenue sensitivities

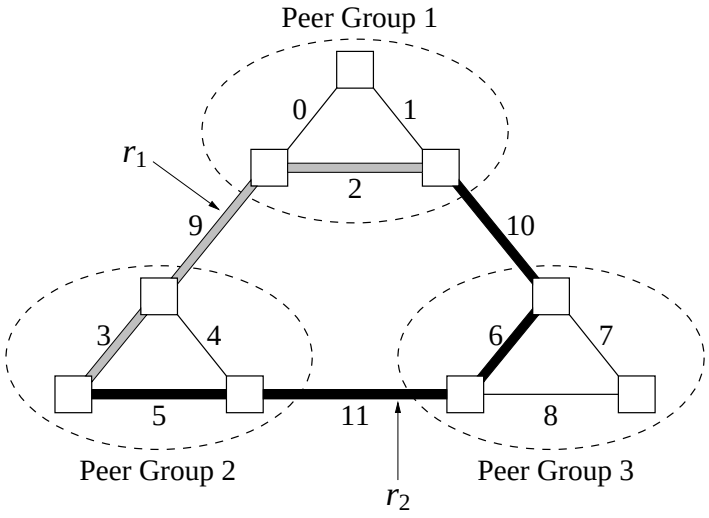


Figure 4.9: Symmetric network with a single level of aggregation.

for all three cases, the expected and maximum relative surplus value differences are equal to the expected and maximum relative revenue sensitivity errors. The results discussed below are summarized in Tables 4.2 and 4.4.

We start with the symmetric network shown in Fig. 4.9 and assign a capacity of 20 to each link. We define a total of 45 routes with offered loads ranging from 1.0 to 3.0 in such a way that the offered loads at each link in the three peer groups are the same and all transit routes use only one link in the peer groups that they pass through. Each accepted connection generates a revenue of 1.0. Under these conditions, the calculated implied costs c and c' are the same. Thus, $\|(s-s')/s\|_\infty = \max_{j,r:j\in r} |(s_r - s'_{r;\,j})/s_r| = 0$, and, as a result, the revenue sensitivities are also the same. For each link in the peer groups, $c_j = 0.015$. For the links connecting the peer groups, $c_j = 0.129$. Compared to our alternative approximation, the differences are quite small: $\|(c-\tilde{c})/c\|_\infty = 0.7\%$, and $\|(s-\tilde{s})/s\|_\infty = 0.04\%$.

Next, we take the symmetric case and increase the load on the links in peer group 1 to near capacity by increasing the offered loads for local routes in peer group 1 to three and a half times their previous values. This causes the implied cost calculations for c and c' to differ slightly, resulting in $\|(c-c')/c\|_\infty = 0.3\%$ and $\|(s-s')/s\|_\infty = 1.5\%$. Due to the heavy loads in peer group 1, the implied costs $\tilde{c}$ are not as accurate: $\|(c-\tilde{c})/c\|_\infty = 9.0\%$, and $\|(s-\tilde{s})/s\|_\infty = 97.5\%$. (Despite the latter result, we note that $\mathbb{E}[(s-\tilde{s})/s] = \sum_{r\in\mathcal{R}}(\nu_r(s_r-\tilde{s}_r)/s_r)/\sum_{r\in\mathcal{R}}\nu_r$ is only

15.0%.[8]) To demonstrate the change in revenue sensitivities from the symmetric case, consider the two alternative routes consisting of the following sets of links: $r_1 = \{2,9,3\}$ and $r_2 = \{10,6,11,5\}$. In the symmetric case, the revenue sensitives for r_1 and r_2 are 0.823 and 0.684, respectively. In the present overloaded case, the revenue sensitivities change to approximately 0.416 and 0.772, respectively.[9] The longer route is now favored because it avoids passing through the overloaded peer group. We note that, using our first hierarchical approximation, the revenue sensitivity may vary along a particular route depending on which link is making the calculation (due to the $s_{r;\,j}$ term). To be exact, all links of a route in a given peer group will compute the same sensitivity, but links of the route in a different peer group may compute a different value. For our current example, the revenue sensitivities vary only slightly along routes, on the order of 0.004 in the worst case.

As another example of an overload scenario, we start with the symmetric case and increase the loads on transit routes between peer groups 1 and 2 by one and a half times, causing link 9 to be near capacity. For this case, the differences between the first two approximations are greater than in the previous overload scenario, $\|(c-c')/c\|_\infty = 1.1\%$ and $\|(s-s')/s\|_\infty = 5.0\%$, but the surplus values $\tilde{s}$ fare much better: $\|(c-\tilde{c})/c\|_\infty = 18.1\%$ and $\|(s-\tilde{s})/s\|_\infty = 4.4\%$. This is due to the fact that the overloaded node consists of only a single link, mitigating the errors due to local averaging of transit route costs. The revenue sensitivities for r_1 and r_2 are approximately 0.335 and 0.686, respectively, which would cause the routing algorithm to send more traffic around the overload as desired. Compared to the previous case, there is greater variation in the revenue sensitivities along each route using s', on the order of 0.013 in the worst case.

For a fourth experiment with a more varied topology, we use the network shown in Fig. 4.5. We define a total of 122 routes with offered loads ranging from 0.1 to 2.0. Two routes are defined between each pair of switches except for the members of peer group 2 which have only one local route between each pair. As before, each accepted connection generates a revenue of 1.0. The link capacities are varied between peer groups: links in peer groups 1, 2, and 3 have capacities 25, 40, and 30, respectively, and the connecting links have a capacity of 35 each. Despite the loss of symmetry, the implied cost calculations are surprisingly close:

[8]Similarly, we define $\mathbb{E}[(c-\tilde{c})/c] = \sum_{j\in\mathcal{J}}(\rho_j(c_j-\tilde{c}_j)/c_j)/\sum_{j\in\mathcal{J}}\rho_j$.

[9]The revenue sensitivity values presented in this section are computed using the surplus values s. Using s' or $\tilde{s}$ results in slightly different values but the same relative ordering.

	Rev. sens. error: $\mathbb{E}[\cdot]$ / $\|\cdot\|_\infty$			
	$\frac{s-s'}{s}$	$\frac{s-\tilde{s}}{s}$	$\partial W/\partial \nu_1$	$\partial W/\partial \nu_2$
Symmetric load	0.0% / 0.0%	0.01% / 0.04%	0.823	0.684
Local overload	0.2% / 1.5%	15.0% / 97.5%	0.416	0.772
Transit overload	0.9% / 5.0%	1.0% / 4.4%	0.335	0.686
Asymmetric net	2.4% / 15.5%	2.2% / 15.5%	—	—

	Imp. cost error: $\mathbb{E}[\cdot]$ / $\|\cdot\|_\infty$				
	$\frac{c-c'}{c}$	$\frac{c-\tilde{c}}{c}$	$L_{\max}$	$\|\delta\|_M$	Iterations
Symmetric load	0.0% / 0.0%	0.5% / 0.7%	2.1%	0.297	5
Local overload	0.1% / 0.3%	5.7% / 9.0%	25%	0.764	10–13
Transit overload	0.4% / 1.1%	6.3% / 18.1%	16%	0.780	8–9
Asymmetric net	0.7% / 2.1%	1.9% / 6.2%	3.8%	0.327	6–7

Table 4.2: Computational results for the four experiments.

the worst-case differences are $\|(c-c')/c\|_\infty = 2.1\%$, $\|(c-\tilde{c})/c\|_\infty = 6.2\%$, and $\|(s-s')/s\|_\infty = \|(s-\tilde{s})/s\|_\infty = 15.5\%$.

Table 4.2 summarizes the main results of the four experiments. $L_{\max}$ is the maximum route blocking probability; the high values for the middle two experiments are for a local route in peer group 1 and a transit route from peer group 1 to 2, respectively. The iterations column denotes the range of iterations needed for convergence of the three implied cost computations. Note that the light load condition $\|\delta\|_M < 1$ holds in every case.

Two comments on the above experiments are in order. First, using our first hierarchical approximation scheme, one can unfortunately construct cases where the revenue sensitivities vary enough along a route to cause an ordering between alternative routes from the source's point of view that is different from that obtained in a flat network. This would cause the adaptive routing algorithm to temporarily shift offered loads in the wrong direction until the sensitivities became farther apart. As a result, the routing algorithm would adapt more slowly, but it is unclear whether this is a common or troubling situation. Second, the bound in Thm. 4.3 appears to be rather weak. It was too high by an order of magnitude in the two overload cases. In the fourth experiment, however, it was less than twice the actual value.

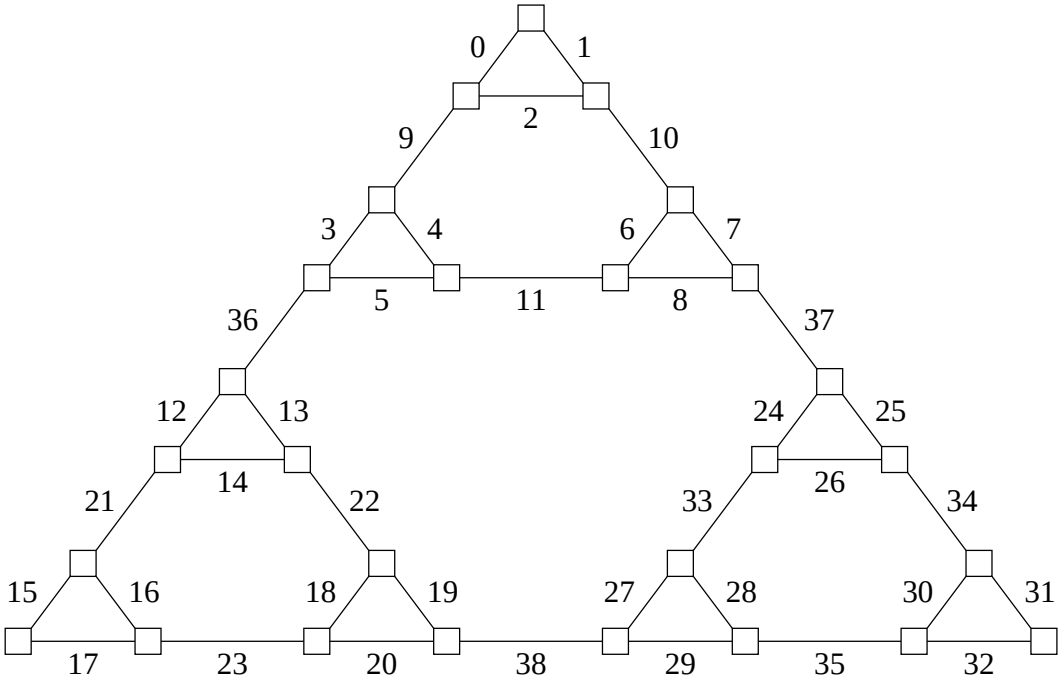


Figure 4.10: A larger symmetric network.

We also performed experiments on the larger network shown in Fig. 4.10 with a variable number of defined groups. The group memberships in terms of the links in each group are listed in Table 4.3. We define a total of 247 routes with offered loads ranging from 0.2 to 3.0. As before, each accepted connection generates a revenue of 1.0. The link capacities vary from 20 to 30, and no attempt was made to equalize the offered loads on the links.

Table 4.4 summarizes the main results of these six experiments. In terms of relative implied cost and revenue sensitivity errors, the 6 groups case performed the best, and the 6 alternate groups and 9 groups performed the worst. For these experiments (with fixed routes and offered loads), the error results seem to be correlated to the number of transit routes per group with a lower average number of transit routes tending to produce better results. We also compute the number of messages per iteration under the assumption that the groups of three switches in a triangle are connected locally using a broadcast medium, i.e., only one message is required to reach the three link controllers in the triangle. For a flat network, 807 messages per iteration are required, so each group structure tested provides a significant reduc-

3 groups	{0–11, 36} {12–23, 38} {24–35, 37}
6 groups	{0–11} {12–23} {24–35} {36} {37} {38}
6 alt. groups	{0–2, 9–10} {3–8, 11, 36} {12–14, 18–20, 22, 38} {15–17, 21, 23} {24–29, 33, 37} {30–32, 34–35}
9 groups	{0–2, 9} {3–5, 11, 36} {6–8, 10} {12–14, 21} {15–17, 23} {18–20, 22, 38} {24–26, 33, 37} {27–29, 35} {30–32, 34}
12 groups	{0–2, 9} {3–5, 11} {6–8, 10} {12–14, 21} {15–17, 23} {18–20, 22} {24–26, 33} {27–29, 35} {30–32, 34} {36} {37} {38}
21 groups	{0–2} {3–5} {6–8} {9} {10} {11} {12–14} {15–17} {18–20} {21} {22} {23} {24–26} {27–29} {30–32} {33} {34} {35} {36} {37} {38}

Table 4.3: Group memberships for the experiments on the larger network.

tion. The most savings occurs with the 6 alternate groups and the 9 groups which, as noted above, provide the worst performance in terms of revenue sensitivity error.

4.7 On-line measurements

We now return to the subject of on-line measurements, as briefly mentioned in Sec. 4.4. Instead of using the Erlang fixed point approximation, we show how estimates of the carried loads and blocking probabilities can be used to implement a hierarchical adaptive routing scheme. Our discussion follows that of Kelly [36], with additional optimizations to take advantage of the hierarchical framework.

We say that two routes have the same *hierarchical path* from the point of view of link j if they use the same set of links in peer group $n(j)$ and follow the same sequence of peer groups outside of $n(j)$. Let $\mathcal{H}_n$ be the set of hierarchical paths from the point of view of node n, and let H_{jh} be the amount of bandwidth used explicitly by hierarchical path $h \in \mathcal{H}_n$ on link j. (H_{jh} is 0 for all links j outside of n.) If we make the assumption that $w_{r_1} = w_{r_2}$ for two routes r_1 and r_2 with the same hierarchical structure from the point of view of link $j \in r_1, r_2$, then $s_{r_1;\,j} = s_{r_2;\,j}$. Recalling that $\rho_j(1-B_j) = \sum_{r\in\mathcal{R}} A_{jr}\lambda_r$ and $\delta_j = \eta_j\rho_j$, we can rewrite (4.11) as

$$c_j = \delta_j \sum_{h\in\mathcal{H}_{n(j)}} H_{jh} \frac{\text{flow carried on path } h}{\text{flow carried through link } j}(s_{h;\,j} + c_j), \quad j \in \mathcal{J}. \tag{4.26}$$

	Rev. sens. error: $\mathbb{E}[\cdot]$ / $\|\cdot\|_\infty$		Imp. cost error: $\mathbb{E}[\cdot]$ / $\|\cdot\|_\infty$	
	$\frac{s-s'}{s}$	$\frac{s-\tilde{s}}{s}$	$\frac{c-c'}{c}$	$\frac{c-\tilde{c}}{c}$
3 groups	3.7% / 63.9%	2.8% / 120.1%	0.7% / 2.9%	1.6% / 5.9%
6 groups	0.3% / 12.2%	0.7% / 16.2%	0.05% / 0.3%	1.7% / 3.9%
6 alt. groups	6.8% / 159.1%	7.1% / 163.1%	1.9% / 4.5%	4.9% / 8.3%
9 groups	10.1% / 136.8%	6.9% / 98.4%	4.0% / 9.6%	4.3% / 9.1%
12 groups	7.7% / 48.6%	4.1% / 46.7%	4.5% / 8.9%	4.2% / 7.7%
21 groups	2.7% / 13.5%	2.5% / 13.5%	1.0% / 2.9%	2.2% / 8.2%

	Messages per Iteration	Avg. Transit Routes per Group	Avg. Local Routes per Group
3 groups	303	14.7	75.0
6 groups	312	12.2	36.5
6 alt. groups	234	49.0	18.0
9 groups	249	43.9	9.7
12 groups	294	35.1	7.0
21 groups	447	31.0	3.1

Table 4.4: Computational results for the larger network.

Suppose we have on-line measures $\hat{\Lambda}_h(t)$ and $\hat{\Theta}_j(t)$ of the carried flows on path h and link j, respectively, over the interval $[t,t+1)$. Smoothed, moving-average estimates $\hat{\lambda}_h(t)$ and $\hat{\theta}_j(t)$ of the mean carried flows can be computed using the iterations

$$\hat{\lambda}_h(t+1) = (1-\gamma)\hat{\lambda}_h(t) + \gamma\hat{\Lambda}_h(t)$$
$$\hat{\theta}_j(t+1) = (1-\gamma)\hat{\theta}_j(t) + \gamma\hat{\Theta}_j(t)$$

where $\gamma \in (0,1)$. If we consider link j to be in isolation with Poisson traffic offered at rate ρ_j, we can estimate ρ_j (and thus δ_j) by solving the equation $\hat{\theta}_j = \rho_j[1 - E(\rho_j, C_j)]$ to obtain $\hat{\rho}_j$. Then we would have $\hat{\delta}_j = \hat{\rho}_j[E(\hat{\rho}_j, C_j - 1) - E(\hat{\rho}_j, C_j)]$.

Now suppose that the implied costs $\hat{c}$ and the associated surplus values $\hat{s}$ have been computed using these estimates and successive substitution. Suppose also that the blocking probability L_h has been estimated for each hierarchical path, possibly using a moving-average estimate similar to the above. The revenue sensitivity $(1 - \hat{L}_h)\hat{s}_{h;\,j}$ tells us the net expected revenue that a call on path h will generate from the perspective of link j. Traffic from a source to a given destination peer group should be split among the possible hierarchical paths based on these revenue sensitivities. A greater share of the traffic should be offered to a path that has a higher value of $(1 - \hat{L}_h)\hat{s}_{h;\,j}$ than the others. Also, if $(1 - \hat{L}_h)\hat{s}_{h;\,j}$ is negative for a particular path, that path should not be used since a net loss in revenue would occur by accepting connections on that path. Any adjustments of the splitting should be done gradually to prevent sudden congestion. Note that we have assumed that routes not satisfying the QoS constraints of a particular connection will be eliminated prior to choosing a path based on the revenue sensitivities.

4.8 Multiservice extensions

To accommodate different types of services, our model can be extended to a multirate loss network. Now we allow $A_{jr} \in \mathbb{Z}^+$. Several additional problems arise in this context. First and foremost, the Erlang B formula no longer suffices to compute the blocking probability at a link for each type of call. Let $\pi_j(n)$ denote the steady-state probability of n circuits being in use at link j. Then the blocking probability for route r at link j is $B_{jr} = \sum_{n=C_j-A_{jr}+1}^{C_j} \pi_j(n)$. We can compute π_j using a recursive formula of complexity $O(C_j K_j)$ where K_j denotes the number of traffic

classes (distinct values of $A_{jr} > 0$) arriving at link j [63]. This result was derived independently by Kaufman and Roberts. To reduce complexity, many asymptotic approximations have been proposed in the literature as the offered load and link capacity are scaled in proportion [30, 44, 51, 61, 67, 73]. We have found Mitra and Morrison's Uniform Asymptotic Approximation (UAA) [51] to be particularly accurate.

The Erlang fixed point approximation can be extended in a straightforward manner to the multiservice case using an appropriate blocking function at each link. Note that, in this case, the fixed point is no longer guaranteed to be unique [63].[10] Based on this approximation, implied cost equations can be derived [19, 52], where we now have a different implied cost at each link for each type of service. The straightforward extension to our hierarchical setting is to further compute an average implied cost for each type of service passing through each peer group. Computing a single average implied cost for each peer group is attractive but would probably result in an unacceptable loss in accuracy.

Define $\mathcal{S}$ to be the set of services offered by the network and partition $\mathcal{R}$ into sets $\mathcal{R}^s, s \in \mathcal{S}$. Let $s(r)$ denote the service type associated with route r.[11] Also, let $\rho_{jr} = \lambda_r/(1-B_{jr})$, and define $\eta_{jrq} = B_{jr}(\vec{\rho}_j, \vec{A}_j, C_j - A_{jq}) - B_{jr}(\vec{\rho}_j, \vec{A}_j, C_j)$, which is the expected increase in blocking probability at link j for route r given that A_{jq} circuits are removed from link j. The multiservice implied costs satisfy the following system of equations:

$$c_{jq} = \sum_{r: j \in r} \eta_{jrq} \rho_{jr} (s_{r;\,j} + c_{jr}), \quad j \in \mathcal{J},\ q \in \mathcal{R}, \tag{4.27}$$

where

$$s_{r;\,j} = w_r - \sum_{k \in r} P_{kj} c_{kr} - \sum_{n \neq n(j)} T_{nr}\, \bar{c}_{ns(r)} \tag{4.28}$$

and

$$\bar{c}_{ns} = \frac{\sum_{r \in \mathcal{R}^s} T_{nr} \lambda_r \left(\sum_{j \in r} E_{jn} c_{jr} \right)}{\sum_{r \in \mathcal{R}^s} T_{nr} \lambda_r}. \tag{4.29}$$

[10] Using a certain single-link blocking function, convergence to a unique fixed point was recently proved in the light load regime only [73].

[11] Note that when multiple service types are carried between two points, we assign various routes that may follow the same path.

Note that $c_{jr} = c_{jq}$ if $A_{jr} = A_{jq}$. In a large capacity network, we can further reduce (4.27) to a system of only J equations by employing the UAA [52]. If we redefine our norm on $\mathbb{R}^{JR}$ (R is the total number of routes) as

$$\|x\|_M = \max_{j,r:j\in r}\{\sum_{k\neq j:k\in r} P_{kj}|x_{kr}| + \sum_{n\neq n(j)} T_{nr}\overline{|x|}_{ns(r)}\}, \tag{4.30}$$

let $\delta = (\delta_{11}, \delta_{12}, \ldots, \delta_{1R}, \delta_{21}, \ldots, \delta_{JR})$ where $\delta_{jq} = \sum_{r:j\in r} \eta_{jrq}\rho_{jr}$, and define $\Delta = \max_{n,r}\{T_{nr}\sum_{m\neq n} T_{mr}|c_r^m - \overline{c}_{ms(r)}|\}$ where $c_r^m = \sum_{j\in r} E_{jm}c_{jr}$, then Thms. 4.1, 4.2, and 4.3 can be easily shown to hold for the multiservice case.

4.9 Chapter summary

This chapter is based on the premise that the use of hierarchical source routing is a key to both reducing complexity and providing acceptable QoS in a large-scale network. Although aggregating network elements into subnetworks is an old idea, we have taken a unique approach to representing the "available" capacity of a subnetwork by formulating an implicit representation based on the average implied cost to go through or into the subnetwork. This average implied cost reflects the congestion in the subnetwork and captures the interdependencies among traffic streams, a feature sorely lacking in explicit representations of available capacity.

We proved that both a synchronous and asynchronous distributed computation of the approximate implied costs will converge to a unique solution under a light load condition. Furthermore, we presented a more aggressive averaging mechanism that also performs local averaging among routes transiting through or into a local subnetwork. We proved that with sufficient damping, a synchronous distributed computation of these new approximate implied costs will converge to a unique solution under any traffic conditions. Our experimental results showed that these approximations are reasonably accurate.

Based on this representation for available subnetwork capacity, we proposed a hierarchical source routing algorithm that adaptively selects high-level routes so as to maximize network revenue. Prior to path selection, routes not likely to meet prespecified QoS constraints, such as end-to-end delay, are eliminated from consideration. Our scheme can incorporate on-line measurements, and it can be extended to a multiservice environment. The low-level routing within subnetworks was de-

liberately not specified, as we feel that some form of dynamic routing would be beneficial in coping with traffic fluctuations at that level.

Chapter 5

Conclusion

5.1 Summary of main results

The goal of this dissertation was to advance the state of the art in applying aggregation to large-scale communication networks in two domains: the aggregation of network flows and network elements. In the flow aggregation area, we first explored the benefits of aggregating multicast demands on VP trees. We proposed a pre- or post-processing step to the VP multicast layout problem, which either reduces the complexity of the required optimization or further improves upon obtained solutions, and we showed that it can be effectively reduce capacity requirements, balance network loads, and reduce the number of VP trees required.

Real networks have time-varying demands and finite signaling resources. We argued that signaling resources may not be sufficient for future demands on large-scale, connection-oriented networks, and we developed adaptive VP capacity allocation algorithms that are based on implied costs and address these constraints.

In some cases it is desirable to modify the manner in which flows are aggregated (the VP layout). In this context we investigated algorithms to migrate from one layout to another, and we found that for incremental changes, the potential for performance losses during migration in terms of call blocking is minimal. However, when dramatic changes in the VP layout are warranted, it is desirable to enhance performance by implementing a simple decentralized algorithm that we have proposed.

In the network aggregation area, we developed an implicit representation of the available capacity of a subnetwork which is based on a distributed compu-

tation of the average implied cost to go through or into the subnetwork. Such implied costs reflect the congestion in the subnetwork as well as the interdependencies among traffic streams in the network. We proved that both a synchronous and asynchronous distributed computation of the implied costs will converge to a unique solution under a light load condition. We also presented an alternative approximation that performs averaging among local transit routes in addition to remote averaging, and we proved that with sufficient damping, distributed computation of these new costs will converge to a unique solution under any traffic conditions. To assess accuracy, we derived a bound on the difference between our (original) implied costs and those calculated for a flat network, and our experiments showed that our costs are indeed quite accurate. In addition, we showed how on-line measurements can be incorporated into the computation, and we outlined extensions to a multiservice environment.

Based on this representation for available subnetwork capacity, we proposed a QoS-sensitive routing algorithm that is able to appropriately route high-level flows while significantly reducing complexity. The algorithm uses effective bandwidths to capture traffic behavior, and it adaptively selects hierarchical routes so as to maximize network revenue, while allowing low-level dynamic routing within subnetworks to respond to traffic fluctuations.

5.2 Application to other areas

The research for this dissertation was performed primarily in the context of ATM networks. We made use of VPs in the flow aggregation area, and PNNI routing motivated the framework for our hierarchical routing scheme. However, it is unclear at this time whether ATM will prevail as the network technology of choice. As argued in Chapter 1, whatever architecture prevails, it is likely that it will at least appear to be connection-oriented at the call level because of resource reservations and call admission. If this is the case, much of our work will be relevant to future broadband networks.

For instance, aggregation of multicast demands is useful in any context where multicast applications need resource reservations to meet their QoS requirements, e.g., an important videoconference or live surgical images multicast to several specialists at a busy time of day. In an IP/RSVP network, based on estimates

for demands for these types of multicast applications, network resources could be pre-reserved and shared among several core-based trees, increasing the success rate for receivers who subsequently request resources dynamically through RSVP. Our pre- or post-processing step could be applied to this "layout" of core-based trees. These types of layouts would also provide a suitable setting for adaptive capacity allocation and migration.

Our implicit representation for subnetwork congestion is also flexible in its application. Independent of the routing algorithm, it could be used to ease performance monitoring by network operators as well as to provide information valuable for determining the best location for future capacity upgrades and how much we should be willing to pay for them.

The proposed hierarchical routing algorithm is at the level of the Border Gateway Protocol (BGP) in the Internet [31]. BGP implements shortest path routing for packets, so many modifications would be needed to support our scheme. However, to cope with such issues as provider selection, charging for traffic, and resource reservations through RSVP, modifications in the direction of a source routing protocol for IP flows may be forthcoming making our algorithm more viable.

5.3 Future research directions

In the flow aggregation area, the adaptive VP capacity allocation scheme for a mixed VP/VC switching network would need more work if signaling capacities emerge as a binding constraint. Heuristics for choosing the threshold parameters and performance evaluation through simulations would be topics of importance.

In the network aggregation area, there are several topics for future research directly related to our routing algorithm, including

- extensions to more than two levels of hierarchy,
- the optimal subnetwork size and switch arrangement to achieve the best trade-off between accuracy and reduced overheads,
- the robustness of the implied costs and routing to link failures,
- investigation of the need to reserve capacity for local traffic using trunk reservation, and

- the role of our algorithm in a layered approach to IP over ATM routing [18].

One issue that has not been addressed at length in this dissertation is heterogeneity in QoS requirements. We have assumed a homogeneous model with unit bandwidth per connection and have outlined extensions to multiservice networks where multiple bandwidth classes are offered. Several additional questions arise in a multiservice environment such as what types of flows should we aggregate (video, video plus audio, web traffic, etc.)? Is it ever a problem that the Erlang fixed point is no longer guaranteed to be unique?

Another fundamental issue is the choice of network versus user optimization in QoS-sensitive routing. Our proposed algorithm lies in the class of network optimal algorithms. Greedy shortest path algorithms are user optimal in the sense that the best route is chosen from the user's perspective without regard to the overall system effect. Our general feeling is that shortest path routing of high-level flows may lead to instabilities in a large-scale network [24, 39]. If this is true, algorithms based on implied costs would be essential to ensuring good performance in future broadband networks.

Bibliography

[1] S. Ahn, R. P. Tsang, S.-R. Tong, and D. H. Du, "Virtual path layout design on ATM networks," in *Proc. IEEE Infocom*, vol. 1, pp. 192–200, 1994.

[2] A. Alles, "ATM internetworking," Cisco Systems, Inc. white paper (http://-www.cisco.com/warp/public/614/12.html), May 1995.

[3] M. H. Ammar, G. C. Polyzos, and S. K. Tripathi, eds., *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, Apr. 1997.

[4] N. G. Aneroussis and A. A. Lazar, "Virtual path control for ATM networks with call level quality of service guarantees," Tech. Rep. 410-95-16, Center for Telecommunications Research, Columbia University, New York, NY 10027, 1995.

[5] A. Ballardie, "Core Based Trees (CBT) multicast routing architecture," RFC 2201, Sept. 1997.

[6] A. Ballardie, "Core Based Trees (CBT version 2) multicast routing: Protocol specification," RFC 2189, Sept. 1997.

[7] F. Bauer and A. Varma, "Distributed algorithms for multicast path setup in data networks," *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 181–191, Apr. 1996.

[8] D. Bertsekas and R. Gallager, *Data Networks*, Englewood Cliffs, NJ: Prentice Hall, 2nd ed., 1992.

[9] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Englewood Cliffs, NJ: Prentice Hall, 1989.

[10] M. Borden, E. S. Crawley, B. S. Davie, and S. G. Batsell, "Integration of real-time services in an IP-ATM network architecture," RFC 1821, Aug. 1995.

[11] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP): Version 1 functional specification," RFC 2205, Sept. 1997.

[12] I. Chlamtac, A. Faragó, and T. Zhang, "Optimizing the system of virtual paths," *IEEE/ACM Transactions on Networking*, vol. 2, no. 6, pp. 581–586, Dec. 1994.

[13] R. Cole, D. Shur, and C. Villamizar, "IP over ATM: A framework document," RFC 1932, Apr. 1996.

[14] D. E. Comer, *Internetworking with TCP/IP, Vol. I: Principles, Protocols, and Architecture*, Englewood Cliffs, NJ: Prentice Hall, 2nd ed., 1991.

[15] M. de Prycker, *Asynchronous Transfer Mode: Solution for Broadband ISDN*, London: Prentice Hall, 3rd ed., 1995.

[16] G. de Veciana, G. Kesidis, and J. Walrand, "Resource management in wide-area ATM networks using effective bandwidths," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 6, pp. 1081–1090, Aug. 1995.

[17] L. M. Dudkin, I. Rabinovich, and I. Vakhutinsky, *Iterative Aggregation Theory*, New York: Marcel Dekker, 1987.

[18] P. Dumortier, "Toward a new IP over ATM routing paradigm," *IEEE Communications*, vol. 36, no. 1, pp. 82–86, Jan. 1998.

[19] A. Faragó, S. Blaabjerg, L. Ast, G. Gordos, and T. Henk, "A new degree of freedom in ATM network dimensioning: Optimizing the logical configuration," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1199–1206, Sept. 1995.

[20] V. J. Friesen, J. J. Harms, and J. W. Wong, "Resource management with virtual paths in ATM networks," *IEEE Network*, vol. 10, no. 5, pp. 10–20, Sept./Oct. 1996.

[21] J. Galambos, *The Asymptotic Theory of Extreme Order Statistics*, New York: John Wiley & Sons, 1978.

[22] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, New York: W. H. Freeman and Co., 1979.

[23] O. Gerstel, I. Cidon, and S. Zaks, "The layout of virtual paths in ATM networks," *IEEE/ACM Transactions on Networking*, vol. 4, no. 6, pp. 873–884, Dec. 1996.

[24] R. J. Gibbens, P. J. Hunt, and F. P. Kelly, "Bistability in communication networks," in *Disorder in Physical Systems* (G. R. Grimmet and D. J. A. Welsh, eds.), pp. 113–128, New York: Oxford University Press, 1990.

[25] R. J. Gibbens and F. P. Kelly, "Network programming methods for loss networks," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1189–1198, Sept. 1995.

[26] A. Girard, *Routing and Dimensioning in Circuit-Switched Networks*, Reading, MA: Addison-Wesley, 1990.

[27] M. Grossglauser, S. Keshav, and D. Tse, "RCBR: A simple and efficient service for multiple time-scale traffic," *Computer Communication Review (Proc. ACM SIGCOMM '95)*, vol. 25, no. 4, pp. 219–230, Oct. 1995.

[28] E. Guarene, P. Fasano, and V. Vercellone, "IP and ATM integration perspectives," *IEEE Communications*, vol. 36, no. 1, pp. 74–80, Jan. 1998.

[29] R. Guérin and A. Orda, "QoS-based routing in networks with inaccurate information: Theory and algorithms," in *Proc. IEEE Infocom*, vol. 1, pp. 75–83, 1997.

[30] J. Y. Hui, *Switching and Traffic Theory for Integrated Broadband Networks*, Boston: Kluwer Academic Publishers, 1990.

[31] C. Huitema, *Routing in the Internet*, Englewood Cliffs, NJ: Prentice Hall, 1995.

[32] R. Hwang, J. Kurose, and D. Towsley, "On call processing delay in high-speed networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, pp. 628–639, Dec. 1995.

[33] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, New York: John Wiley & Sons, 1991.

[34] F. Kamoun and L. Kleinrock, "Stochastic performance evaluation of hierarchical routing for large networks," *Computer Networks*, vol. 3, no. 5, pp. 337–353, Nov. 1979.

[35] F. P. Kelly, "Blocking probabilities in large circuit-switched networks," *Advances in Applied Probability*, vol. 18, no. 2, pp. 473–505, June 1986.

[36] F. P. Kelly, "Routing in circuit-switched networks: Optimization, shadow prices, and decentralization," *Advances in Applied Probability*, vol. 20, no. 1, pp. 112–144, Mar. 1988.

[37] F. P. Kelly, "Routing and capacity allocation in networks with trunk reservation," *Mathematics of Operations Research*, vol. 15, no. 4, pp. 771–793, Nov. 1990.

[38] F. P. Kelly, "Loss networks," *The Annals of Applied Probability*, vol. 1, no. 3, pp. 319–378, Aug. 1991.

[39] F. P. Kelly, "Network routing," *Philosophical Transactions of the Royal Society of London, Series A*, vol. 337, no. 1647, pp. 343–367, Dec. 1991.

[40] F. P. Kelly, "Notes on effective bandwidths," in *Stochastic Networks: Theory and Applications* (F. P. Kelly, S. Zachary, and I. B. Ziedins, eds.), pp. 141–168, New York: Oxford University Press, 1996.

[41] S.-B. Kim, "An optimal establishment of virtual path connections for ATM networks," in *Proc. IEEE Infocom*, vol. 1, pp. 72–79, 1995.

[42] S.-B. Kim, "An optimal VP-based multicast routing in ATM networks," in *Proc. IEEE Infocom*, vol. 3, pp. 1302–1309, 1996.

[43] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks," *Computer Networks*, vol. 1, no. 3, pp. 155–174, Jan. 1977.

[44] J.-F. P. Labourdette and G. W. Hart, "Blocking probabilities in multitraffic loss systems: Insensitivity, asymptotic behavior, and approximations," *IEEE Transactions on Communications*, vol. 40, no. 8, pp. 1355–1366, Aug. 1992.

[45] W. C. Lee, "Spanning tree method for link state aggregation in large communication networks," in *Proc. IEEE Infocom*, vol. 1, pp. 297–302, 1995.

[46] W. C. Lee, "Topology aggregation for hierarchical routing in ATM networks," *Computer Communication Review*, vol. 25, no. 2, pp. 82–92, Apr. 1995.

[47] W. C. Lee, M. G. Hluchyj, and P. A. Humblet, "Routing subject to quality of service constraints in integrated communication networks," *IEEE Network*, vol. 9, no. 4, pp. 46–55, July/Aug. 1995.

[48] D. H. Lorenz and A. Orda, "QoS routing in networks with uncertain parameters," in *Proc. IEEE Infocom*, vol. 1, pp. 3–10, 1998.

[49] M. H. MacDougall, *Simulating Computer Systems: Techniques and Tools*, Cambridge, MA: The MIT Press, 1987.

[50] D. Medhi, "Multi-hour, multi-traffic class network design for virtual path-based dynamically reconfigurable wide-area ATM networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, pp. 809–818, Dec. 1995.

[51] D. Mitra and J. A. Morrison, "Erlang capacity and uniform approximations for shared unbuffered resources," *IEEE/ACM Transactions on Networking*, vol. 2, no. 6, pp. 558–570, Dec. 1994.

[52] D. Mitra, J. A. Morrison, and K. G. Ramakrishnan, "ATM network design and optimization: A multirate loss network framework," *IEEE/ACM Transactions on Networking*, vol. 4, no. 4, pp. 531–543, Aug. 1996.

[53] M. Montgomery and G. de Veciana, "Hierarchical source routing through clouds," in *Proc. IEEE Infocom*, vol. 2, pp. 685–692, 1998.

[54] M. Montgomery and G. de Veciana, "Virtual path capacity allocation and migration," INFORMS Telecommunications Conference, Boca Raton, FL, Mar. 1998.

[55] D. Niehaus *et al.*, "Performance benchmarking of signaling in ATM networks," *IEEE Communications*, vol. 35, no. 8, pp. 134–143, Aug. 1997.

[56] A. Orda, "Routing with end to end QoS guarantees in broadband networks," in *Proc. IEEE Infocom*, vol. 1, pp. 27–34, 1998.

[57] A. Orda, G. Pacifici, and D. E. Pendarakis, "An adaptive virtual path allocation policy for broadband networks," in *Proc. IEEE Infocom*, vol. 1, pp. 329–336, 1996.

[58] S. Ramanathan, "Multicast tree generation in networks with asymmetric links," *IEEE/ACM Transactions on Networking*, vol. 4, no. 4, pp. 558–568, Aug. 1996.

[59] N. S. V. Rao and S. G. Batsell, "QoS routing via multiple paths using bandwidth reservation," in *Proc. IEEE Infocom*, vol. 1, pp. 11–18, 1998.

[60] F. S. Roberts, *Applied Combinatorics*, Englewood Cliffs, NJ: Prentice Hall, 1984.

[61] J. Roberts, U. Mocci, and J. Virtamo, eds., *Broadband Network Teletraffic: Performance Evaluation and Design of Broadband Multiservice Networks; Final Report of Action COST 242*, Berlin: Springer-Verlag, 1996.

[62] R. Rom, "PNNI routing performance: An open issue," in *Washington University Workshop on Integration of IP and ATM*, (http://www.arl.wustl.edu/arl/-workshops/atmip/proceedings.html), Nov. 1996.

[63] K. W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks*, London: Springer-Verlag, 1995.

[64] K. A. Ross, *Elementary Analysis: The Theory of Calculus*, New York: Springer-Verlag, 1980.

[65] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, Sept. 1995.

[66] S. Sigarto. Private communication, SBC Technology Resources, Inc., Mar. 1998.

[67] A. Simonian, J. W. Roberts, F. Théberge, and R. Mazumdar, "Asymptotic estimates for blocking probabilities in a large multi-rate loss network," in *Proc. 33rd Annual Allerton Conf. on Communication, Control, and Computing*, pp. 726–735, 1995.

[68] M. E. Steenstrup, ed., *Routing in Communication Networks*, Englewood Cliffs, NJ: Prentice Hall, 1995.

[69] C.-F. Su and G. de Veciana, "On statistical multiplexing, traffic mixes, and VP management," in *Proc. IEEE Infocom*, vol. 2, pp. 643–650, 1998.

[70] R. Syski, *Introduction to Congestion Theory in Telephone Systems*, vol. 4 of *Studies in Telecommunication*, Amsterdam: Elsevier Science, 2nd ed., 1986.

[71] The ATM Forum, "Private Network-Network Interface specification version 1.0," ftp://ftp.atmforum.com/pub/approved-specs/af-pnni-0055.000.pdf, Mar. 1996.

[72] The ATM Forum, "Traffic management specification version 4.0," ftp://-ftp.atmforum.com/pub/approved-specs/af-tm-0056.000.pdf, Apr. 1996.

[73] F. Théberge and R. R. Mazumdar, "New reduced load heuristic for computing blocking in large multirate loss networks," *IEE Proceedings: Communications*, vol. 143, no. 4, pp. 206–211, Aug. 1996.

[74] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, Sept. 1996.

[75] J. P. Wong and R. K. Pankaj, "A diameter based method for virtual path layout in ATM networks," in *Broadband Communications: Global Infrastructure for the Information Age* (L. Mason and A. Casaca, eds.), pp. 501–512, London: Chapman & Hall, 1996.

[76] W.-L. Yang, "Estimation and abstraction of available capacity in large-scale networks," Master's thesis, The University of Texas at Austin, Austin, TX 78712, 1997.

[77] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource ReSerVation Protocol," *IEEE Network*, vol. 7, no. 5, pp. 8–18, Sept. 1993.

Vita

Michael Charles Montgomery was born in Oak Ridge, Tennessee on May 25, 1971, the son of Charles and Linda Montgomery. He graduated from Oak Ridge High School in June 1989 and enrolled at Virginia Tech in the fall. As an undergraduate, he majored in computer engineering and graduated with Summa Cum Laude honors in May 1993. During the summers of 1991 and 1992, he worked in the Scientific Workstation Support group at Oak Ridge National Laboratory.

A week after receiving his Bachelor of Science degree, Michael married Heather Dugan, a native of Blacksburg and also a student at Virginia Tech. He began graduate studies in electrical engineering that summer and completed his Master of Science degree in July 1994. He enrolled at the University of Texas at Austin that fall and completed his Ph.D. in electrical engineering in August 1998. He is now a member of the Network Research group at Oak Ridge National Laboratory.

Permanent Address: 126 Newhaven Road, Oak Ridge, Tennessee 37830

This dissertation was typeset with LaTeX 2ε[1] by the author.

[1] LaTeX 2ε is the latest version of LaTeX, a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX program. TeX is a trademark of the American Mathematical Society.