ABSTRACT

Title of Dissertation: NETWORK PERFORMANCE MODELING, DESIGN AND DIMENSIONING METHODOLOGIES

Mingyan Liu, Doctor of Philosophy, 2000

Dissertation directed by: Professor John S. Baras
Electrical and Computer Engineering

Providing accurate estimates of performance in large heterogeneous internetworks, for the purposes of network design and planning and for service provisioning has become a critical problem. This is due to the heterogeneity of the physical medium, the size of current and future networks and the different quality of service requirements for multimedia services. In this thesis we describe our work on the development of connection level mathematical models used for estimating network performance characteristics such as throughput, delay and blocking probability. The type of models we use and investigate are of the "Loss Network" type, which have been used widely in legacy telephone networks and in cellular networks for estimating "availability". These network models can also be used in estimating performance in general packet-switched networks using effective bandwidth concepts. In particular, these models are directly applicable in studying connection blocking for networks using QoS routing schemes. Computational complexity is a serious usability bottleneck for such algorithms. We describe fast (two to three orders of magnitude faster than discrete event simulation) approximation algorithms we have developed for accurately estimating blocking probability in a random topology network, using state-dependent routing, with multiple classes of traffic. We describe even faster algorithms based on a hierarchical loss network model we have developed. The latter are well matched to networks that have a natural hierarchical architecture, or which use some form of hierarchical routing to further reduce computational cost. We also developed models for networks using delay-based QoS routing. An important objective of our work is to demonstrate the utility of these models for effective design and dimensioning of a large network so that a certain set of QoS requirements are met. We show how typical design and dimensioning problems can be formulated as a multi-objective constrained optimization problem, using performance estimation network models; an approach that leads naturally to very useful trade-off analysis. We describe our research in the development of a general network design and dimensioning methodology by linking our performance models with Automatic Differentiation and multi-objective optimization algorithms and tools. We present examples and applications that demonstrate the speed and versatility of our methodology and algorithms.

NETWORK PERFORMANCE MODELING, DESIGN AND DIMENSIONING METHODOLOGIES

by

Mingyan Liu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2000

DEDICATION

To my Mom and Dad.

ACKNOWLEDGMENTS

I would like to take this opportunity to express my gratitude to all those who have given me advice, help and support throughout my work. Foremost among these is my advisor Professor John S. Baras.

I am very grateful to Professors André L. Tits and Armand M. Makowski for their help and advice with this work. I am very grateful to Professors William S. Levine and A. Udaya Shankar for kindly consenting to join the advisory committee and review this thesis.

I am also grateful to the Center for Satellite and Hybrid Communication Networks for the support of my research. This work was supported under NASA contract 01-5-26412, and under US Department of the Army Army Research Laboratory Cooperative Agreement DAAL 01-4-33109 (under the Federated Lab Advanced Telecommunications and Information Distribution Research Program).

Acknowledgment also goes to my friends and co-workers for their help and encouragement, especially Manish Karir.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Title of Dissertation: NETWORK PERFORMANCE MODELING, DESIGN AND DIMENSIONING METHODOLOGIES

Mingyan Liu, Doctor of Philosophy, 2000

Dissertation directed by: Professor John S. Baras

Electrical and Computer Engineering

Providing accurate estimates of performance in large heterogeneous internetworks, for the purposes of network design and planning and for service provisioning has become a critical problem. This is due to the heterogeneity of the physical medium, the size of current and future networks and the different quality of service requirements for multimedia services. In this thesis we describe our work on the development of connection level mathematical models used for estimating network performance characteristics such as throughput, delay and blocking probability. The type of models we use and investigate are of the "Loss Network" type, which have been used widely in legacy telephone networks and in cellular networks for estimating "availability". These network models can also be used in estimating performance in general packet-switched networks using effective bandwidth concepts. In particular, these models are directly applicable in studying connection blocking for networks using QoS routing schemes. Computational complexity is a serious usability bottleneck for such algorithms. We describe fast (two to three orders of magnitude faster than discrete event simulation) approximation algorithms we have developed for accurately estimating blocking probability in a random topology network, using state-dependent routing, with multiple classes of traffic. We describe even faster algorithms based on a hierarchical loss network model we have developed. The latter are well matched to networks that have a natural hierarchical architecture, or which use some form of hierarchical routing to further reduce computational cost. We also developed models for networks using delay-based QoS routing. An important objective of our work is to demonstrate the utility of these models for effective design and dimensioning of a large network so that a certain set of QoS requirements are met. We show how typical design and dimensioning problems can be formulated as a multi-objective constrained optimization problem, using performance estimation network models; an approach that leads naturally to very useful trade-off analysis. We describe our research in the development of a general network design and dimensioning methodology by linking our performance models with Automatic Differentiation and multi-objective optimization algorithms and tools. We present examples and applications that demonstrate the speed and versatility of our methodology and algorithms.

NETWORK PERFORMANCE MODELING, DESIGN AND DIMENSIONING METHODOLOGIES

by

Mingyan Liu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2000

DEDICATION

To my Mom and Dad.

ACKNOWLEDGMENTS

I would like to take this opportunity to express my gratitude to all those who have given me advice, help and support throughout my work. Foremost among these is my advisor Professor John S. Baras.

I am very grateful to Professors André L. Tits and Armand M. Makowski for their help and advice with this work. I am very grateful to Professors William S. Levine and A. Udaya Shankar for kindly consenting to join the advisory committee and review this thesis.

I am also grateful to the Center for Satellite and Hybrid Communication Networks for the support of my research. This work was supported under NASA contract 01-5-26412, and under US Department of the Army Army Research Laboratory Cooperative Agreement DAAL 01-4-33109 (under the Federated Lab Advanced Telecommunications and Information Distribution Research Program).

Acknowledgment also goes to my friends and co-workers for their help and encouragement, especially Manish Karir.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

The main focus of this work centers around the connection level study of network performance analysis and network design and dimensioning. There are two sets of questions that interest us and that we attempt to answer by this study:

- For an existing network design, i.e., topology and connectivity, network resource allocation, e.g., link capacity & buffer size, user traffic pattern, certain routing schemes, etc., how to estimate the end-to-end performance/Quality of Service (QoS) including throughput, delay, and loss probability? Or when certain performance bounds are guaranteed by the routing scheme (QoS routing), to what extent is the routing successful under the given network scenario?

- Given a set of QoS requirements, how to dimension/provision a network so that these requirements can be met? Furthermore, how to optimize network design while guaranteeing the QoS?

These questions are of great interest to network engineers in such issues as: Assessing an existing network regarding the number of users it can support with certain QoS levels in order to decide whether a network is under-utilized or congested; Determining the size and resource allocation of a corporate network prior to deployment; Deciding how much bandwidth an ISP should lease off the backbone, given the number of existing/anticipated subscribers and the guaranteed QoS.

These two sets of problems are closely related and more often than not the answer to the second problem relies on how we solve the first one. Consequently this thesis is roughly divided into two parts. In the first part, we study network connection level mathematical models for performance estimates. In the second part we combine the models we developed with various numerical techniques to approach the network design problem.

The family of performance models we analyze in this thesis is called *Loss Network Models*, which originated from circuit switched network (Details are given in Section 1.2 Literature Review). Therefore, the performance estimates it gives is of the blocking probability type, which is one of the primary performance metric in telephone networks. The detailed model varies with the specific routing scheme under consideration. In recent years, studies claim that this technology can be applied to packet switched networks as well, via the technique of effective bandwidth [1, 2]. By encapsulating cell/packet level behavior using the notion of effective bandwidth, one can study connection level performance using methods originated from circuit switched networks under certain conditions.

We observe that this type of modeling can be used to study performance for data networks employing QoS routing. The requirement for timely delivery of digitized audio-visual information has become increasingly important for next-generation integrated services broadband networks. The notion of quality of service (QoS) has been proposed to capture the qualitatively or quantitatively defined performance contract between the service provider and the user applications. The goal of QoS routing is to satisfy requested QoS requirements for every admitted connection and achieve global efficiency in resource allocation, by selecting network routes with sufficient resources for the requested QoS parameters (Details on QoS routing is given in the subsequent subsection). QoS routing is different from traditional best-effort routing. The former is normally connection-oriented with resource reservation to provide the guaranteed service. The latter can be either connection-oriented or connectionless with dynamic performance subject to the current availability of shared resources. Meeting the QoS requirement of each individual connection and reducing the call-blocking rate are important in QoS routing, while fairness, overall throughput, and average response time are the essential issues in traditional routing.

Therefore, our focus in this thesis is on the emerging integrated services data network employing QoS routing, and in the following chapters we will develop and apply models that gives estimates on connection blocking using different QoS routing schemes.

In the next section we will give a detailed literature review on related work to give necessary background and also show what's been done and what remains open. Our motivation for this work and our main contributions are summarized in Section 3. Section 4 concludes this chapter with organization of the thesis.

1.2 Literature Review

In this section we provide background on connection/call blocking study, QoS routing and network design and dimensioning. Throughout this thesis we will use "connection" and "call" interchangeably.

1.2.1 Connection Blocking

There has been extensive research in technologies of design and optimization of traditional circuit switched networks, e.g., [3, 4, 5, 6]. The basic model underlying most of these techniques is the loss network model [7, 8], which gives estimates on network blocking probabilities. A loss network is basically a circuit switched network, where a call requires certain amount of bandwidth on every link on a path between the source and the destination. If the network has the required bandwidth on those links when it gets the request, the call is admitted and it will be using the requested capacities for some time; otherwise the call is rejected. Blocking probability associated with the loss network is the probability that a call finds the network unavailable when it arrives and is thus rejected.

A telephone system is a typical loss network. An ATM network can also be viewed as a loss network with virtual circuits at the connection level. In this study we also treat data networks with QoS routing as a loss network, since QoS naturally provides call admission control and connections whose QoS requirements can not be satisfied by the routing are either rejected or renegotiated (see details in the next subsection).

Erlang's formula [9]

$$E(\nu, C) = \frac{\nu^C}{C!}[\sum_{n=0}^{C} \frac{\nu^n}{n!}]^{-1} \tag{1.1}$$

gives exact theoretical results on the loss probability estimates for a single link with single traffic class with no call admission control. Calls arrive at a link as a Poisson process of rate ν. The link comprises C circuits, and a call is blocked and lost if all C circuits are occupied. Otherwise the call is accepted and occupies a single circuit for the holding period of the call. Call holding periods are independent of each other and of arrival times and are identically distributed with unit mean. Erlang's formula gives the proportion of calls that are lost. If call holding periods are exponentially distributed, the formula gives the steady state probability that all C circuits are busy. This result is also insensitive to the distribution of call holding times, given that the arrival of calls is Poisson.

This formula has become the fundamental result for most of the later work in studying a single link system. Analytically, when there are multiple traffic classes, with different bandwidth requirements, a link can be modeled as a multi-dimensional Markov process with the state space dimension being the number of classes. When call admission control is used, generally the product form no longer holds. The other end of the spectrum of study in this area is the network models – by definition, with multiple links. When fixed routing is used, a network can be modeled as a multi-dimensional Markov process with the dimension being the product of the number of routes allowed in the network and the number of traffic classes. When alternative routes are present in addition to fixed routes, again, the product form is destroyed. In both cases the equilibrium state probability can be obtained by writing out the entire set of detailed balance equations and solving it. This approach is obviously not practical in dealing with large networks with hundreds of thousands of routes and integrates services with multiple service rates due to prohibitive computational complexity. Thus research focuses have been on special approximation and computational techniques for obtaining the loss probabilities. The computational complexity of loss networks is detailed in [10].

We describe work in both single link models and network models in the following two subsections, respectively.

1.2.1.1 Single Link System

Erlang's formula (1.1) gives blocking estimate for single unbuffered shared resource (single link) with single class of traffic. When multiple service rates are present, i.e., with traffic having different bandwidth requirements, the problem is formulated as follows.

Consider a single link with capacity C units of bandwidth. S classes of traffic arrive each as an independent Poisson process with rate λ_s, unit mean duration and each with a bandwidth requirement b_s. Let $\mathbf{n} = (n_1, n_2, ..., n_S)$ denote the number of calls of each class in progress on the link, and let $\mathbf{b} = (b_1, b_2, ..., b_S)$. Assuming no call admission control is used, the state distribution is then given by

$$p(\mathbf{n}) = G^{-1}(C) \prod_{s=1}^{S} \frac{\lambda_s^{n_s}}{n_s!}, \quad \mathbf{n} \in \Omega(C), \tag{1.2}$$

where

$$\Omega(C) = \{\mathbf{n} \in \mathcal{Z}_+ : 0 \leq \mathbf{n} \cdot \mathbf{b} \leq C\}$$

and

$$G(C) = \sum_{\mathbf{n} \in \Omega(C)} \prod_{s=1}^{S} \frac{\lambda_s^{n_s}}{n_s!} \tag{1.3}$$

is the normalizing constant (or partition function). Note that the Erlang's formula (1.1) is a single class special case of (1.2). The blocking experienced by class-s type of traffic is given by

$$B_s = 1 - \sum_{\mathbf{n} \in \Omega(C - b_s)} p(\mathbf{n}) = 1 - \frac{G(C - b_s)}{G(C)}. \tag{1.4}$$

However solving this simple explicit form is nontrivial. Because the cardinality of Ω grows roughly as C^S, even relatively modest size problems rule out a brute force computation of the normalizing constant (1.3).

Kaufman [11] developed an efficient recursion that completely bypasses this problem by considering the random variable $n = \mathbf{n} \cdot \mathbf{b}$, which is the total number of bandwidth units occupied. Thus the distribution of n is given by

$$p(n) = \sum_{\{\mathbf{n}:\mathbf{n}\cdot\mathbf{b}=n\}} \prod_{s=1}^{S} \frac{\lambda_s^{n_s}}{n_s!} \cdot G^{-1}(C), \tag{1.5}$$

and it can be computed via the simple recursion

$$\begin{aligned} np(n) &= \sum_{s=1}^{S} \lambda_s b_s p(n-b_s), \quad n = 0, 1, ..., C \\ \sum_{n=0}^{C} p(n) &= 1, \quad p(x) = 0 \text{ for } x < 0. \end{aligned} \tag{1.6}$$

Mitra and Morrison [12, 13] further developed Uniform Asymptotic Approximation(UAA) and Refined Uniform Asymptotic Approximation(RUAA) for loss probabilities specifically aiming at large link capacity and large arrival rate, when Kaufman's recursion becomes computational costly.

When call admission control is used, one needs special approximation to count for the call admission scheme. Call admission control is a way of regulating traffic within a network, by determining whether to accept or reject a new call at the time of connection setup according to the current load and the available resources of the network.

In circuit switched networks, one common (also the most frequently studied) way of call admission control is *trunk reservation* [7], where a call is only admitted when the free bandwidth on the links that the call requests is above a certain level. In a loss network, if alternative routes use more network resources than first-choice routes, then allowing a blocked call to attempt an alternative route may actually increase the loss probability of a network, and this effect may become even more pronounced if a blocked call can attempt a sequence of alternative routes. An explanation for this phenomenon is that if a link accepts an alternatively routed call, it may later have to block a directly routed call which will then attempt to find a two-hop or multi-hop route elsewhere in the network. A natural response would be to for the link to reject an alternatively routed call if the free circuits on the link are below a certain level – trunk reservation.

Trunk reservation has been extensively studied, e.g., [14] and [15, 16]. Numerous approximation schemes have been proposed based on the kaufman recursion for the model with trunk reservation, e.g., Bean, Gibbens and Zachary [17, 18].

A probabilistic call admission control policy is proposed and discussed by Medhi in [19, 20]. Krishnan derived a form of call admission control from a single link approximation in [21] via the concept of *Howard relative costs* [22]. In [23], Dasylva and Srikant gave performance bounds for any coordinate-convex call admission policy on a single link.

In data networks employing QoS routing, admission control is often considered a by-product of QoS routing and resource reservation. If resource reservation is successfully done along the route(s) selected by the routing algorithm, the connection request is accepted; otherwise the request is rejected.

1.2.1.2 Routing and End-to-end Loss Network Model

Generally a specific routing policy is an essential part of an end-to-end model formulation. First consider the simplest case with fixed routing where there is a predetermined route between any source-destination node pair.

Suppose a network has J links, indexed by $j = \{1, 2, ..., J\}$, and each link has capacity C_j. Let $\mathcal{R}$ be the set of all possible routes and a call on route r uses b_{jr} units of bandwidth from link j, where $b_{jr} \in \mathbf{Z}_+$, and matrix $A = (b_{jr}, j = 1, 2, ..., J, r \in \mathcal{R})$. Calls requesting route r arrive as a Poisson stream of rate λ_r and independent unit mean holding time. A call requesting route r is blocked if on any link j there are fewer than b_{jr} units of bandwidth free.

Let $n_r(t)$ be the number of calls in progress at time t on route r, and define the vectors $\mathbf{n}(t) = (n_r(t), r \in \mathcal{R})$ and $\mathbf{C} = (C_1, C_2, ..., C_J)$. Then the stochastic process $(\mathbf{n}(t), t \geq 0)$ has a unique stationary distribution and under this distribution $\pi(\mathbf{n}) = \text{Prob}\{\mathbf{n}(t) = \mathbf{n}\}$ is given by (similar to the single link system stationary distribution)

$$\pi(\mathbf{n}) = G^{-1}(\mathbf{C}) \prod_{r \in \mathcal{R}} \frac{\lambda_r^{n_r}}{n_r}!, \quad \mathbf{n} \in \Omega(\mathbf{C}), \tag{1.7}$$

where

$$\Omega(\mathbf{C}) = \{\mathbf{n} \in \mathcal{Z}_+ : A\mathbf{n} \leq \mathbf{C}\} \tag{1.8}$$

and $G(\mathbf{C})$ is the normalizing constant (partition function)

$$G(\mathbf{C}) = \left(\sum_{\mathbf{n} \in \Omega(\mathbf{C})} \prod_{r \in \mathcal{R}} \frac{\lambda_r^{n_r}}{n_r}! \right). \tag{1.9}$$

And the stationary probability that a call requesting route r is lost can be expressed as:

$$B_r = 1 - \sum_{\mathbf{n} \in \Omega(\mathbf{C} - Ae_r)} \pi(\mathbf{n}) = 1 - G^{-1}(\mathbf{C}) G(\mathbf{C} - Ae_r), \tag{1.10}$$

where $e_r = (I[r' = r], r' \in \mathcal{R})$ is the unit vector describing just one call in progress on route r.

Once again we are facing the impractical task of computing G directly. Therefore a theme of recent work and interest has been to find approaches and/or approximations to the model which avoid these computational problems. It can be shown [7] that under the limiting regime in which the capacities $C_j, j = 1, 2, ..., J$, and the offered traffic load $\lambda_r, r \in \mathcal{R}$ are increased together (with ratios C_j/λ_r held fixed), there is a parameter L_j associated with link j, such that

$$1 - B_r \longrightarrow \prod_{j=1}^{J} (1 - L_j)^{A_{jr}}, \quad , r \in \mathcal{R}. \tag{1.11}$$

It is as if links block independently, link j blocking with probability L_j.

This result has led to a class of approximation procedures known as the *reduced load approximation* or *fixed point approximation*. As a special case for example, for a loss network with fixed routing, we can derive the Erlang fixed point as follows.

$$\begin{aligned} E_j &= E(\rho_j, C_j), \quad j = 1, 2, ..., J, \\ \rho_j &= \sum_r A_{jr} \lambda_r \prod_{i = r - \{j\}} (1 - E_i)^{A_{jr}} \end{aligned} \tag{1.12}$$

where E is Erlang's formula (1.1). The fixed point (ρ_j, E_j) is achieved by iteration.

And the approximation for the loss probability on route r is given by

$$1 - B_r = \prod_j (1 - E_j)^{A_{jr}}. \tag{1.13}$$

Here the two underlying assumptions are

- *link independence assumption*, which assumes that the blocking occurs independently from link to link, so that the probability that a call is accepted on a certain route is the product of the probabilities that the call is accepted on each individual link on that route;

- *Poisson assumption*, which assumes that the traffic flow to each individual link is Poisson and that the corresponding traffic rate is the original offered rate thinned by blocking on other links of that path, thus called the reduced load.

In general a reduced load approximation is constructed as follows [7]. Model link $j, j = 1, 2, ..., J$, as if calls on routes passing through link j arrive as independent Poisson streams at given reduced loads, calls on route r require A_{jr} circuits and call holding times are exponential. The resulting finite state Markov chain provides blocking probabilities at link j for each route passing through link j, as a function of the reduced loads on link j. Finally, calculate the reduced load on a link by assuming that blocking events are independent from link to link along each route.

In addition to fixed routing, routing policies commonly studied include fixed alternative routing, sequential routing and state-dependent routing. In [7], Kelly gave a very good summary and review on work in formulation reduced load models for fixed routing and fixed alternative routing. State-dependent routing is also studied and special networks, e.g., fully connected, symmetric networks, were considered. Work in fixed routing or sequential routing also includes [24, 25, 9]. Greenberg and Srikant in [25] developed an approximation scheme for multi-rate, multi-hop loss network with sequential routing via the concept of network reliability, but at the same time introduced additional computation in solving the network reliability problem.

In recent years, more and more research has been done on state-dependent routing due to deployment of real systems like AT&T long-distance domestic network [9, 26], as well as the adaptive nature of state-dependent routing [6, 15]. The most common form of state-dependent routing is *Least Loaded Routing* (LLR), where one tries to pick the least congested route. The Residual Capacity Routing used in [27] also belongs to this category. As we will see from the next section, LLR is also a type of QoS routing proposed for integrated services data networks. In [15] Mitra and Gibbens presents a general form of state-dependent routing, the aggregated-least-busy-alternative routing, where the states of each link are lumped into aggregates, and the route of each call is determined by local information on the aggregate states of the links of the alternate routes at the time of the call's arrival. One of the extremes of this routing scheme, where there is no aggregation, is least loaded routing. However, the end-to-end model proposed by this work does not easily generalize to networks with routes longer than two hops.

There has also been extensive research on finding the optimal dynamic routing strategy and/or finding the performance bounds [28, 29, 7, 30, 16].

1.2.2 QoS Routing

We include this section on introduction to QoS routing for self-sufficient purposes, since we model and study data networks that use QoS routing for performance analysis. Most of the material in this section are taken from [31].

The timely delivery of digitized audio-visual information over local or wide area networks is now becoming realistic, due to fruitful research in high-speed networks, image processing, and video/audio compression. On the other hand, the emerging distributed multimedia applications also raise new challenges for networking. Currently on the Internet, data packets of a session may follow different paths to the destination. The network resources are shared by packets from different sessions. However, this architecture does not meet the requirements of future integrated services networks that will carry heterogeneous data traffic. First, it does not support resource reservation, which is vital for provisioning of guaranteed end-to-end performance (bounded delay, bounded delay jitter, and/or bounded loss ratio). Second, data packets may experience unpredictable delays and arrive at the destination out of order, which is undesirable for continuous real-time media. Hence the next generation of high-speed wide area networks are likely to be connection-oriented for real-time traffic.

The notion of quality of service (QoS) has been proposed to quantify the performance contract between service and user application. The QoS requirement of a connection is given as a set of constraints, which can be link constraints or path constraints. For unicast, the goal of QoS routing is to find a path that meets the requirement of a connection between two users. For multicast the problem is to find a tree, rooted at a sender, which covers all receivers with every internal path from the sender to a receiver satisfying the requirement. For the purpose of this thesis we only focus on unicast QoS routing.

There are two major types of QoS routing. One is called *link-optimization/constrained routing*, aimed at QoS metrics such as residual bandwidth and residual buffer space, where the state of a path is determined by the state of the bottleneck link. An example of bandwidth-optimization/constrained routing, which is to find a path that has the largest bandwidth/above-required-value bandwidth on the bottleneck link. The other type is called *path-optimization/constrained routing*, aimed at QoS metrics such as delay, delay jitter and cost, where the state of a path is determined by the combined state over all links on the path. An example of path-optimization routing is least-cost routing, which to find a path whose total cost is minimized. An example of path-constrained routing is the delay-constrained routing, which is to find a path whose delay is bounded by a required value.

Many composite routing problems can be derived from the above four basic types. As we have pointed out before, meeting the QoS requirement of each individual connection and reducing the call-blocking rate are important in QoS routing, while fairness, overall throughput, and average response time are the concerns in traditional best-effort routing. If resource reservation is successfully done along the route(s) selected by the routing algorithm, the connection request is accepted. If a QoS routing algorithm fails to find a feasible path for a new connection, the system can either reject the connection or negotiate with the application for a looser QoS constraint.

There are three routing strategies: source routing, distributed routing and hierarchical routing. They are classified according to how the state information is maintained and how the search of feasible paths is carried out. In the following three chapters we will develop performance models for Least-Loaded/bandwidth-optimized routing, the delay-based QoS routing and hierarchical routing.

1.2.3 Network Design and Dimensioning

Network design relates to determining the size, connectivity, configuration and capacity/resource allocation of a network. Network dimensioning on the other hand is a problem of figuring out the number of users a network can support with a certain level of QoS. These problems are closely related to network performance evaluation and studies in bother areas are often carried out side by side, as can be shown by the references in the previous sections. While performance evaluation can be used for analyzing existing networks, our ultimate goal is to use performance evaluation for network optimization, design and dimensioning.

Typically, a design and dimensioning model is formulated into a constraint optimization problem. The objective function can be an overall network revenue in terms of network deployment cost, reward from traffic carried through and penalty on traffic lost. The constraints are often QoS requirements for different traffic classes, whose calculation is determined by the performance evaluation model that may count for multi-rate traffic, adaptive routing or call admission control policy. Generally the performance evaluation is done iteratively to guide the search algorithm and check the feasibility of QoS constraints.

Works on network dimensioning for dynamic routing networks for the single-service case include [3, 5, 32, 33, 14, 34]. Multi-service networks have been addresses in [35, 36, 37, 38].

Due to the iterative nature of most performance models themselves, many of such works try to derive a closed form presentation of the shadow price or implied cost, often involving gradient calculation, to make the optimization model easier to program and compute numerically [36, 37, 4, 7]. Such derivations may vary depending on the performance model used by the optimization. In many cases obtaining the exact form of gradients is not possible and further approximation is required. In general manually deriving gradients makes the design process highly dependent on the underlying performance model.

In [39, 40] Medhi proposed a network dimensioning model that is decomposed into two steps: a bandwidth estimation problem and a multi-commodity flow model or a routing and capacity design problem.

1.3 Motivation

Our emphasis is on integrated services data networks employing QoS routing. While research results are abundant for circuit-switched scenarios, there has not been much study that applies to data networks. There has been extensive research in technologies of design and optimization of traditional circuit switched networks, e.g., [3, 4, 5, 6], and plenty of research results for fully connected, symmetric networks with fixed, sequential or state-dependent routing [7], especially for networks with no more than two hops on their routes [15], or when network traffic is of single rate [9]. There has been relatively lesser study that considers large random networks with both multiple traffic rates and state-dependent routing [7, 25, 41, 38]. Furthermore, all of such methods are for flat networks and flat routing schemes.

We observe that the evolving integrated service networks typically support traffic with varying bandwidth requirements. They are typically much sparser and have a more hierarchical topology. Correspondingly, routing schemes are becoming increasingly hierarchical in order to scale up with the size of the network. Routes can comprise of a much larger number of hops and there are typically a large number of possible routes between source and destination nodes. Therefore it is important to develop network performance models that take into account the random topology, different bandwidth requirements and hierarchical routing schemes.

There is also a lack of comprehensive tools and systematic methods to use these performance models for network design, planning, control and management.

This has constituted the main motivation for this study. Here we summarize our contributions and their significance.

- *Fast/Inexpensive end-to-end performance model.* One major source of complexity with the modeling of random networks with multiple alternative routes and a dynamic routing scheme is the overlaps between different routes. Under our assumption of probabilistic traffic distribution, we bypassed this problem and obtained a fast algorithm. Our model is a reduced load approximation, where the fixed point is obtained by iteration between four sets of unknown parameters. Our model works for random networks with the Least-Loaded Routing policy, with or without trunk reservation type of admission control and multiple traffic classes. Our model gives good conservative estimates for networks under heavy traffic, and it performs better as networks become larger and more random.

- *Models for hierarchical networks and/or hierarchical routing schemes.* To the best of our knowledge, so far all reduced load models are for flat networks with flat routing schemes. As networks increase in size, nodes tend to form clusters geographically, and hierarchical routing schemes are more commonly used to be able to scale with the network size. This has naturally led us to consider a hierarchical version of the reduced load model. We have developed hierarchical models for both fixed hierarchical routing and dynamic hierarchical routing, via the notion of network abstraction, route segmentation, traffic segregation and aggregation. Computation is done separately within each cluster (local) and among clusters (global), and fixed point is obtained by iteration between local and global computation. This work is the first in its area, and the goal is not to artificially introduce hierarchy into a network model, but to develop a more efficient and scalable way of performance analysis for networks that have a natural hierarchy and/or when some form of hierarchical routing is used. Our numerical results showed significant reduction in computational cost.

- *Models for networks using delay-based QoS routing.* End-to-end delay is often difficult to analyze in a data network. We showed how reduced load models can be used for estimating connection blocking in a network that employs delay-based QoS routing, and thus be used to estimate how efficient the routing scheme is in a given network scenario. The delay requirement is first mapped into rate requirement, and then we use *reclassification* and *quantization* to construct the model.

- Using Automatic Differentiation for Sensitivity Analysis:

  The technique of Automatic Differentiation (AD) lies in the fact that differentiation of functions defined by formulas is a mechanical process done according to fixed rules, e.g., the chain rule. Therefore it is highly suitable for automation along the same lines as function evaluation. We believe that it is a very novel approach to use AD for sensitivity analysis of our reduced load approximation. Since the approximation is an iterative process, analytical sensitivity analysis usually requires further approximation and the process of obtaining the derivatives is highly dependent on the approximation model. Using AD for computing derivatives of the fixed point along with the calculation of the fixed point itself is both accurate and efficient. We examined the convergence of derivatives of fixed point problems especially of the type that we investigate in this study. We also derived sufficient conditions for derivative convergence in this type of model.

- *Multi-Objective Design.* Network design and dimensioning often involve trade-offs between different objectives that sometimes may have opposing effects on design. At the same time, some objectives are mandatory while others are optional. We used the multi-objective optimization package CONSOL-OPTCAD for the design of trunk reservation parameters. We also studied link capacity design by using CFSQP along with our model and its sensitivities generated by AD. Putting together these techniques presents a systematic way of network design, regardless of the details of the underlying performance evaluation model.

1.4 Organization

In the chapters that follow, we will be describing our work in performance modeling and network design. In Chapter 2 we present a model estimating connection blocking for multi-class random topology networks using least-loaded routing. In Chapter 3 we develop a hierarchical version of the loss network model for networks that either have a natural hierarchy and/or some form of hierarchical routing is used, to achieve faster approximation. In Chapter 4 we formulate a model to evaluate delay-based QoS routing. We then link our performance models with numerical techniques for network design and dimensioning. We present our work using Automatic Differentiation and multi-objective design in Chapters 5 and 6, respectively.

Chapter 2

BLOCKING PROBABILITY IN A RANDOM TOPOLOGY NETWORK, WITH BANDWIDTH-OPTIMIZATION ROUTING

2.1 Introduction

In this chapter we investigate connection blocking in a random topology network that carries multiple classes of traffic, each with different QoS requirement, which in this case is the required bandwidth to be reserved along the communication path. The routing scheme we consider is of the bandwidth-optimization type, or least- loaded routing.

As we have mentioned in the Introduction, Most of the earlier works on the fixed-point method, either studied the multirate traffic situation with fixed routing, such as the Knapsack approximation and Pascal approximation in [24], or focused on state-dependent routing schemes with single traffic rate [9], or multirate service with single link (resource) [18]. In [25], Greenberg and Srikant proposed a fixed-point method to approximate blocking probabilities in a multirate multihop network using sequential routing, but additional computational effort in solving the associated network reliability problem is needed. It is also a common assumption that there exist a direct or two-hop routes between sources and destinations.

Our goal is to develop a model that takes into consideration the network topology, service types and routing policy that are more realistic to the emerging integrated services data network, and use the model to generate fast approximation to connection blocking and hence throughput. Part of our work is motivated by [25] and [27]. One major difficulty in modeling adaptive routing arises from the fact that when there are multiple feasible routes between two nodes, there are possible overlapping links among these routes, and as we will see from subsequent discussion, this constitutes a source of heavy computation. We present our approximation, as well as the exact computation to illustrate the fact.

The organization of the paper is as follows: The next section describes the network model and the adaptive routing schemes proposed for the approximation. Section 3 is the proposed fixed-point approximation method. In Section 4 we give details on one part of the fixed-point approximation and show how we achieved fast approximation. Asymptotic analysis is given in Section 5. In Section 6 we present approximation results compared to simulation results. Section 7 concludes this chapter.

2.2 Network Model

2.2.1 Notation

Consider a network with N nodes and J links, each indexed by j. C_j denotes the capacity of link j, in unit bandwidth/circuit. R is the set of all node pairs, each indexed by r. The total number of node pairs is thus $N(N-1)/2$. For each node pair r, there is an associated set of M ordered routes, each indexed by m, representing the m^{th} route in that set. Therefore, the pair (r, m) uniquely defines a specific route. The network supports a total of S classes of traffic, indexed by s, and thus (r, s) uniquely defines a specific incoming call request. The bandwidth requirement of such a call on link j is denoted by b_{js}. Note that for different node pairs the classification of calls does not have to be the same. So strictly speaking calls (r_1, s) and (r_2, s) can have different bandwidth requirement on a same link if we allow r_1 and r_2 to have different sets of traffic classes. However, we choose to use this notation because a single uniform classification can always be achieved by increasing the number of classes.

Calls arrive at the network as a Poisson process with an offered load λ_{rs}. A call is accepted if some route has available bandwidth on each of its links to accommodate this call, and the call is routed on that route and holds the bandwidth for a duration with mean time μ_{rs}. If none of the routes are available, the call is rejected. The end-to-end blocking probability of a call (r, s) is denoted by B_{rs}. Throughout this thesis the links are considered to be duplex and bi-directional.

We use trunks, units of circuits and units of bandwidth interchangeably.

Trunk reservation is considered in our model. An attempting alternatively routed call is only accepted if on each link of the alternative route the number of occupied circuits is less than $C_j - b_{js} - r_s$, where r_s is the *trunk reservation parameter* and may vary with link and class.

2.2.2 Routing Scheme

The common routing policies which have been studied are fixed routing, alternative routing, sequential alternative routing and adaptive alternative routing. We focus on the last. One important scheme of this kind is called the *Least Loaded Routing* (LLR), where the call is first tried on the direct route, if there is one. If it cannot be setup along the direct route, the two-link alternative route with the largest number of point-to-point free circuits is chosen. A version of LLR was implemented in the AT&T long-distance domestic network [7].

A direct extension of LLR to networks where routes tend to have a larger number of hops instead of direct or two-hop routes is a min-max scheme: Pick the link which has the minimum free bandwidth for each route, then pick the route which has the maximum free bandwidth on this link. Incidentally, this is also the bandwidth-optimization routing proposed for QoS routing in data networks.

Each source-destination node pair r is given a list of alternative routes M_r. When a call arrives, each of the route on the list is evaluated to determine the number of free circuits on its links.

Let C_j^f denote the free/available bandwidth on link j when the call of type (r,s) arrives and consider a route (r,m). Then the route is in a state of admitting this call if and only if

$$C_j^f \geq b_{js}, \text{for all} j \in (r,m)$$

under no trunk reservation admission control, or

$$C_j^f \geq b_{js} + r_s, \text{for all} j \in (r,m)$$

under trunk reservation admission control.

Consider a route (r,m) which is presently available for a type (r,s) call, the *most congested link on the route* is defined as the link with the fewest free circuits on this route:

$$\mathcal{L}_{rm} = argmin_{j \in (r,m)} C_j^f. \tag{2.1}$$

When there is more than one route available in the alternative route set, the one with the maximum free bandwidth on its most congested link is selected for accepting the call. If none of the routes are admissible, then the call is blocked.

This maximal residual capacity routing scheme tries to avoid bottlenecks on a route. However, while choosing the route which has the most free bandwidth, we might end up taking the longer or the longest routes in the available set and thus using more network resources. This could eventually force calls arriving later to be routed on their longer/longest route as well. Therefore, using trunk reservation along with this routing scheme is a valid choice especially when traffic is heavy.

A more general way of deciding the routing can be expressed as a cost function which takes into account both the length of the route and the congestion level of the route. (For optimization on routing and blocking, Mitra and Morrison present an elaborate form of network revenue in [36] and investigate network optimization problems. Our focus here is limited to admission control.) A simple cost function for route (r, m) could be:

$$w_1 \sum_{j \in (r,m)} b_{js} - w_2 C^f_{\mathcal{L}_{rm}}, \tag{2.2}$$

where the first term is the total number of bandwidth that would be occupied if the route is chosen, and the second term indicates the level of congestion on the route. w_1 and w_2 are weighting parameters. The route which minimizes this cost is chosen. Clearly a longer route will increase the cost. And if w_1 is zero, this becomes the min-max routing we just described.

2.3 The Fixed-point Method

The fixed point is achieved by mappings between the following four sets of unknown variables:

ν_{js}: the reduced load/arrival rate of class-s calls on link j;

a_{js}: the probability that link j is in a state of admitting class-s calls;

$p_j(n)$: the steady state occupancy probability distribution of link j, i.e., the probability that exactly n units of circuits are being used on link j. n takes any integer value between 0 and C_j, the capacity of link j;

q_{rms}: the probability that a call request (r, s) is attempted on route (r, m). This originates from the fact that different routes have different levels of congestion.

First, we fix a_{js} and q_{rms} to get ν_{js}. Then we let ν_{js} be fixed to get $p_j(n)$ and a_{js}. Finally we fix $p_j(n)$ to get q_{rms}. By repeated substitution, the equilibrium fixed point can be solved for all four sets of unknowns. The mappings are illustrated in the figure bellow:

$$\begin{tikzcd}
a_{js} \arrow[dr] \arrow[drr, shift left] & & \\
& \nu_{js} \arrow[r] & p_j(n) \arrow[ull, shift left] \arrow[dll] \\
q_{rms} \arrow[ur] & &
\end{tikzcd}$$

Figure 2.1: Mappings between variables

2.3.1 Mapping 1: a_{js}, $q_{rms} \longrightarrow \nu_{js}$

Define ν_{jrms} as the arrival rate on link j contributed by traffic (r, s) on route (r, m), given that link j is in a state of admitting a class-s call. Then it is given by the reduced load approximation as:

$$\nu_{jrms} = \lambda_{rs} q_{rms} I[j \in (r, m)] \prod_{i \in (r,m), i \neq j} a_{is} \tag{2.3}$$

where I is the indicator function. The aggregated load of class-s calls on link j is given by

$$\nu_{js} = \sum_{(r,m)} \nu_{jrms}. \tag{2.4}$$

2.3.2 Mapping 2: $\nu_{js} \longrightarrow a_{js},\ p_j(n)$

Given ν_{js}, we can compute the link occupancy probabilities $p_j(n)$ for each link in the network. This can be done by either using Kaufman's simple recursion [11] when there is no trunk reservation present, or using approaches proposed by Bean, Gibbens and Zachary in [17] and [18] as suggested by Greenberg in [25]. By the link independence assumption, this mapping is conducted on a per-link basis, and each link is calculated separately and similarly.

In the absence of admission control, classical product-form holds (equation 1.2) for describing the equilibrium call blocking probabilities. In [11], Kaufman gives a simple one dimensional recursion for calculating the link occupancy distribution probabilities. Let n_s denote the number of class-s calls in progress on this link and b_{js} is the bandwidth requirement of class-s calls on link j. Let n denote the total units of bandwidth occupied on link j. Then for link j we have

$$np_j(n) = \sum_s b_{js} \frac{\nu_{js}}{\mu_{rs}} p_j(n - b_{js}), \quad n = 1, ..., C_j, \tag{2.5}$$

where $p_j(n) = 0$ if $n < 0$ and

$$\sum_{n=0}^{C_j} p_j(n) = 1. \tag{2.6}$$

Also it's easy to see that $n = \sum_s b_{js} n_s$.

The probability that a class-s call is admitted to link j is given by

$$a_{js} = 1 - \sum_{n=C_j - b_{js}+1}^{C_j} p_j(n) = \sum_{n=0}^{C_j - b_{js}} p_j(n). \tag{2.7}$$

Admission control destroys the product form of the link occupancy probabilities $p_j(n)$, which in turn destroys the efficient exact computation of those probabilities just described. To solve for these probabilities, we need to solve for the equilibrium distribution of the associated Markov chain, whose state space is a lattice embedded in the simplex $\sum_s b_{js} n_s \leq C_j,\ n_s \geq 0$. The computational cost is prohibitive, even for moderate C_j and $S = 2$. A method to compute the aggregated occupancy probabilities $p(n)$ at a cost linear in C_j is needed. Approaches proposed in [17, 18] transform the problem into a one-dimensional one by assuming that while n_s, the number of calls in progress of a class s varies, the proportion of such calls in progress remains fixed (or varies slowly).

In [25], the following method is used. Let α_{js} denote the average number of calls of type s in progress on link j,

$$\alpha_{js} = a_{js} \nu_{js} / \mu_{rs}, \tag{2.8}$$

since calls enter into service at rate $a_{js}\nu_{js}$ and depart at rate $\alpha_{js}\mu_{rs}$.

Consider the one-dimensional Markov chain, for any given state n and call class s, with the following state transition rates:

From state n to state $n + b_{js}$, $\nu_{js} I(C_j - n \geq r_s + b_{js})$;

From state n to state $n - b_{js}$, $\mu_{rs} n \frac{\alpha_{js}}{\sum_t \alpha_{jt}} I(n \geq b_{js})$.

The probability of admitting a call of class s is given by

$$a_{js} = 1 - \sum_{n=C_j - b_{js} - r_s + 1}^{C_j} p_j(n) = \sum_{n=0}^{C_j - b_{js} - r_s} p_j(n). \tag{2.9}$$

Note that $p_j(n) \longrightarrow \alpha_{js} \longrightarrow p_j(n)$ forms another fixed-point problem, which can be solved by iteration to get the equilibrium distribution $p_j(n)$ and thus a_{js}.

If we use the cost function presented in the previous section to make routing decisions, a trunk reservation scheme will no longer be necessary since the idea of trunk reservation admission control is to prevent routing calls onto those longer routes and the cost function has already taken the length of the route into consideration.

2.3.3 Mapping 3: $p_j(n) \longrightarrow q_{rms}$

Given $p_j(n)$, define for link j, the probability of no more than n trunks are free (at most n trunks are free) as:

$$t_j(n) = \sum_{k=0}^{n} p_j(C_j - k). \tag{2.10}$$

Consider the case of no trunk reservation admission control, and use the min-max routing scheme extended from LLR described in the previous section. The probability of attempting a call of (r,s) on route (r,m) is the probability that all routes before the m^{th} route on the routing list have fewer free trunks on their most congested links, and all routes after the m^{th} route have at most the same number of free trunks on their most congested links, which can be expressed as:

$$q_{rms} = \sum_{n=1}^{C_{\mathcal{L}rm}} p_{\mathcal{L}_{rm}}(C_{\mathcal{L}_{rm}} - n) \prod_{k=1}^{m-1} t_{\mathcal{L}_{rk}}(n-1) \prod_{k=m+1}^{M} t_{\mathcal{L}_{rk}}(n). \tag{2.11}$$

We consider steady state and the free bandwidth on link j, C_j^f is replaced by $E[C_j^f]$, the expected average free capacity. We proceed to derive this value.

Lemma 2.1 *The expected number of busy trunks on a link, denoted by $E[n]$, is given by the following*

$$E[n] = \sum_{s=1}^{S} \frac{\nu_s b_s}{\mu_s} \cdot a_s.$$

Proof. Using Kaufman's recursion

$$\sum_{s=1}^{S} \frac{\nu_s b_s}{\mu_s} p(n - b_s) = n \cdot p(n),$$

Sum over 0 through C on both sides:

$$\begin{aligned}
\text{LHS} &= \sum_{n=0}^{C} \sum_{s=1}^{S} \frac{\nu_s b_s}{\mu_s} p(n - b_s) \\
&= \sum_{s=1}^{S} \frac{\nu_s b_s}{\mu_s} \sum_{n=b_s}^{S} p(n - b_s) \\
&= \sum_{s=1}^{S} \frac{\nu_s b_s}{\mu_s} \cdot a_s \\
\text{RHS} &= \sum_{n=0}^{C} np(n) \\
&= E[n]
\end{aligned}$$

Hence,

$$E[n] = \sum_{s=1}^{S} \frac{\nu_s b_s}{\mu_s} \cdot a_s.$$

Q.E.D.

Therefore for link j, denoting by n_j the number of busy circuits,

$$\begin{aligned}
E[C_j^f] &= C_j - E[n_j] \\
&= C_j - \sum_{s=1}^{S} \frac{\nu_{js} b_s}{\mu_s} \cdot a_s,
\end{aligned}$$

and consequently the most congested link on route (r, m) becomes the stochastically most congested link:

$$\mathcal{L}_{rm} = argmin_{j \in (r,m)} (C_j - \sum_{s=1}^{S} \frac{\nu_{js} b_s}{\mu_s} \cdot a_s).$$

The probability q_{rms} is obviously an approximation to the real value that could be computed from the complete stationary distribution $p(n)$ of all links, since we ignored dependencies between routes due to sharing the same links. In the next section we show the exact computation of this probability and show why approximation is necessary.

When there is admission control with trunk reservation parameter r_s, the probability of choosing a route becomes:

$$\begin{aligned} q_{rms} &= \sum_{n=1}^{C_{\mathcal{L}rm}} p_{\mathcal{L}_{rm}}(C_{\mathcal{L}_{rm}} - n) t_{\mathcal{L}_{r1}}(n-1) \prod_{k=2}^{m-1} t_{\mathcal{L}_{rk}}(n + r_s - 1) \cdot \\ & \prod_{k=m+1}^{M} t_{\mathcal{L}_{rk}}(n + r_s) \end{aligned} \tag{2.12}$$

assuming that we do not impose trunk reservation on the first route in a set, since naturally that would be the shorted one among all even if it is not the direct route.

If we use the cost function proposed in the previous section, then since the choice of the m^{th} route for routing the call indicates that all the routes before m^{th} route have a higher cost than the m^{th} route, and all the routes after the m^{th} route have at least the same cost, the probability of attempting the call on the m^{th} route can be expressed as:

$$q_{rms} = \sum_{n=1}^{C_{\mathcal{L}rm}} p_{\mathcal{L}_{rm}}(C_{\mathcal{L}_{rm}} - n) \prod_{k=1}^{m-1} t_{\mathcal{L}_{rk}}(\frac{w_1}{w_2} \sum_{j \in (r,k)} b_{js} - \frac{w_1}{w_2} \sum_{j \in (r,m)} b_{js} + n - 1) \cdot$$

$$\prod_{k=m+1}^{M} t_{\mathcal{L}_{rk}}(\frac{w_1}{w_2}\sum_{j\in(r,k)} b_{js} - \frac{w_1}{w_2}\sum_{j\in(r,m)} b_{js} + n). \tag{2.13}$$

2.3.4 End-to-end blocking probabilities

Finally, the end-to-end blocking probability for calls of class s between source-destination node pair r is given by

$$B_{rs} = 1 - \sum_m q_{rms} \prod_{j\in(r,m)} a_{js}. \tag{2.14}$$

Repeated substitution is used to obtain the equilibrium fixed point. And the end-to-end blocking probabilities can be calculated from the fixed point.

2.4 Computation of Route Probability q

In this section we proceed to derive the exact expression for the probability of choosing a particular route q_{rms}, and show how overlapping links among routes makes this calculation complex.

Define a *route set* $\mathcal{R}$ as a set of links that constitute one or more routes, or constitutes the common links between two or more routes. For example, each of the following operations defined on routes result in a route set. Here to simplify notation, rewrite (r,m), the m^{th} route for node pair r as r_m, and let

$$\begin{aligned} r_m \times r_n &= r_m \cap r_n = \{j : j \in r_m \ \&\ j \in r_n\}; \\ r_m - r_n &= r_m \cap \bar{r}_n = \{j : j \in r_m \ \&\ j \notin r_n\} \\ r_m + r_n &= r_m \cup r_n = \{j : j \in r_m \text{ or } j \in r_n\}. \end{aligned}$$

Define $A_n(\mathcal{R})$ as the event that all links in the route set $\mathcal{R}$ have at least n free trunks ($\geq n$), and $\bar{A}_n(\mathcal{R})$ is the event that at least one link in $\mathcal{R}$ has less than n free trunks ($< n$). Thus

$$\bar{A}_n(\mathcal{R}) = 1 - A_n(\mathcal{R}).$$

Define $\tilde{A}_n(\mathcal{R})$ as the event that all links in $\mathcal{R}$ have at least n free trunks and at least one link in $\mathcal{R}$ has exactly n free trunks.

Denote the probability of $A_n(\mathcal{R})$ as $g_n(\mathcal{R})$, the probability of $\bar{A}_n(\mathcal{R})$ as $\bar{g}_n(\mathcal{R})$, and the probability of $\tilde{A}_n(\mathcal{R})$ as $\tilde{g_n}(\mathcal{R})$, and we have

$$\begin{aligned} g_n(\mathcal{R}) &= \Pr[A_n(\mathcal{R})] = \prod_{j\in\mathcal{R}} \sum_{k=1}^{C_j-n} p_j(k); & (2.15)\\ \bar{g}_n(\mathcal{R}) &= \Pr[\bar{A}_n(\mathcal{R})] = 1 - g_n(\mathcal{R}); & (2.16)\\ \tilde{g_n}(\mathcal{R}) &= \Pr[\tilde{A}_n(\mathcal{R})] = \Pr[A_n(\mathcal{R})] - \Pr[A_{n+1}(\mathcal{R})] \\ &= g_n(\mathcal{R}) - g_{n+1}(\mathcal{R}) \\ &= \prod_{j\in\mathcal{R}} \sum_{k=1}^{C_j-n} p_j(k) - \prod_{j\in\mathcal{R}} \sum_{k=1}^{C_j-n-1} p_j(k). & (2.17) \end{aligned}$$

Denote $\min C_{r_m}$ as the minimum link capacity along route r_m. By definition,

$q_{rms} = \Pr$[route (r,m) is chosen for class s|route (r,m) is admissible for class s].

Therefore we have

$$\begin{aligned} q_{rms} &= \sum_{n=0}^{\min_{C_{r_m}}} \Pr[\bar{A}_n(r_1) \cap ... \cap \bar{A}_n(r_{m-1}) \cap \bar{A}_{n+1}(r_{m+1}) \cap ... \cap \bar{A}_{n+1}(r_M)| \\ &\quad \tilde{A}_n(r_m), A_{b_s}(r_m)] \cdot \Pr[\tilde{A}_n(r_m)|A_{b_s}(r_m)] \\ &= \sum_{n=b_s}^{\min_{C_{r_m}}} \Pr[\bar{A}_n(r_1) \cap ... \cap \bar{A}_n(r_{m-1}) \cap \bar{A}_{n+1}(r_{m+1}) \cap ... \cap \bar{A}_{n+1}(r_M)| \\ &\quad \tilde{A}_n(r_m)] \cdot \Pr[\tilde{A}_n(r_m)|A_{b_s}(r_m)] \qquad (2.18) \end{aligned}$$

The second term on the RHS of (2.18), for $n \geq b_s$:

$$\begin{aligned}
\Pr[\tilde{A}_n(r_m) \mid A_{bs}(r_m)] &= \frac{\Pr[\tilde{A}_n(r_m), A_{bs}(r_m)]}{\Pr[A_{bs}(r_m)]} \\
&= \frac{\Pr[\tilde{A}_n(r_m)]}{\Pr[A_{bs}(r_m)]} \\
&= \frac{g_n(r_m) - g_{n+1}(r_m)}{g_{bs}(r_m)}. \qquad (2.19)
\end{aligned}$$

To calculate the first term on the RHS of (2.18), consider the two simplest cases, with $M = 2, m = 2$ and $M = 3, m = 3$, respectively.

$$\Pr[\bar{A}_n(r_1) \mid \tilde{A}_n(r_2)] = \Pr[\bar{A}_n(r_1 - r_2)] = 1 - g_n(r_1 - r_2). \qquad (2.20)$$

$$\begin{aligned}
&\Pr[\bar{A}_n(r_1) \cap \bar{A}_n(r_2) \mid \tilde{A}_n(r_3)] = \Pr[\bar{A}_n(r_1 - r_3) \cap \bar{A}_n(r_2 - r_3)] \\
= \;&\Pr[\bar{A}_n(r_1 \times r_2 - r_m) \cap A_n(r_1 - r_2 - r_m) \cap A_n(r_2 - r_1 - r_m)] + \\
&\Pr[\bar{A}_n(r_1 \times r_2 - r_m) \cap \bar{A}_n(r_1 - r_2 - r_m) \cap A_n(r_2 - r_1 - r_m)] + \\
&\Pr[\bar{A}_n(r_1 \times r_2 - r_m) \cap A_n(r_1 - r_2 - r_m) \cap \bar{A}_n(r_2 - r_1 - r_m)] + \\
&\Pr[\bar{A}_n(r_1 - r_2 - r_m) \cap \bar{A}_n(r_2 - r_1 - r_m)]. \qquad (2.21)
\end{aligned}$$

As the number of allowed routes (M) increases, this term requires on the order of 2^{M-1} operations, and there is no easy way of computing it. It can be much simplified if there is no overlapping among routes $r_1, r_2, ..., r_M$, which would give us:

$$\begin{aligned}
&\Pr[\bar{A}_n(r_1) \cap ... \cap \bar{A}_n(r_{m-1}) \cap \bar{A}_{n+1}(r_{m+1}) \cap ... \cap \bar{A}_{n+1}(r_M) \mid \tilde{A}_n(r_m)] \\
= \;&\Pr[\bar{A}_n(r_1)] \cdots \Pr[\bar{A}_n(r_{m-1})] \cdot \Pr[\bar{A}_{n+1}(r_{m+1})] \cdots \Pr[\bar{A}_{n+1}(r_M)] \\
= \;&\bar{g}_n(r_1) \cdots \bar{g}_n(r_{m-1}) \cdot \bar{g}_{n+1}(r_{m+1}) \cdots \bar{g}_{n+1}(r_M)
\end{aligned}$$

$$= \prod_{i=1}^{m-1} (1 - g_n(r_i)) \prod_{i=m+1}^{M} (1 - g_{n+1}(r_i)) \tag{2.22}$$

Unfortunately overlapping is often the case, which makes the calculation much more complicated. One possible way to approximate this value seems to be to acknowledge the dependence between r_m and other routes but ignore overlapping among all routes other than r_m, which then gives us:

$$\begin{aligned} q_{rms} &= \sum_{n=b_s}^{\min C_{r_m}} \Pr[\bar{A}_n(r_1) \cap ... \cap \bar{A}_n(r_{m-1}) \cap \bar{A}_{n+1}(r_{m+1}) \cap ... \cap \bar{A}_{n+1}(r_M) \mid \tilde{A}_n(r_m)] \\ &\quad \cdot \Pr[\tilde{A}_n(r_m) | A_{b_s}(r_m)] \\ &\approx \sum_{n=b_s}^{\min C_{r_m}} \Pr[\bar{A}_n(r_1 - r_m)] \cdots \Pr[\bar{A}_n(r_{m-1} - r_m)] \cdot \Pr[\bar{A}_{n+1}(r_{m+1} - r_m)] \cdots \\ &\quad \Pr[\bar{A}_{n+1}(r_M - r_m)] \cdot \frac{\Pr[\tilde{A}n(r_m)]}{\Pr[A_{b_s}(r_m)]} \\ &= \sum_{n=b_s}^{\min C_{r_m}} \prod_{i=1}^{m-1} \bar{g}_n(r_i - r_m) \prod_{i=m+1}^{M} \bar{g}_{n+1}(r_i - r_m) \cdot \frac{g_n(r_m) - g_{n+1}(r_m)}{g_{b_s}(r_m)}. \end{aligned} \tag{2.23}$$

This still requires on the oder of M operations of $g_n(\cdot)$. The approximation we introduced in the previous section (2.11) further reduces computation by considering only the most congested link (on average) along a route.

2.5 Asymptotic Analysis

By using Brouwer's fixed point theorem, it's easy to show that there exists a fixed point under the proposed fixed point approximation. In this section, we analyze the asymptotic property of our fixed point approximation. First we give a steady state explanation for q_{rms}, and formulate an optimization problem to solve for the most probable state for a single link. Then we establish a limiting regime and show that under the specified limiting regime, the blocking probability converges to our fixed point approximation.

Under steady state, for traffic stream (r, s), the probability of attempting the call on the m^{th} route is q_{rms}. In reality, when a call request comes, using an adaptive routing scheme, the call is routed according to the actual traffic load in the network at that point in time. This type of traffic dispersement is called "metering" [42]. However, in our approximation the routing is modeled as if each traffic stream has fixed probabilities to be routed onto a set of routes, and those probabilities add up to 1. This method is called "randomization". The metering method generally gives a better performance over randomization. Therefore, our approximation represents a conservative estimate.

Accepting this assumption, since randomly splitting a Poisson process according to a fixed probability distribution results in processes which are individually Poisson, we have $\nu_{rms} = \lambda_{rs} \cdot q_{rms}$; the equivalent offered load onto route (r, m) from traffic (r, s), which is still a Poisson process.

Define vector $\mathbf{n} = \{n_{rms}\}$, where n_{rms} is the number of calls in progress on route (r, m) from traffic stream (r, s). For clearer notation purposes, let b_{jrms} be the bandwidth requirement on link j from call (r, s) on route (r, m), and define matrix $\mathbf{b} = \{b_{jrms}\}$. Also define vector $\mathbf{C} = \{C_j\}$ to be the link capacity. For a single link the stationary distribution $\pi(\mathbf{n})$ is given by:

$$\pi(\mathbf{n}) = G(\mathbf{n})^{-1} \prod_{(r,m)} \prod_{s} \frac{\nu_{rms}^{n_{rms}}}{n_{rms}!}, \quad \mathbf{n} \in A(\mathbf{C}), \tag{2.24}$$

where

$$A(\mathbf{C}) = \{\mathbf{n} > \mathbf{0} : \mathbf{b} \cdot \mathbf{n} \leq \mathbf{C}\} \tag{2.25}$$

defines the set for all feasible $\mathbf{n}$ under the link capacity constraint, and $G(\mathbf{n})$ is the normalizing factor:

$$G(\mathbf{n}) = \sum_{\mathbf{n}} (\prod_{(r,m)} \prod_{s} \frac{\nu_{rms}^{n_{rms}}}{n_{rms}!}). \tag{2.26}$$

Following Kelly's method in [7], we form the optimization problem of maximizing $\pi(\mathbf{n})$ to find the most probable state $\mathbf{n}$:

$$\begin{aligned} &\text{Max} \sum_{(r,m)} \sum_{s} (n_{rms} log \nu_{rms} - log \nu_{rms}!) \\ &\text{S.t. } \mathbf{n} \geq 0,\ \mathbf{b} \cdot \mathbf{n} \leq \mathbf{C}. \end{aligned} \tag{2.27}$$

Using Sterling's formula $\log n! \approx n \log n - n$, and replacing $\mathbf{n}$ by real vector $\mathbf{x} = \{x_{rms}\}$, the primal problem becomes:

$$\begin{aligned} &\text{Max} \sum_{(r,m)} \sum_{s} (x_{rms} \log \nu_{rms} - x_{rms} \log x_{rms} + x_{rms}) \\ &\text{S.t. } \mathbf{x} \geq 0,\ \mathbf{b} \cdot \mathbf{x} \leq \mathbf{C}. \end{aligned} \tag{2.28}$$

The objective function is differentiable and strictly concave over $x_{rms} \geq 0$; the feasible region is a closed, bounded convex set. Therefore there exists a unique maximum. Using Lagrangian method, the maximum can be found to be:

$$x_{rms}(\mathbf{y}) = \nu_{rms} \cdot \exp(-\sum_{j} y_j b_{jrms}). \tag{2.29}$$

where $\mathbf{y} = \{y_j\}$ is the Lagrangian multiplier. The constraints become

$$\mathbf{x}(\mathbf{y}) \geq 0, \ \mathbf{C} - \mathbf{b} \cdot \mathbf{x}(\mathbf{y}) \geq 0. \tag{2.30}$$

By introducing transformed variable

$$d_j = 1 - exp(-y_j) \tag{2.31}$$

we can rewrite the maximum in (2.29) as

$$x_{rms} = \nu_{rms} \cdot \prod_{j \in (r,m)} (1 - d_j)^{b_{jrms}}, \tag{2.32}$$

and d_j is any solution to the following:

$$\sum_{(r,m)} \sum_{s} b_{jrms} \nu_{rms} \cdot \prod_{i} (1 - d_i)^{b_{irms}} \begin{cases} = C_j & \text{if} \quad d_j > 0 \\ \leq C_j & \text{if} \quad d_j = 0 \end{cases} \tag{2.33}$$

and $d_j \in [0, 1)$.

Using the limiting scheme due to Kelly [7], we consider a sequence of networks indexed by N with increasing link capacity and offered traffic load. In addition, we also allow the number of alternative routes for each source-destination node pair to increase with N. This results in the following limiting regime:

$$\begin{aligned} &\frac{\lambda_{rs}(N)}{N} \longrightarrow \lambda_{rs}, \quad \frac{C_j(N)}{N} \longrightarrow C_j \quad \text{as} \quad N \longrightarrow \infty \quad \text{and} \\ &\sum_{M} q_{rms} = 1 \quad M \longrightarrow \infty \end{aligned} \tag{2.34}$$

with λ_{rs}/C_j fixed, and M is the total number of alternative routes. Let

$$k_{jrms} = \frac{\nu_{rms}}{C_j} = \frac{\lambda_{rs} q_{rms}}{C_j} \tag{2.35}$$

also be fixed based on our assumption with q_{rms}.

Following from [7], the blocking probability $B_{rms}(N)$, which is the stationary probability that a call from source (r,s) is accepted by route (r,m) is given by:

$$1 - B_{rms}(N) = q_{rms} \prod_j (1-d_j)^{b_{jrms}} + o(1) \tag{2.36}$$

where d_j is the solution to (2.32).

Now consider the link occupancy probability distribution of link j given by the Kaufman recursion (this is a restatement of (2.7)):

$$np_j(n) = \sum_{rms} b_{jrms} \frac{\nu_{jrms}}{\mu_{rs}} p_j(n - b_{jrms}), \quad n = 0, 1, ..., C_j(N). \tag{2.37}$$

We observe that $d_j = p_j(C_j)$ forms a valid solution to (2.32), which is the probability that the link is fully occupied. Denote n' as the number of free circuits on link j (n is the number of circuits occupied), and p' as the distribution of n'. As $N \longrightarrow \infty$ and $C_j(N) \longrightarrow \infty$, the distribution $p'_j(n') = p_j(C_j(N) - n)$ converges weakly to the geometric distribution given by [8]:

$$p'_j(n') = (1-p)p^{n'} \tag{2.38}$$

where p is the positive root of

$$1 = \sum_{rms} \frac{b_{jrms}}{\mu_{rs}} k_{jrms} p^{b_{jrms}}. \tag{2.39}$$

Substitute $n' = 0$ into (2.38),

$$p = 1 - p'_j(0) = 1 - p_j(C_j(N)) = 1 - d_j. \tag{2.40}$$

So the probability that a call from source (r, s) attempted on route (r, m) can be admitted on link j is

$$
\begin{aligned}
a_{js} &= \sum_{n'=b_{jrms}}^{C_j(N)} p'_j(n') \\
&= (1-p)(p^{b_{jrms}} + p^{b_{jrms}+1} + \ldots + p^{C_j(N)}) \\
&= p^{b_{jrms}} - p^{C_j(N)+1} \\
&= {}^{C_j(N)\to\infty} p^{b_{jrms}} \\
&= (1-d_j)^{b_{jrms}}. \qquad (2.41)
\end{aligned}
$$

Hence the asymptotic form (2.36) can be written as

$$
\begin{aligned}
1 - B_{rms}(N) &= q_{rms} \prod_j (1-d_j)^{b_{jrms}} + o(1) \\
&= q_{rms} \prod_j a_{js} + o(1) \qquad (2.42)
\end{aligned}
$$

Therefore we get the approximation

$$
\begin{aligned}
B_{rs}(N) &= 1 - \sum_m (1 - B_{rms}(N)) \\
&= {}^{M,N\longrightarrow\infty} 1 - \sum_m q_{rms} \prod_j a_{js}, \qquad (2.43)
\end{aligned}
$$

which is the form we presented in (2.14) of the algorithm.

Similarly, from (2.32), load on route (r, m) from source (r, s) becomes

$$x_{rms} = \nu_{rms} \cdot \prod_{j\in(r,m)} a_{js} = q_{rms}\lambda_{rs} \prod_{j\in(r,m)} a_{js}. \qquad (2.44)$$

Therefore, as seen by any individual link i,

$$x_{irms} = q_{rms}\lambda_{rs} \prod_{j\in(r,m),j\neq i} a_{js}, \qquad (2.45)$$

which is our first mapping in the algorithm.

2.6 Experiments and Evaluation

In this section we give two network examples to compare the approximation results with that of simulation.

The first example is a five-node fully connected network depicted in Figure 2.2.

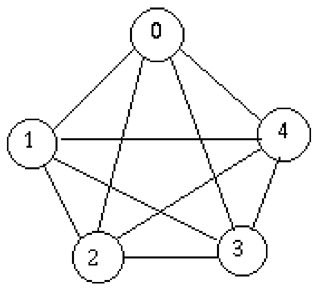

Figure 2.2: Topology of Example One.

Capacity for each link is set to be 100. There are three classes of connections, which are indexed 1, 2, and 3. They have bandwidth requirements of 1, 2, and 3, respectively. When call admission control is used under heavy traffic, the trunk reservation parameter for each class is 2, 4, and 6, respectively.

For each node pair, the direct route and all two-hop routes are allowed. The direct route is listed first in the routing list, and the two-hop routes are listed in random order.

The medium traffic rates are listed in Appendix A. Heavy traffic rates are set to be double the medium rates.

We only display three node pairs here for comparison between fixed-point approximation (FPA) and discrete event simulation (DES). All simulations were run to get a 95% confidence interval. The results are listed in Table 2.1 through Table 2.3, with Table 2.1 showing results for medium traffic, and Tables 2.2 and 2.3 for heavy traffic with and without trunk reservation, respectively.

Although the traffic in this case is highly asymmetric, since routing is "symmetric" in the sense that all node pairs are using one direct route and three two-hop routes, each traffic stream is imposing on the network in the same way. Therefore, we observe that connections of the same class, with the same bandwidth requirement, encounter approximately the same blocking probability regardless of their source-destination node pair and input rate. However, they do vary slightly from one to another reflecting the random order in which the two-hop routes are listed.

The second example is borrowed from [25] with minor changes. The topology is derived from an existing commercial network and is depicted in Figure 2 below.

There are 16 nodes and 31 links, with link capacity ranging from 60 to 180 trunks. The detailed link-by-link traffic statistics and link capacities are provided in in Appendix A. The traffic in the network consists of four types, namely class-1, 2, 3, and 4, and require bandwidth of 1, 2, 3, and 4 trunks, respectively. No admission control is employed in this experiment.

In routing, any node pair is allowed routes that have at most 4 hops. Multiple routes for one node pair are listed in order of increasing hops, with ties broken randomly. Each link is considered to have same unit length, so only the hop number is counted.

Node Pair	Class	FPA	DES
$(0,3)$	1	0.035432	(0.0341, 0.0348)
	2	0.298688	(0.2274. 0.2276)
	3	0.535481	(0.4730, 0.4735)
$(1,2)$	1	0.035938	(0.0358, 0.0360)
	2	0.299227	(0.2209, 0.2212)
	3	0.535798	(0.4701, 0.4710)
$(2,4)$	1	0.035406	(0.0336, 0.0340)
	2	0.296997	(0.2300, 0.2303)
	3	0.533397	(0.4673, 0.4677)
Number of Iterations		11	
CPU Time(seconds)		1.55	7.1×10^3

Table 2.1: Ex.1 with Medium Traffic

Node Pair	Class	FPA	DES
$(0,3)$	1	0.232178	(0.1921, 0.1923)
	2	0.672037	(0.6059, 0.6070)
	3	0.826792	(0.8340, 0.8342)
$(1,2)$	1	0.229546	(0.1941, 0.1942)
	2	0.668035	(0.6076, 0.6078)
	3	0.822406	(0.8257, 0.8261)
$(2,4)$	1	0.225318	(0.1849, 0.1851)
	2	0.663327	(0.6002, 0.6004)
	3	0.828391	(0.8290, 0.8291)
Number of Iterations		9	
CPU Time(seconds)		1.24	1.3×10^4

Table 2.2: Ex.1 with Heavy Traffic and no Trunk Reservation

Node Pair	Class	FPA	DES
$(0,3)$	1	0.006721	(0.0023, 0.0024)
	2	0.549455	(0.6117, 0.6120)
	3	0.963617	(0.9763, 0.9764)
$(1,2)$	1	0.005072	(0.0019, 0.0020)
	2	0.544652	(0.6041, 0.6043)
	3	0.955777	(0.9741, 0.9743)
$(2,4)$	1	0.009249	(0.0014, 0.0016)
	2	0.524917	(0.5765, 0.5766)
	3	0.945543	(0.9709, 0.9711)
Number of Iterations		12	
CPU Time(seconds)		10.86	1.3×10^4

Table 2.3: Ex.1 with Heavy Traffic and Trunk Reservation

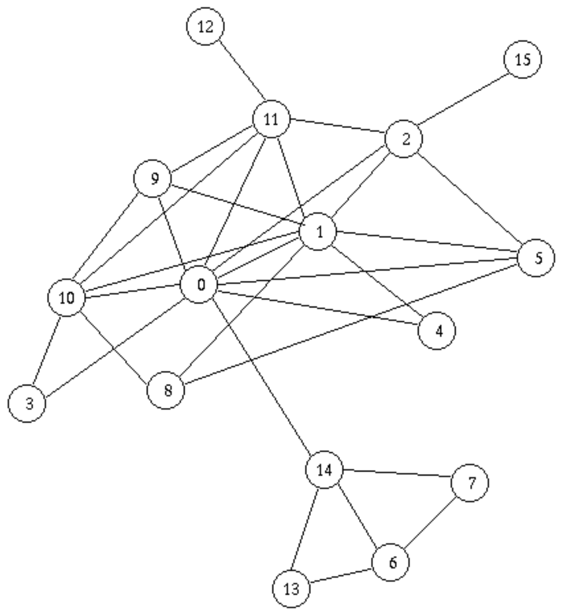

Figure 2.3: Topology of Example Network

Results for some selected node pairs and classes are listed in Table 2.4 through 2.8, each corresponding to a different traffic load. Table 2.4 corresponds to the "nominal" traffic which is provided in Appendix A. Tables 2.5 through 2.8 show the results for traffic 1.4, 1.6 and 1.8 times the nominal traffic, respectively.

The proposed fixed-point approximation gives conservative estimates generally, and it improves as the load gets heavier and as the network becomes more random. These results strengthen the argument that these approximations are indeed very useful as estimators of worst case performance.

Node Pair	Class	FPA	DES
$(0,4)$	4	0.000178	(0.0, 0.0)
$(0,13)$	1	0.003341	(0.0021, 0.0034)
$(1,6)$	1	0.003473	(0.0030, 0.0034)
$(5,6)$	3	0.020463	(0.0189, 0.0201)
$(6,10)$	2	0.013222	(0.0109, 0.0138)
$(9,13)$	4	0.028468	(0.0185, 0.0245)
Number of Iterations		18	
CPU Time(seconds)		94.1	3.7×10^4

Table 2.4: Ex.2 Nominal Traffic.

Node Pair	Class	FPA	DES
$(0,4)$	4	0.003234	(0.0, 0.0)
$(0,13)$	1	0.036512	(0.0351, 0.0369)
$(1,6)$	1	0.036999	(0.0303, 0.0311)
$(5,6)$	3	0.114667	(0.1103, 0.1137)
$(6,10)$	2	0.073531	(0.0543, 0.0573)
$(9,13)$	4	0.164185	(0.1213, 0.1268)
Number of Iterations		23	
CPU Time(seconds)		120.35	3.9×10^4

Table 2.5: Ex. 2 1.2 Times The Nominal Traffic.

Node Pair	Class	FPA	DES
$(0,4)$	4	0.018213	(0.0122, 0.0179)
$(0,13)$	1	0.074434	(0.0729, 0.0766)
$(1,6)$	1	0.077371	(0.0697, 0.0701)
$(5,6)$	3	0.229528	(0.2262, 0.2278)
$(6,10)$	2	0.147436	(0.1420, 0.1483)
$(9,13)$	4	0.307191	(0.2794, 0.2848)
Number of Iterations		28	
CPU Time(seconds)		145.43	4.3×10^4

Table 2.6: Ex. 2 1.4 Times The Nominal Traffic.

Node Pair	Class	FPA	DES
$(0,4)$	4	0.055354	(0.0512, 0.0549)
$(0,13)$	1	0.107588	(0.0987, 0.1012)
$(1,6)$	1	0.117211	(0.1113, 0.1121)
$(5,6)$	3	0.332202	(0.3137, 0.3142)
$(6,10)$	2	0.212533	(0.2164, 0.2210)
$(9,13)$	4	0.424501	(0.3380, 0.3465)
Number of Iterations		24	
CPU Time(seconds)		125.55	5.6×10^4

Table 2.7: Ex. 2 1.6 Times The Nominal Traffic.

Node Pair	Class	FPA	DES
$(0,4)$	4	0.112658	(0.0025, 0.0026)
$(0,13)$	1	0.155564	(0.1492, 0.1500)
$(1,6)$	1	0.146322	(0.1445, 0.1466)
$(5,6)$	3	0.399781	(0.3922, 0.3940)
$(6,10)$	2	0.269145	(0.2572, 0.2583)
$(9,13)$	4	0.519083	(0.4791, 0.4793)
Number of Iterations		24	
CPU Time(seconds)		125.11	2.3×10^6

Table 2.8: Ex. 2 1.8 Times The Nominal Traffic.

2.7 Conclusion

In this chapter we presented an approximation scheme of calculating the end-to-end, class-by-class blocking probability of a loss network with multirate traffic and adaptive routing scheme. It provides fairly good estimates of call blocking probabilities under normal and heavy traffic, orders of magnitude faster than discrete event simulation based estimation. We also presented asymptotic analysis of the fixed point algorithm and showed that this algorithm gives conservative estimates in general.

Chapter 3

HIERARCHICAL LOSS NETWORK MODELS

3.1 Introduction

In this chapter we present a hierarchical version of the loss network model for estimating connection blocking probabilities. The objective is not to artificially introduce hierarchy into a network model, but rather to develop a more efficient and scalable way of performance analysis for networks that bear a natural hierarchy and/or when certain type of hierarchical routing is used. It's worth pointing out that modern networks are getting larger and larger, and it is necessary that network engineering tools have scale-up capabilities. With the increase in size, a network tends to have clusters of nodes geographically, and hierarchical routing schemes are more commonly used in order to cope with large network size, e.g., ATM PNNI, IP based routing OSPF and hierarchical variations of OSPF. On the other hand, as we have pointed out in the introduction, past research in this area has addressed almost exclusively flat networks with flat routing schemes [7, 15, 9, 25]. This has provided us with strong motivation (as well as the related applied problems) to develop a hierarchical model to estimate end-to-end network performances.

We will examine two types of hierarchical routing schemes and the corresponding end-to-end connection level models. One is fixed, or near fixed routing, with the typical example being OSPF, which is widely used for Internet, IP based routing. Under this routing scheme, routes are established based on shorted distance principle, with ties broken according to lower IP address. Considering the fact that links normally fail on a much larger time scale compared to connection durations, this is a fixed routing scheme. In this case, the hierarchy of the network is primarily geographical, which comes from the fact that each workstation/network node within a LAN is connected to remote nodes via gateways on different levels, although there has been extensive study in adaptive variations of OSPF. The abstraction of the physical network results in interconnected gateways on higher layer(s).

The other type is dynamic/state dependent/adaptive hierarchical routing with the typical example being PNNI. Various proposals for QoS routing in the Internet also fall under this category [43, 44]. In this case, the centering point is "partial information". Networks are divided into clusters or peer groups that consist of neighboring nodes, with some nodes being "border nodes" that connect to other peer groups. All non-border nodes are only aware of their own peer group, and all border nodes are only aware of their own peer group and border nodes of other peer groups. Clearly border nodes represent some form of aggregation of the rest of the network in terms of routing. Routes are established on different layers based on complete information within a peer group and aggregated information between peer groups. The advantage of having a hierarchical end-to-end model is that it closely couples with the hierarchical nature of routing and uses only partial information on different layers. By segregating the network into layers we can also develop models for situations where different routing schemes are used within a group and between groups.

In the next section we describe network abstraction and aggregation. Hierarchical models for fixed hierarchical routing and dynamic hierarchical routing are presented in Section 3 and 4, respectively. In Section 5 we present numerical results for the fixed hierarchical routing case, which gained approximately 4-fold improvements in computational cost, as well as the results of dynamic hierarchical routing compared to simulation. We analyze the savings on computational cost in Section 6. Section 7 concludes this chapter.

3.2 Network Abstraction

We only consider large networks that have either physical hierarchies or routing hierarchies vs. a complete mesh since a hierarchical model promises clear incentives only for the former, if it is at all possible for the latter. Throughout this chapter we use a two-layer example shown in Figure 3.1.

There are three clusters in this example, with the dash-circles surrounding each one. Each group has a label/address, e.g., 1.1 indicates Layer 1, Peer Group 1.

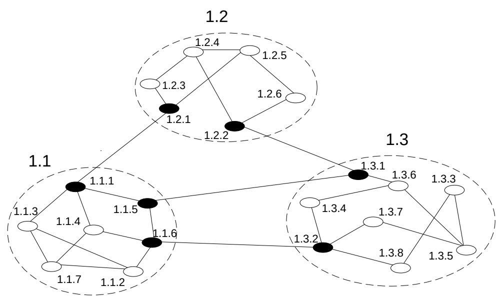

Figure 3.1: Network with three clusters – Layer One

Each node has an address as well, e.g., 1.1.3 is Node 3 of Peer Group 1.1. All border nodes are shown in black and non-border nodes are shown in white. A cluster can have a single or multiple border nodes. A border node can be connected to different clusters, e.g., Node 1.3.1. A non-border node does not necessarily have a direct link connected to border nodes, although this is often true with IP networks. Note that all links on this layer are actual, physical links.

The way aggregation and abstraction are done is as follows:

- All border nodes are kept in the higher layer – in this case Layer 2;
- Border nodes belonging to the same cluster are fully connected via "logical links".

This results in the Layer 2 abstraction shown in Figure 3.2.

A logical link may correspond to an actual link on the lower layer, e.g., Link 1.1.1 $\longleftrightarrow$ 1.1.5. The real logical links are shown in dashed lines, indicating a feasible path rather than a direct link.

As pointed out in [45], creating a logical link between each pair of border nodes

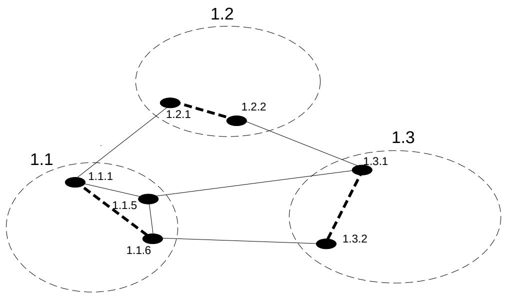

Figure 3.2: Network with three clusters – Layer Two

is the *full-mesh* approach, while collapsing the entire group into a single point is the *symmetric-point* approach. Our aggregation approach is a full-mesh one. While it may not be the most economic way of aggregation, this model clearly couples best with the underlying network physical structure and routing structure. It's worth pointing out that a bandwidth parameter is often assigned to a logical link, e.g., representing the maximum/average available bandwidth on the paths between two border nodes, and this may cause problems when different paths overlap [44]. It is obvious that some form of aggregated information has to be associated with either the logical links or the border nodes. However, as we will see in subsequent sections a bandwidth parameter is not necessarily the parameter in our model for calculation on the higher layer, thus avoiding the aforementioned problem. In our model, for the fixed routing case, this parameter is the blocking probability resulted from previous iterations within the group. For the dynamic routing case, this parameter can be implied costs, hop number or other criteria based on the dynamic/QoS routing policies being used.

3.3 Hierarchical Model for Fixed Routing

3.3.1 Notations

$G(1.n)$: the n^{th} cluster/peer group on Layer 1, where $n = 1, ..., N_1$, and N_1 is the total number of clusters in Layer 1.

$1.n.x_i$: node x in cluster $G(1.n)$, where $i = 1, ..., X_n$, and X_n is the total number of nodes in $G(1.n)$.

$1.n.y_i$: border nodes in cluster $G(1.n)$, where $i = 1, ..., Y_n$, and Y_n is the total number of border nodes in $G(1.n)$.

$1.n.x_1 \longrightarrow 1.n.x_2$: link from node $1.n.x_1$ to node $1.n.x_2$. Links in our model are directional.

$\lambda_s(1.n_1.x_1 \longrightarrow 1.n_2.x_2)$: offered load for class-s traffic from source $1.n_1.x_1$ to destination $1.n_2.x_2$, where $s = 1, ..., S$, and S is the total number of different traffic classes. It is also written as λ_{ps} with p as the p^{th} source-destination node pair.

$\mathcal{P} : (1.n_1.x_1 \longrightarrow 1.n_2.x_2)$: the route between node $1.n_1.x_1$ and $1.n_2.x_2$. $\mathcal{P}_p$ is the route for the p^{th} node pair.

3.3.2 Route and Route Segments

For our modeling purposes, each route is broken down into route segments as follows.

Path $\mathcal{P}(1.n_1.x_1 \longrightarrow 1.n_2.x_2)$ is a sequence of directed links with beginning and ending nodes. We break down a route into segments whenever a route exits or enters a cluster. So a route segment can be one of the following: source $\longrightarrow$ destination (when both source and destination nodes belong to the same cluster, same as original route); source $\longrightarrow$ border node; border node $\longrightarrow$ border node; border node $\longrightarrow$ destination.

Therefore, a typical route $\mathcal{P}(1.n_1.x_1 \longrightarrow 1.n_2.x_2)$ is segmented into the following k segments, assuming that $n_1 \neq n_2$ and that neither $(1.n_1.x_1)$ nor $(1.n_2.x_2)$ is a border node:

$$\mathcal{P}^1 : (1.n_1.x_1 \longrightarrow 1.n_1.y_2)$$

$$\mathcal{P}^2 : (1.n_1.y_2 \longrightarrow 1.n_i.y_1)$$

$$\mathcal{P}^3 : (1.n_i.y_1 \longrightarrow 1.n_i.y_2)$$

$$\ldots$$

$$\mathcal{P}^{k-1} : (1.n_j.y_2 \longrightarrow 1.n_2.y_1)$$

$$\mathcal{P}^k : (1.n_2.y_1 \longrightarrow 1.n_2.x_2),$$

where y_1 represents a border node from which traffic enters a cluster, and y_2 represents a border node from which traffic exits a group. These vary depending on the source-destination node pair. Note that when a segment $\mathcal{P}(1.n_i.y_1 \longrightarrow 1.n_i.y_2)$ exists, the route traverses an intermediate cluster before reaching the destination cluster. We denote the set of route segments for the p^{th} source-destination node by $\mathcal{P}_p^{'}$, and the segments $\mathcal{P}_p^1, ..., \mathcal{P}_p^k$, respectively.

The reason for a segmentation like the above is to segregate local computation (within each cluster) and higher layer (inter-cluster) computation.

3.3.3 Initial Offered Load and Local Relaxation

With the segmentation of routes, we need a corresponding way of representing the offered traffic load prior to running the hierarchical algorithm.

The offered load of class-s traffic of the p^{th} node pair $(1.n_1.x_1, 1.n_2.x_2)$ is $\lambda_s^0(1.n_1.x_1 \longrightarrow 1.n_2.x_2)$. We substitute this with a combination of the following, in a similar way as route segmentation:

$$\begin{array}{ll}
\lambda_{ps}^0(1.n_1.x_1 \longrightarrow 1.n_1.y_2) & \text{source cluster traffic} \\
\lambda_{ps}^0(1.n_1.y_2 \longrightarrow 1.n_i.y_1) & \text{inter-cluster traffic} \\
\lambda_{ps}^0(1.n_i.y_1 \longrightarrow 1.n_i.y_2) & \text{cluster } i \text{ traffic} \\
\qquad\qquad \dots & \\
\lambda_{ps}^0(1.n_j.y_2 \longrightarrow 1.n_2.y_1) & \text{inter-cluster traffic} \\
\lambda_{ps}^0(1.n_2.y_1 \longrightarrow 1.n_2.x_2) & \text{destination cluster traffic}
\end{array}$$

These terms all take the value of the initial offered load $\lambda_s^0(1.n_1.x_1 \longrightarrow 1.n_2.x_2)$. Thus we have complete traffic input information (together with route segments) for each cluster.

For the i^{th} cluster, offered loads indexed with same node pair are added up to represent the aggregated traffic for this node pair. We assume that at least one of the nodes is a border node since no aggregation process is needed in cases where both nodes are non-border nodes within the same group. Without loss of generality, assume that the destination node is a border node,

$$\lambda_s^1(1.n_i.x_1 \longrightarrow 1.n_i.y_2) =$$

$$\sum_{\{p:(1.n_i.x_1 \longrightarrow 1.n_i.y_2)\in \mathcal{P}_p'\}} \lambda_{ps}^0(1.n_i.x_1 \longrightarrow 1.n_i.y_2). \tag{3.1}$$

Note that the source node x_1 can also be a border node.

In doing so, we assume initial condition of zero inter-cluster blocking, and zero blocking in remote clusters. This is part of the initial values and blocking on inter-cluster links are calculated in the next step of the algorithm.

The reduced load model for fixed routing is then applied to every cluster separately using these offered loads to calculate group-wide blocking probabilities:

$$\begin{aligned} B_s(\mathcal{P}^1) &= B_s(1.n_1.x_1 \longrightarrow 1.n_1.y_2) \\ B_s(\mathcal{P}^3) &= B_s(1.n_i.y_1 \longrightarrow 1.n_i.y_2) \\ &\dots \\ B_s(\mathcal{P}^k) &= B_s(1.n_2.y_1 \longrightarrow 1.n_2.x_2) \end{aligned}$$

3.3.4 Reduced Load and Higher Layer Relaxation

On the higher layer (second layer in our example), only border nodes exist. We construct a new network with border nodes, inter-group links and logical links as illustrated in Figure 3.2. For this logical network we consolidate the previous route segments into three parts: within the source cluster ($\mathcal{P}^1$), between source and destination clusters ($\mathcal{P}^o = \mathcal{P}^2 \cup ... \cup \mathcal{P}^{k-1}$) and within the destination cluster

$(\mathcal{P}^k)$. We have the following offered load for the second segment:

$$\begin{aligned}
&\lambda_s^1(1.n_1.y_2 \longrightarrow 1.n_2.y_1) = \\
&\lambda_s^0(1.n_1.y_2 \longrightarrow 1.n_2.y_1) + \\
&\sum_{\{p:(1.n_1.y_2 \longrightarrow 1.n_2.y_1) \in \mathcal{P}'_p\}} \lambda_{ps}^0(1.n_1.y_2 \longrightarrow 1.n_2.y_1) \cdot \\
&B_s(1.n_1.x_1 \longrightarrow 1.n_1.y_2) \cdot B_s(1.n_2.y_1 \longrightarrow 1.n_2.x_2)
\end{aligned}$$

i.e.,

$$\begin{aligned}
\lambda_s^1(\mathcal{P}^o) &= \lambda_s^0(\mathcal{P}^o) + \\
&\sum_{\{p:\mathcal{P}^o \in \mathcal{P}'_p\}} \lambda_{ps}^0(\mathcal{P}_p^o) \cdot B_s(\mathcal{P}_p^1) \cdot B_s(\mathcal{P}_p^k).
\end{aligned}$$

This is the initial offered load thinned by blocking in both the source and destination clusters.

We now have the complete traffic input on the higher layer. We apply again the reduced load approximation to this layer and calculate second-layer end-to-end blocking probabilities. Note that on this layer, we do not have a "capacity" parameter for the logical links, but instead, an end-to-end blocking probability that resulted from previous approximation within the cluster (local approximation). This is kept fixed throughout the approximation on this layer and is only updated through local approximation. The result of this step is the blocking probability between border nodes:

$$B_s(1.n_i.y_1 \longrightarrow 1.n_i.y_2) \quad \text{and}$$

$$B_s(1.n_i.y_2 \longrightarrow 1.n_j.y_1).$$

3.3.5 Iterations

Using the results from the higher layer approximation, update the offered load in (3.1) with the blocking probabilities calculated from the higher layer:

$$\begin{aligned}
&\lambda_s^1(1.n_i.x_1 \to 1.n_i.y_2) = \\
&\sum_{\{p:(1.n_i.x_1 \to 1.n_i.y_2 \in \mathcal{P}'_p\}} \lambda_{ps}^0(1.n_i.x_1 \to 1.n_i.y_2) \cdot \\
&\prod_{\{k:\mathcal{P}^k \neq (1.n_i.x_1 \to 1.n_i.y_2)\}} B_s(\mathcal{P}_p^k). \qquad (3.2)
\end{aligned}$$

This is essentially the original offered load thinned by blocking on inter-group links and remote groups. This becomes the new input for local relaxation. Local and higher layer relaxations are then repeated till the difference between results from successive iterations are within certain criteria.

3.4 Hierarchical Model for Dynamic Hierarchical Routing

There are numerous existing and proposed dynamic/QoS hierarchical routing schemes, each of which results in different end-to-end performances determined by the scope and design trade-off of the routing scheme. Our primary goal here is not to design an end-to-end model for each of these schemes. Rather, we attempt to present an end-to-end performance modeling framework that considers a generic type of dynamic hierarchical routing, which captures some of the most basic properties of a majority of such routing schemes. We make assumptions for simplicity purposes, but our work shows how an end-to-end performance model can be closely coupled with routing policies to provide an efficient way of analysis. Furthermore, our model enables us to analyze situations where different routing schemes are used on different levels of a network.

3.4.1 Dynamic Hierarchical Routing

One key property of any dynamic hierarchical routing is *inaccurate/incomplete information* [46]. A node has complete information on its own group, but only aggregated information on other groups advertised by the border nodes. So a node typically sees its own group in detail but other groups “vaguely”, in form of certain representation of the aggregated information, which is inevitably incomplete and inaccurate. This aggregated information can be one or more of various metrics specified by the routing algorithm: implied cost of a group (cost/additional blocking incurred by allowing a traffic stream to go through), maximum available bandwidth between border node pairs, delay incurred by going through a group, etc.. This information is typically associated with and advertised by border nodes.

In source routing, a path is selected with detailed hop-by-hop information in the originating group but only group-to-group information beyond the originating group. The detailed routing within each other group is determined locally. For example, a route from 1.1.7 to 1.3.3 is selected from the source point of view as 1.1.7 $\longrightarrow$ 1.1.6 $\longrightarrow$ 1.3, or as 1.1.7 $\longrightarrow$ 1.1.1 $\longrightarrow$ 1.2 $\longrightarrow$ 1.3. The choice of routes within a group is determined using shortest path routing, least loaded routing and so on, along with the aggregated information advertised by border nodes. We do not specify the details in formulating the model since they do not affect the general method of analysis. However we give specific examples for numerical and simulation studies.

In our model, we focus on dynamic hierarchical source routing where routes are selected in a way described above. We do not consider crankback in which a connection is routed on an alternative route if the first choice is not available. A call is blocked if the route selected according to the dynamic routing policy does not have the required bandwidth.

3.4.2 Probabilistic Offered Load Distribution and Traffic Aggregation

With dynamic hierarchical routing the model becomes more complicated because there is no longer a single fixed route between nodes. The key point of the model is to successfully separate traffic from cluster to cluster. In order to do so, we made the following observation. One of the main advantages of dynamic routing is load balancing, i.e., dynamically distribute traffic flow onto different paths of the network to achieve greater utilization of network resources. We argue that under steady state, a particular traffic flow (defined by class, source-destination node pair) is distributed among all feasible routes, and among multiple border nodes that connect to other groups. (This problem does not exist when there is only one border node. Routes are still dynamically chosen, but all routes ultimately go through that single border node.) The fraction of a traffic flow that goes through a certain border node is directly related to the aggregated information/metrics for the group-to-group route the border node advertises, and remains relatively fixed under steady state.

Based on this, for a pair of nodes belonging to different clusters, the feasible route set is divided into three subsets: route segments within the source cluster, route segments between clusters and route segments within the destination cluster. In this section we use $\mathcal{P}$ to represent a set of routes since multiple routes are allowed for each node pair in dynamic hierarchical routing.

To simplify notation, assume the route does not traverse an intermediate cluster (extensions can be made). For route set $\mathcal{P}(1.n_1.x_1 \longrightarrow 1.n_2.x_2)$ the subsets are:

$$\begin{aligned}
&\mathcal{P}^1 : (1.n_1.x_1 \longrightarrow 1.n_1.y_i),\\
&\mathcal{P}^2 : (1.n_1.y_i \longrightarrow 1.n_2.y_i), \quad \text{and}\\
&\mathcal{P}^3 : (1.n_2.y_i \longrightarrow 1.n_2.x_2),
\end{aligned}$$

where y_i indicates all possible border nodes. Each of these segments presents possibly many routes. We are simply breaking down the initial source-destination into multiple intermediate source-destination node pairs so that routes within a group and between groups are segregated.

We rewrite the second subset as $\mathcal{P}^2(1.n_1 \rightarrow 1.n_2)$, a collection of routes from source cluster $1.n_1$ to the destination cluster $1.n_2$. For each route in $\mathcal{P}^1$, we have the initial offered load

$$\lambda_{ps}^{0}(1.n_1.x_1 \to 1.n_1.y_i) = a_i \lambda_s^0(1.n_1.x_1 \to 1.n_2.x_2)$$

where $\lambda_s^0(1.n_1.x_1 \to 1.n_2.x_2)$ is the offered load of the class-s traffic for node pair $(1.n_1.x_1 \to 1.n_2.x_2)$, and each border node y_i gets to route a portion a_i of the traffic with $\sum_i a_i = 1$.

The traffic is further distributed to each route in $\mathcal{P}^2$ based on the underlying routing scheme used on the second layer:

$$\lambda_{ps}^{0}(1.n_1.y_i \to 1.n_2.y_j) = a_i q_{ij} \lambda_s^0(1.n_1.x_1 \to 1.n_2.x_2)$$

where $\sum_j q_{ij} = 1$ for all i, and can be calculated based on the reduced load mode used for this layer.

Finally for the routes in the destination cluster, $\mathcal{P}^3$, the initial offered load is

$$\lambda_{ps}^{0}(1.n_2.y_j \to 1.n_2.x_2) =$$
$$\sum_i a_i q_{ij} \lambda_s^0(1.n_1.x_1 \to 1.n_2.x_2)$$

assuming the initial condition of zero blocking else where.

Within each cluster the traffic is then aggregated over all node pairs that have the same route in one of their route sets, similar to what we described in the previous section.

3.4.3 Updates and Iterations

From the aggregate traffic we apply the reduced load model to each cluster and compute

$$B_s(1.n_1.x_1 \rightarrow 1.n_1.y_i),$$

$$B_s(1.n_1.y_i \rightarrow 1.n_2.y_j),$$

$$B_s(1.n_1.y_j \rightarrow 1.n_2.x_2).$$

As discussed earlier, the distribution of traffic flow onto the different inter-cluster routes should match the aggregated information (delay, blocking probability, implied cost, available bandwidth, etc.) advertised by different border nodes. Ultimately one of the goals for any dynamic routing scheme is to balance traffic load on different alternative routes, and the end result is that these alternative routes should have equivalent QoS under steady state. For example, if we use blocking probability as a criteria to adjust the traffic distribution $a_i, i = 1, 2, ..., n$, with n being the total number of border nodes in a cluster, then the border node with a blocking probability higher than median gets a decreased portion, and border nodes with a blocking probability lower than median gets an increased portion:

$$a_i := a_i + \delta \quad \text{if} \;\; B_s(1.n_1.y_i \rightarrow 1.n_2) < B_m;$$

$$a_i := a_i - \delta \quad \text{if} \;\; B_s(1.n_1.y_i \rightarrow 1.n_2) > B_m,$$

where $B_s(1.n_1.y_i \rightarrow 1.n_2)$ is the average blocking over all routes from $1.n_1.y_i$ to $1.n_2$, and δ is a small incremental value and B_m is the median blocking probability among all routes. Alternatively a_i can be set to be inversely proportional to the blocking probabilities with sum 1. Other means of relating traffic distribution to route QoS can also be specified.

q_{ij} is updated accordingly based on the reduced load model we use for the second layer. An example in least-loaded routing can be found in [38].

Using these new distribution values along with the blocking probabilities we update the aggregated traffic load for node pairs within the same cluster, similar to the fixed routing scenario. Another round of iteration is then started and the process continues until both the distribution a_i and the link blocking probabilities converge.

3.5 Numerical Results

In this section we present numerical experiment results for the network example shown in 3.1 using fixed hierarchical routing scheme, and dynamic hierarchical routing scheme. This is a 21-node, 30-link, 3-clusters, 2-layer network model. We use a single class of traffic requiring unit bandwidth. Link capacities vary from 80 to 160, which are listed in Appendix A. We also provide the offered traffic load between node pairs at a "nominal" level in the Appendix. The intensity of this traffic load is shown in Table 3.1, in which *load* is defined as the ratio between the total rate of traffic coming out of a node and the total out-going link bandwidth connecting to this node. At the nominal level, the value of this ratio for each node is around 0.05. In addition to this offered traffic load we also define a “weight” in our experiment as a multiplier to the nominal traffic, so that we get twice, three times of the nominal traffic, etc..

3.5.1 Fixed Hierarchical Routing

Since when using fixed hierarchical routing a network can always be treated as flat, we compare the performance of flat fixed-point approximation (FPA) and the hierarchical fixed-point approximation. It can be shown [7] that under a certain limiting regime for fixed routing the fixed point approximation is asymptotically correct.

Results of Flat FPA and Hierarchical FPA

Table 3.2 is a comparison between flat fixed-point approximation and hierarchical fixed-point approximation on individual link blocking probabilities (end-to-end blocking probabilities are computed directly from these for fixed routing). We used seven times nominal traffic (weight $= 7$).

We see that the hierarchical scheme gives very close results compared to that of the flat approximation scheme, but achieved $3 \sim 4$-fold improvement in computation.

Varying Traffic Load

Table 3.3 is a comparison between the runtime of flat FPA and hierarchical FPA while varying the traffic load by increasing the value of "weight", which is multiplied to the nominal traffic rate. This result is also plotted in Figure 3.3. As the traffic load increases, the gain in computational savings becomes bigger.

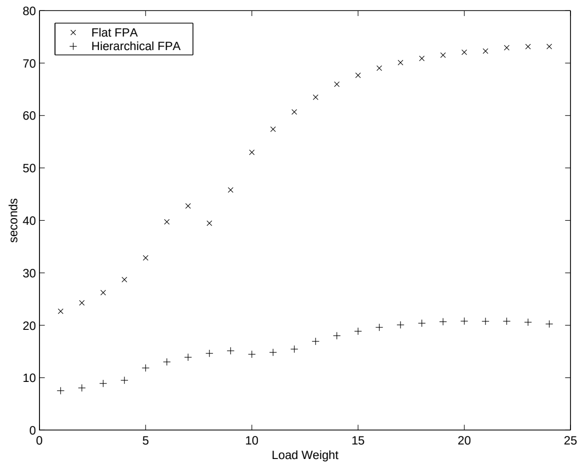

Figure 3.3: Run time vs. traffic load.

We see that as the traffic load increases, the gain in computation savings becomes significant. More importantly, the computation of the hierarchical FPA only increases marginally with the increase of traffic load.

Varying Cross-group Link Capacity

Since the sequence of iterations is determined by clusters, it is reasonable to expect that changes to capacities of links in a particular cluster/layer will have an effect on how much faster the hierarchical scheme runs comparing to the flat scheme.

Table 3.4 is a comparison between the runtime of flat FPA and hierarchical FPA while varying the capacities of links connecting two different clusters. This results is also shown in Figure 3.4. We see that the run time difference does not change much with the increase in link capacities. However the gain slightly increases when the link capacities are reduced. A possible explanation is that these links can easily become bottlenecks when the capacities are reduced, and by separating the global computation from local computation we get faster convergence.

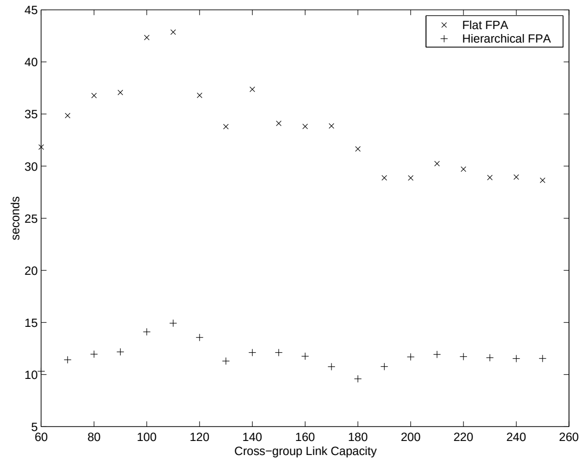

Figure 3.4: Run time vs. link capacities.

3.5.2 Dynamic Hierarchical Routing

We use the same network example, with shortest path routing within each cluster, but use least-loaded routing between clusters. As we pointed out before, Least-loaded routing (LLR) is a form of bandwidth-optimization QoS routing [31]. A source node chooses a border node based on the advertised average blocking between the border node and the destination cluster $B_s(1.n_1.y_i \rightarrow 1.n_2)$), and a border node chooses the route that has the most free capacity among all routes from itself to the destination cluster. We used the reduced load model for least-loaded routing we developed in Chapter 2. The distribution of traffic among border nodes is inversely proportional to the advertised blocking probability. and the distribution q_{ij} is the probability that route $(1.n_1.y_i \rightarrow 1.n_2.y_j)$ has the most free bandwidth.

The following tables show the comparison between the results of the hierarchical model and the discrete event simulation (DES), with weight being 5, 10, 15 and 20, respectively.

We see that as the traffic increases, the model generates better approximations. Overall the approximation is satisfactory. The run time for the approximation is around 11-15 seconds while the simulation typically takes 5-20 minutes to converge depending on the traffic load.

3.6 Conclusion

In this chapter we presented the hierarchical reduced load approximation method for networks with either fixed hierarchical routing or dynamic hierarchical routing policies. It can also be used in cases where different routing schemes are used in different regions of a network. Our numerical experiment results showed significant improvement in computational cost, and the validity of this method. Our experiment network is relatively small, however we believe that this is a novel approximation method for efficient and scalable performance analysis for much larger networks.

Table 3.1: Nominal offered traffic load

Node	Rate	Cap.	load
1.1.1	11.15	270	0.041296
1.1.2	7.20	160	0.045000
1.1.3	10.60	180	0.058889
1.1.4	7.90	150	0.052667
1.1.5	8.60	210	0.040952
1.1.6	8.95	220	0.040682
1.1.7	10.45	180	0.058056
1.2.1	8.95	210	0.042619
1.2.2	8.80	220	0.040000
1.2.3	6.65	130	0.051154
1.2.4	9.15	180	0.050833
1.2.5	9.70	180	0.053889
1.2.6	7.95	140	0.056786
1.3.1	9.55	230	0.041522
1.3.2	12.0	290	0.041379
1.3.3	3.70	80	0.046250
1.3.4	7.90	140	0.056429
1.3.5	8.25	160	0.051562
1.3.6	9.00	180	0.050000
1.3.7	8.00	130	0.061538
1.3.8	5.75	100	0.057500

Table 3.2: Comparison of results, weight = 7.

link	Hier. FPA	Flat FPA
(1.1.1-1.1.5)	0.235701	0.235704
(1.1.1-1.2.1)	0.409961	0.409955
(1.1.2-1.2.6)	0.526158	0.526159
(1.1.4-1.1.6)	0.148341	0.148333
(1.1.5-1.1.6)	0.006189	0.006190
(1.1.2-1.1.7)	0.000000	0.000000
(1.1.5-1.3.1)	0.004523	0.004523
(1.1.6-1.3.2)	0.054767	0.054766
(1.2.2-1.3.1)	0.122970	0.122967
(1.2.2-1.2.4)	0.000007	0.000007
(1.3.3-1.3.5)	0.000000	0.000000
time (sec)	13.90	42.76

Weight	1	2	3	4	5	6
Flat FPA (sec)	22.66	24.28	26.23	28.70	32.85	39.73
Hierarchical FPA (sec)	7.52	8.05	8.90	9.51	11.86	13.00
Weight	7	8	9	10	11	12
Flat FPA (sec)	42.76	39.46	45.81	52.98	57.40	60.69
Hierarchical FPA (sec)	13.90	14.64	15.12	14.46	14.83	15.46
Weight	13	14	15	16	17	18
Flat FPA (sec)	63.49	65.97	67.67	69.03	70.09	70.89
Hierarchical FPA (sec)	16.93	18.00	18.85	19.59	20.08	20.40
Weight	19	20	21	22	23	24
Flat FPA (sec)	71.51	72.07	72.29	72.94	73.15	73.17
Hierarchical FPA (sec)	20.67	20.79	20.76	20.78	20.58	20.25

Table 3.3: Run time difference with increasing traffic load

Cross-group link capacity	60	70	80	90	100
Flat FPA	31.83	34.85	36.77	37.07	42.34
Hierarchical FPA	10.32	11.42	11.96	12.18	14.10
Cross-group link capacity	110	120	130	140	150
Flat FPA	42.86	36.79	33.79	37.37	34.10
Hierarchical FPA	14.93	13.56	11.30	12.11	12.11
Cross-group link capacity	160	170	180	190	200
Flat FPA	33.81	33.86	31.65	28.88	28.87
Hierarchical FPA	11.76	10.76	9.59	10.77	11.69
Cross-group link capacity	210	220	230	240	250
Flat FPA	30.24	29.71	28.91	28.95	28.64
Hierarchical FPA	11.93	11.72	11.62	11.53	11.54

Table 3.4: Run time difference with increasing capacity for cross-group links

Node Pair	Hier. FPA	DES
(1.1.3-1.3.3)	0.00077	0.00000
(1.1.2-1.2.4)	0.00000	0.00000
(1.1.6-1.3.7)	0.00000	0.00000
(1.1.1-1.2.3)	0.00000	0.00000
(1.2.1-1.2.6)	0.00191	0.00453
(1.3.1-1.3.8)	0.00000	0.00000
(1.3.5-1.3.6)	0.00000	0.00000

Table 3.5: Comparison of results, weight = 5.

Node Pair	Hier. FPA	DES
(1.1.3-1.3.3)	0.28784	0.25235
(1.1.2-1.2.4)	0.00470	0.00166
(1.1.6-1.3.7)	0.09745	0.09995
(1.1.1-1.2.3)	0.10185	0.11089
(1.2.1-1.2.6)	0.35567	0.36216
(1.3.1-1.3.8)	0.06322	0.05275
(1.3.5-1.3.6)	0.00000	0.000000

Table 3.6: Comparison of results, weight = 10.

Node Pair	Hier. FPA	DES
(1.1.3-1.3.3)	0.54327	0.52136
(1.1.2-1.2.4)	0.05856	0.04963
(1.1.6-1.3.7)	0.29617	0.30721
(1.1.1-1.2.3)	0.23567	0.24354
(1.2.1-1.2.6)	0.53049	0.54061
(1.3.1-1.3.8)	0.26342	0.25285
(1.3.5-1.3.6)	0.00000	0.00000

Table 3.7: Comparison of results, weight = 15.

Node Pair	Hier. FPA	DES
(1.1.3-1.3.3)	0.66527	0.66421
(1.1.2-1.2.4)	0.16374	0.16717
(1.1.6-1.3.7)	0.42122	0.43112
(1.1.1-1.2.3)	0.33676	0.34187
(1.2.1-1.2.6)	0.62975	0.62494
(1.3.1-1.3.8)	0.39753	0.39476
(1.3.5-1.3.6)	0.00000	0.00000

Table 3.8: Comparison of results, weight = 20.

Chapter 4

PERFORMANCE MODELS USING DELAY BASED QoS ROUTING

4.1 Introduction

In the previous two chapters, we presented models for estimating connection blocking when using least-loaded routing and hierarchical routing, the former being a specific routing scheme and the latter being a routing strategy (vs. source routing and distributed routing). Least-loaded routing is also called bandwidth-optimization routing, in terms of QoS when the QoS metrics are residual bandwidth or residual buffer space.

In this chapter we examine another category of QoS routing called path-optimization/constrained routing, which is often used when the QoS metrics are delay, delay jitter, and cost. The state of a path is determined by the combined state over all links on the path. An example of this category is delay-constrained routing, which is to find a path whose delay is bounded by a required value.

A major task in establishing a connection with certain QoS guarantees is to map the end-to-end QoS requirements into local (nodal) requirements, which indicates how to reserve resources along the route. For example, the ability to provide end-to-end delay guarantees depends to a large extent on the scheduling and on the policy and service discipline employed by the nodes. Such disciplines at the nodes determine the bounds on the maximal delay incurred at a node, from which the bound on the end-to-end delay can be derived [47].

Such a bound is generally used to quantify the quality of a path in terms of meeting the delay requirement. The routing problem then is to find a path that has the best performance based on such a bound and w.r.t. the specified QoS requirement. The corresponding problem of connection blocking analysis is to find out to what extent such a path exists (can be found). Normally schedulers map delay guarantees into rate guarantees and advertise the residual rate available at a node. In particular, the guaranteed service class proposed for the Internet [48] is based on such "rate-based" principles. Consequently, some recent studies [46, 49, 50] were also aiming at taking advantage of the properties of such schedulers when computing paths that satisfy end-to-end delay bounds.

In this chapter, we also use such properties of the rate based schedulers and map delay components into rate components, and apply similar ideas of fixed point approximation to build our model for estimating connection blocking. In the discussion that follows, we first present formulations for QoS routing with rate based schedulers, and then present the connection blocking analysis based on such

QoS routing schemes.

4.2 Delay-Based QoS Routing

Following some of our notations in previous chapters, consider a network with N nodes, J links indexed by j. Each link j is characterized by the following two values:

- A maximal rate (capacity) C_j and residual bandwidth R_j. The latter is the maximal available rate a link can offer to a new connection. When a new connection with a reserved rate $r < R_j$ is established through link j, this value reduces to $R_j - r$;
- A constant delay d_j, related to the link speed, propagation delay and maximal transfer unit.

A connection t is characterized by the following values:

- Source-destination node pair indexed by t;
- A bias value σ_t, related to the connection's maximal burst;
- The maximal packet size c_t;
- The maximal end-to-end delay constraint D_t;
- The bandwidth requirement b_t,

where the last two items are QoS requirements in this case.

Let p denote a valid route/path between node pair i, and let $h(p)$ denote the number of hops on this route. Also, denote by $r(p)$ the maximal residual rate along the path, i.e.,

$$r(p) = \max_{j \in p} R_j.$$

Assuming that the scheduling policy in the network belongs to the "rate-based" class as we discussed in Section 4.1, when a connection t is routed over this path p with a reserved rate r, where $r \leq r(p)$, we have the following upper bound on the end-to-end delay [47]:

$$D_t(p, r) = \frac{\sigma_t + h(p) \cdot c_t}{r} + \sum_{j \in p} d_j. \tag{4.1}$$

Let $D_t(p)$ be the minimal possible value of this bound, i.e.,

$$D_t(p) = \min_r D_t(p, r) = D_t(p, r(p)). \tag{4.2}$$

$D_t(p)$ is referred to as the *guaranteed delay* of p. A path is said to be *feasible* if both of the following hold:

$$\begin{aligned} D_t(p) &\leq D_t; \\ r(p) &\geq b_t. \end{aligned} \tag{4.3}$$

Therefore, for each connection t, finding a corresponding feasible path p_t includes both finding the path and a rate r_t so that the following conditions are true:

- For all t, $((\sigma_t + h(p_t) \cdot c_t)/r_t) + \sum_{j \in p_t} d_j \leq D_t$;

- For all j, $\sum_{\{t:j\in p_t\}} r_t \leq C_j$;

- For all t, $r_t \geq b_t$.

4.3 Connection Blocking Analysis

From Equations (4.1) and (4.3), in order to carry connection t on path p with a bound on delay requirement D_t, the rate r has to satisfy the following:

$$r \geq \frac{\sigma_t + h(p)\cdot c_t}{D_t - \sum_{j\in p} d_j}. \tag{4.4}$$

Along with the requirement on bandwidth b_t, i.e., $r \geq b_t$, the minimum rate we need to reserve form this connection is the maximum of $\{\frac{\sigma_t+h(p)\cdot c_t}{D_t-\sum_{j\in p} d_j}, b_t\}$, given that the residual bandwidth on each link along this path is no less than this value.

By now we have successfully mapped the end-to-end delay requirements into link rate requirements, and this rate accounts for all four components of delay: propagation delay, processing delay and transmission delay, through $h(p)$, c_t and d_j. Queuing delay is not an issue here since by reserving equal rate/bandwidth along the path, we eliminated queuing delay at nodes incurred by service scheduling. Queuing delay due to burstiness is accounted for by σ_t.

For a fixed connection and a fixed route, r_t can be pre-computed and b_t is given, i.e., reserved rate is known, therefore connection blocking for this delay-based QoS routing can be calculated by the fixed-point approximation constructed in a way similar to what we described in previous chapters.

However, one major difference arises in this case in that the required bandwidth for a connection is no longer fixed, but depends on routing, i.e., the reserved bandwidth varies from one route to another due to different hop counts and link propagation delays. The difference could be marginal in local areas but will not be negligible with the presence of parameters like link costs. On the other hand we still need to use the reserved bandwidth as a metric to identify a “class” of traffic in order to be able to use similar single link analysis. This leads to *reclassification*, under which a class of traffic is identified by its QoS requirements, arrival rate, origin and destination nodes and the path on which it is routed. As a result, the state space we need to consider is increased roughly by N_R times, with N_R being the average number of allowed routes between two nodes.

Furthermore, the bandwidth requirements are no longer discrete numbers. This in theory will not affect the link stationary distribution computation since the state space is finite. Recall from Chapter 2, that the stationary distribution of the number of calls from each class in progress on a single link is provided by

$$p(\mathbf{n}) = G^{-1} \prod_s \frac{r_s^{n_s}}{n_s!}, \qquad \sum_s r_s n_s \leq C$$

where r_s is the reserved bandwidth for class-s traffic. Consider certain *quantization* that uses a small unit for bandwidth requirement and link capacity such that all rates r_s are multiples of this unit. Following this, the Kaufman recursion can still be applied.

4.4 Conclusions

We presented how to map delay requirements into rate requirements using rate-based schedulers. We further showed how to use the reduced load model to estimate connection blocking of a delay-based QoS routing scheme in such a network. There could be potential computational difficulties because of increased state space, which is a possible future research area.

Chapter 5

AUTOMATIC DIFFERENTIATION FOR SENSITIVITY ANALYSIS

5.1 Introduction

As pointed out in the introduction chapter, typically a network design and dimensioning problem is formulated into a constraint optimization problem, with the objective function being maximizing revenue or minimizing cost, and the constraints being QoS requirements, which are determined by some performance evaluation model such as we presented in previous chapters. This formulation is usually solved iteratively to check the feasibility of QoS constraints and to guide the search algorithm. This involves gradient calculation. A lot of previous work in this area try to derive a closed form presentation of the shadow price or implied cost. Such derivations may vary depending on the performance model being used. Furthermore, manually calculating gradients is often difficult for complicated mathematical models, if at all possible, and thus generally leads to further approximation, a part from the approximation introduced by the models themselves.

We identify the technique of Automatic Differentiation (AD) as an appropriate tool to use for our purpose of developing a systematic approach, which is relatively independent of the underlying performance model, to solving the constraint optimization problem.

Automatic, or computational, differentiation is a chain rule based technique for evaluating the derivatives of functions defined by algorithms, usually in the form of computer programs written in Fortran, C, or some other high level language. If the program theoretically can be unrolled into a finite sequence of arithmetic operations and elementary function calls, then derivatives can be propagated recursively. Exceptions arise when there is a division by zero or when one of the elementary functions is evaluated at a point of non-differentiability. These local contingencies are easily detected and arise only in marginal situations where the undifferentiated evaluation algorithm is already poorly conditioned.

However, just because the derivatives computed by automatic differentiation are those defined by the statements that were executed by a particular program run, which is what was actually computed and may differ significantly from the derivative of the function one intended to compute [51]. This is especially true in the iterative evaluation of a function defined implicitly or otherwise. Usually the iteration continues until the value of $f(x)$ meets certain criteria. However, this may not be true for the value of $f'(x)$ or higher derivatives. On the other hand, many engineering problems are impossible or impractical to be expressed in an explicit or exact form, and we have to turn to approximations which often take an iterative form. The problems lie in the design of programs to which AD is to be applied and can be handled most effectively by the programmer, especially for pitfalls arising from branching or iteration.

In this chapter we focus on using AD for iterative processes, especially fixed point problems like those presented in the previous chapters. It is organized as follows: in Section 2 we investigate the sufficient conditions under which the derivatives of such fixed point functions, computed using automatic differentiation, converge. In Section 3, we use the AD package ADIC for our fixed point algorithm presented in Chapter 2 to get the sensitivities of blocking probabilities with respect to traffic load. The efficiency of ADIC is also evaluated. Section 4 concludes this chapter.

The idea behind automatic differentiation (AD) is not a new one. The utility of computers for evaluating functions defined by formulas has long been recognized. And since differentiation of functions defined by formulas is a mechanical process done according to fixed rules, it is highly suitable for automation along the same lines as function evaluation. However, the technique of automatic differentiation did not become popular and really applicable until pretty recently, due to the remarkable work by Andreas Griewank and Christian Bischof, who is also the major contributor of the AD package ADIFOR for FORTRAN and the recently available ADIC (test version) for C.

5.2 Derivative Convergence of Functions Involving One or More Fixed Points

Recall the fixed point algorithm we discussed in previous chapters. Without loss of generality we assume two fixed points with two mappings throughout this chapter, corresponding to the link reduced load and the link admissibility probability, respectively. Extensions to a single fixed point or more can easily be obtained through similar discussion.

We have two mappings

$$\begin{aligned} y &= f(x,t), \quad f : \mathcal{R}^m \times \mathcal{R}^n \to \mathcal{R}^l \\ x &= g(y,t), \quad g : \mathcal{R}^l \times \mathcal{R}^n \to \mathcal{R}^m \end{aligned}$$

where t is the independent variable. The fixed point algorithm can be presented in the following iterative process:

```
Given $t$ and $x_0$,
Until $\|y_{k+1} - y_k\| \le Tol_y$ or $\|x_{k+1} - x_k\| \le Tol_x$
    $y_k = f(x_k, t)$
    $x_{k+1} = g(y_k, t)$
    $k = k + 1$
```

Suppose (x_k, y_k) converges to the fixed point (x_*, y_*), we get

$$z_* = h(x_*, y_*).$$

Applying automatic differentiation, the resulting derivative evaluation along with the original function evaluation is as follows, denoting $f_x(x,t)$ and $f_t(x,t)$ as the partial derivatives w.r.t. the first and second arguments, respectively:

```
Given $t$, $x_0$ and $x_0^{'}$,
Until $\|y_{k+1} - y_k\| \le Tol_y$ or $\|x_{k+1} - x_k\| \le Tol_x$
Until $\|y_{k+1}^{'} - y_k^{'}\| \le Tol_{y'}$ or $\|x_{k+1}^{'} - x_k^{'}\| \le Tol_{x'}$
    $y_k = f(x_k, t)$
    $y_k^{'} = f_x(x_k, t)x_k^{'} + f_t(x_k, t)$
    $x_{k+1} = g(y_k, t)$
    $x_{k+1}^{'} = g_y(y_k, t)y_k^{'} + g_t(x_k, t)$
    $k = k + 1$
```

Suppose in addition to $(x_k, y_k) \longrightarrow (x_*, y_*)$, the derivatives $(x_k^{'}, y_k^{'}$ converge to $(x_*^{'}, y_*^{'},$

$$\begin{aligned} z_* &= h(x_*, y_*) \\ z_*^{'} &= h_x(x_*, y_*)x_*^{'} + h_y(x_*, y_*)y_*^{'}. \end{aligned}$$

We proceed to derive the sufficient condition for derivative convergence of the above iterative process. Our proof of the following lemma is motivated by Griewank and Bischop [52].

Lemma 5.1 *For an iterative process shown above, assume that*

1. *for any given $t \in \mathcal{R}^n$, the fixed point (x_*, y_*) is a solution to the iteration,*

i.e.,

$$\begin{aligned} y_* &= y_*(t) = f(x_*, t) \\ x_* &= x_*(t) = g(y_*, t); \end{aligned}$$

2. *for all (x, y) such that $\|(x, y) - (x_*, y_*)\| < \rho$ for some small ρ, both f and g are Lipschitz continuously differentiable with respect to both arguments, such that for some constant L_1, L_2, L_3 and L_4,*

$$\|f_x(x, t) - f_x(x_*, t)\| \leq L_1\|x - x_*\| \tag{5.1}$$

$$\|f_t(x, t) - f_t(x_*, t)\| \leq L_2\|x - x_*\| \tag{5.2}$$

$$\|g_y(y, t) - g_y(y_*, t)\| \leq L_3\|y - y_*\| \tag{5.3}$$

$$\|g_t(y, t) - g_t(y_*, t)\| \leq L_4\|y - y_*\|, \tag{5.4}$$

In addition, for some constant L_5, L_6 and L_7,

$$\|g_y(y, t)f_x(x, t) - g_y(y_*, t)f_x(x_*, t)\| \leq L_5\|x - x_*\| \tag{5.5}$$

$$\|g_y(y, t)f_t(x, t) - g_y(y_*, t)f_t(x_*, t)\| \leq L_6\|x - x_*\| \tag{5.6}$$

$$\|g_t(y, t)f_x(x, t) - g_t(y_*, t)f_x(x_*, t)\| \leq L_7\|x - x_*\|, \tag{5.7}$$

3. *$g_y(y, t)f_x(x, t) - I$ is nonsingular, and the Jacobian $g_y(y, t)f_x(x, t)$ satisfies*

$$\delta_k = \|g_y(y_k, t)f_x(x_k, t)\| \leq \delta < 1 \tag{5.8}$$

with

$$\delta_* = \lim_{k\to\infty} \delta_k \leq \delta;$$

then (x_k, y_k) *converges to* (x_*, y_*), $(x_k^{'}, y_k^{'})$ *converges to* $(x_*^{'}, y_*^{'})$ *and the rate of convergence is bounded by the rate at which* $(x_k, y_k) \longrightarrow (x_*, y_*)$.

Proof. Construct the following implicit function

$$\begin{aligned} F(x,t) &= g(y,t) - x \\ &= g(f(x,t),t) - x. \end{aligned} \tag{5.9}$$

Therefore

$$F_x(x,t) = g_y(y,t) f_x(x,t) - I, \tag{5.10}$$

$$F(x_*, t) = 0 \quad \text{and} \tag{5.11}$$

$$F_x(x_*, t) x_*^{'} + F_t(x_*, t) = 0. \tag{5.12}$$

From the iterative process we have updates

$$\begin{aligned} x_{k+1} &= x_k + g(f(x_k,t),t) - x_k \\ &= x_k + F(x_k, t) \end{aligned} \tag{5.13}$$

and

$$x_{k+1}^{'} = x_k^{'} + F_x(x_k,t) x_k^{'} + F_t(x_k,t). \tag{5.14}$$

Subtract x_* from both sides of (5.13)

$$\begin{aligned} x_{k+1} - xk &= (x_k - x_*)(I + F_x(x_k,t)) - F_x(x_k,t)(x_k - x_*) + F(x_k,t) \\ &= (I + F_x(x_k,t))(x_k - x_*) + r_k \end{aligned} \tag{5.15}$$

where the residual term

$$\begin{aligned} r_k &= F(x_k,t) - F_x(x_k,t)(x_k - x_*) \\ &= F(x_k,t) - F(x_*,t) - F_x(x_k,t)(x_k - x_*) \\ &= (F_x(x_*,t) - F_x(x_k,t))(x_k - x_*) + O\|x_k - x_*\|^2 \qquad (5.16) \end{aligned}$$

We know from (5.5)

$$\|F_x(x_*,t) - F_x(x_k,t)\| = \|f_y(y_*,t)g_x(x_*,t) - f_y(y_k,t)g_x(x_k,t)\| \leq L_5\|x_k - x_*\|$$

thus

$$r_k \equiv \mathcal{O}\|x_k - x_*\|^2.$$

From (5.8)

$$\|I + F_x(x_k,t)\| = \|I + f_y(y_k,t)g_x(x_k,t) - I\| \leq \delta < 1,$$

so

$$\|x_{k+1} - x_*\| \leq \delta\|x_k - x_*\| + \mathcal{O}\|x_k - x_*\|^2.$$

Therefore x_k converges to x_*. Similar argument can be used for y_k by constructing $F(y,t) = f(g(y,t),t) - y$ to get $y_k \longrightarrow y_*$.

Similarly, subtracting $x_*^{'}$ from both sides of (5.14)

$$\begin{aligned} x_{k+1}^{'} - xk^{'} &= x_k^{'} - x_*^{'} + F_x(x_k,t)x_k^{'} + F_t(x_k,t) \\ &\quad -F_x(x_k,t)x_*^{'} + F_x(x_k,t)x_*^{'} \\ &= (I + F_x(x_k,t))(x_k^{'} - x_*^{'}) + F_x(x_k,t)x_*^{'} + F_t(x_k,t) \\ &= (I + F_x(x_k,t))(x_k^{'} - x_*^{'}) + r_k \qquad (5.17) \end{aligned}$$

where the residual term

$$\begin{aligned} r_k &= F_x(x_k,t)x_*^{'} + F_t(x_k,t) \\ &= F_x(x_k,t)(-F_x(x_*,t)^{-1}F_t(x_*,t) + F_t(x_k,t) \\ &\quad -F_t(x_*,t) + F_t(x_*,t) \\ &= (F_x(x_*,t) - F_x(x_k,t))F_x(x_*,t)^{-1}F_t(x_*,t) + (F_t(x_k,t) - F_t(x_*,t)) \end{aligned} \tag{5.18}$$

We know from (5.5) and Condition (2)

$$\begin{aligned} \|F_x(x_*,t) - F_x(x_k,t)\| &= \|f_y(y_*,t)g_x(x_*,t) - f_y(y_k,t)g_x(x_k,t)\| \leq L_5\|x_k - x_*\| \\ \|F_t(x_k,t) - F_t(x_*,t)\| &= \|g_y(y_k,t)f_t(x_k,t) + g_t(y_k,t) - g_y(y_*,t)f_t(x_*,t) - g_t(y_*,t)\| \\ &= \leq (L_6 + L_4)\|x_k - x_*\| \end{aligned}$$

and $F_x(x_*,t)^{-1}F_t(x_*,t)$ is bounded because of continuous differentiability, thus

$$r_k \equiv \mathcal{O}\|x_k - x_*\|.$$

Using (5.8)

$$\|x_{k+1}^{'} - x_*^{'}\| \leq \delta\|x_k^{'} - x_*^{'}\| + \mathcal{O}\|x_k - x_*\|.$$

Therefore $x_k^{'} \longrightarrow x_*^{'}$ at a rate bounded by that of $x_k \longrightarrow x_*$. Similar argument can be used for $y_k^{'}$ by constructing $F(y,t) = f(g(y,t),t) - y$ to get $y_k^{'} \longrightarrow y_*^{'}$.

Q. E. D.

Note that the iteration process we consider here is a special case of the general implicit function $F(x,t) = 0$, computed using update

$$x_{k+1} = x_k - P_k F(x_k,t)$$

with $P_k = -I$.

Separate stopping criteria are needed since the derivatives often do not converge as fast as function evaluation. Here we include some of the results from [52] and [53] for completeness. In [52] the following result was presented. For $F(x,t) = 0$, denote

$$\rho_k \equiv \|x_k - x_*\| \quad \text{and} \quad \mu_k \equiv \|x_k^{'} - x_*^{'}\| \tag{5.19}$$

and set

$$\eta_k \equiv (Lc_1 + \|P_k^{'}\|)\rho_k \quad \text{with} \quad c_1 \equiv 2(c_0^2 + 1), \tag{5.20}$$

where L and c_0 are constants, it can be shown that

$$\mu_k \leq \frac{1}{(1-\delta)}\|P_k(F_x(x_k,t)x_k^{'} + F_x(x_k,t))\| + \frac{1}{2}Lc_0c_1\rho_k, \tag{5.21}$$

$$\mu_{k+1} \leq \delta_k\mu_k + c_0\eta_k, \quad mboxand \;\; \mathcal{O}(\|x_k - x_*\|) \leq c_1c_0L\rho_k, \tag{5.22}$$

for all $\rho_k < \rho$. This provides us with a stopping criterion for the derivative iteration if we can make some reasonable estimates on L, c_0 and δ.

In [53], a stopping criterion is provided in terms of the desired accuracy of $\xi\|r\|$, where $\xi < 1$ and r is a fixed arbitrary row vector:

$$\|x_{k+1} - x_k\| < \frac{\xi(1-\tau)^3}{2C(1-\tau_k)}, \tag{5.23}$$

and

$$\|x_k^{'} + r - x_{k+1}^{'}\| < \frac{\xi(1-\tau)}{2k} \cdot \|r\|, \tag{5.24}$$

where τ and k are bounds for $\|\Phi_y\|$ and $\|\Phi_u\|$, respectively, in a neighborhood of $(y_*(u), u)$, and C is the Lipschitz constant for the map Φ' in this neighborhood.

5.3 Numerical Experiments

The independent variable t in the iteration process discussed in the last section usually represents system related design parameters. Naturally we are interested in $\frac{\partial z}{\partial t}$, the sensitivity of the results of the performance model with respect to certain network design parameters. In this section we examine the sensitivity of blocking probability with respect to offered traffic load, and with respect to link capacities. Eventually we would like to solve the following optimization problem:

$$\min f(B(x,y)), \quad \text{such that} \quad g(x) \geq 0,$$

where $f(\cdot)$ is some cost/penalty function and $g(\cdot)$ is the restriction on network designs. Details on this formulation are given in the next chapter.

In the first experiment, the AD package ADIC [54] is chosen to generate derivative code to calculate $\frac{\partial z}{\partial t}$, with t being the offered traffic load. There was no separate stopping criterion generated for derivative iteration. The whole iteration is terminated when the function convergence criterion is satisfied. So stopping criterion for derivatives had to be manually added.

We used the first example from Chapter 2, a five-node fully connected network, with three different classes of traffic. The computations are quite satisfying, and correspond to previous discussions, B'_x did converge but was slower than B itself, 18 iterations vs. 11 iterations. Some of the results are shown in Table 5.1.

The second example is not quite as successful, and this relates to one of ADIC's limitations. We want to compute the sensitivities of connection blocking with respect to link capacities, i.e., $\partial f/\partial C$. Link capacities (C) are defined as integers throughout this thesis. In order to compute this we extend the definition of f to non-integer values of C by linear interpolation, and at integer values define the derivatives w.r.t C to be the left derivative. So what we really want is

$$\frac{\partial f}{\partial C} = f(C) - f(C-1).$$

However ADIC does not do this automatically. In general ADIC only treats variables of floating point type to be potential independent variables, and integer type variables are left unprocessed. On the other hand simply declaring C to be a "double" does not help because we will have to type cast it back to "int" when it comes to operations such as summation and factorial. In this case additional fixing is needed, which makes the process less "automatic". This experiment is used in the next chapter for designing link capacities/resource allocation.

5.4 Conclusion

This chapter focuses on the application of automatic differentiation techniques on iterative processes. Conditions for derivative convergence for the fixed point problems were derived and results on additional stopping criteria were discussed. Our

Node Pair	Class	Blocking Prob.	Sensitivities to offered load		
(i,j)	s	B	$\frac{\partial B}{\partial \lambda_1}$	$\frac{\partial B}{\partial \lambda_2}$	$\frac{\partial B}{\partial \lambda_3}$
(0,1)	0	0.223711	5.664913e-04	8.852376e-04	1.034676e-03
(0,1)	1	0.398828	8.714347e-04	1.361373e-03	1.591602e-03
(0,1)	2	0.535275	1.002060e-03	1.565485e-03	1.831259e-03
(0,2)	0	0.225985	4.512100e-04	6.921243e-04	7.937614e-04
(0,2)	1	0.402440	6.913813e-04	1.055653e-03	1.204807e-03
(0,2)	2	0.539571	7.917958e-04	1.203237e-03	1.366365e-03
(0,3)	0	0.223627	4.350501e-04	6.667808e-04	7.638223e-04
(0,3)	1	0.398999	6.685918e-04	1.018729e-03	1.159727e-03
(0,3)	2	0.535853	7.677732e-04	1.162738e-03	1.315044e-03
(2,3)	0	0.224485	3.323044e-04	5.046990e-04	5.711664e-04
(2,3)	1	0.400556	4.949528e-04	7.496102e-04	8.462674e-04
(2,3)	2	0.537943	5.485987e-04	8.286788e-04	9.334801e-04
(2,4)	0	0.222634	3.320084e-04	5.023101e-04	5.662438e-04
(2,4)	1	0.397356	4.977124e-04	7.500698e-04	8.423990e-04
(2,4)	2	0.533818	5.558128e-04	8.343709e-04	9.336176e-04
(3,4)	0	0.218676	2.870769e-04	4.387947e-04	5.000116e-04
(3,4)	1	0.390262	4.199866e-04	6.409087e-04	7.295093e-04
(3,4)	2	0.524390	4.558871e-04	6.948347e-04	7.903890e-04

Table 5.1: Sensitivity Computation

experiments show that this is a valid and useful technology for network sensitivity analysis, and can further be used toward solving network design problems.

Chapter 6

NETWORK DESIGN USING PERFORMANCE MODEL AND NUMERICAL OPTIMIZATION PACKAGES

Our approach for network design, optimization and dimensioning is to use existing numerical tools. Our reason for this is two-folded: One, an analytical approximation algorithm can be easily linked to mathematical programming tools to get network performance optimization and trade-off analysis. Secondly, we aim at developing a systematic way of network design and optimization, which is relatively independent of the underlying detailed performance model. We believe that by using properly selected mathematical programming tools we can achieve this.

There are potentially many numerical tools that we could use. For the examples we present in this chapter we used CONSOL-OPTCAD and CFSQP. Our choice of these two for this research is based on the following. CONSOL-OPTCAD and CFSQP are both developed at the University of Maryland by groups led by Professor

André L. Tits and are freely available. We had considered using OPL (Optimization Programming Language) and its industrial implementation the OPL Studio from ILOG, Inc.. Unfortunately OPL is primarily targeted at linear programming (e.g., CPLEX), integer programming and combinatorial optimization problems, which made it a less favorable choice for our purposes since our model is nonlinear. Same considerations apply to languages like AMPL and GAMS. On the other hand, CONSOL-OPTCAD is a tool for optimization-based design of a large class of systems, which include both linear and nonlinear problems. More importantly, it evaluates the performance of instances of the system under consideration and allows parameters to take on any real value in a given range so that they can be optimally adjusted, and is thus very good for trade-off analysis. CFSQP (C code for Feasible Sequential Quadratic Programming) is a set of C functions for solving large scale constrained nonlinear minimax optimization problems, generating iterates satisfying all inequality constraints. It takes user provided objective and constraint function code. It provides default finite differencing for gradient evaluation of these functions but can also take user provided gradient evaluation code for objective and constraint functions. This last feature makes it a very nice package to be used along with ADIC which generates gradient evaluation code as we described in the last chapter (ADIC is also freely available from Argone National Lab).

In the next section we use the reduced load approximation method with CONSOL-OPTCAD [55] for trunk reservation parameter design. In the second application example we take the reduced load model along with its derivative code generated by ADIC and use CFSQP [56] for link capacity design.

6.1 Application One: Design of Trunk Reservation Parameters

As a way of call admission control, trunk reservation regulates individual classes of traffic as well as their inter-relationship. The combination of trunk reservation parameters $\{r_1, r_2, ..., r_S\}$ for each class of traffic s could potentially affect average blocking probability and the total carried traffic by the network. In this example we choose the objective to be the weighted average of blocking probabilities, and the constraints to be the bounds on the blocking probability of individual classes of traffic.

Following our previous notations, this design problem is formulated as follows:

Design parameters: $r_1, r_2, ..., r_S$.

$$\min \quad \frac{\sum_{(r,s)} \lambda_{rs} \cdot B_{rs}}{\sum_{(r,s)} \lambda_{rs}} \tag{6.1}$$

S. t.

$$B_{rs} < \text{bound}_{rs}, \quad \text{all} \quad (r, s)$$

where λ_{rs} is the offered traffic of class s on route r.

In CONSOL-OPTCAD, we can provide two values, namely the good value and bad value, for each $bound_{k,l}(s)$, indicating our level of satisfaction. Trade-off analysis can be carried out between minimizing the objective function value and satisfying the constraints, and thus decide the combination of $\{r_1, r_2, ..., r_S\}$.

By applying this formula to the first example of fully connected network in Chapter 2, in the case with trunk reservation admission control, and restricting the trunk reservation parameters to be less or equal to 5, we get the trade-off between the blocking probability of each class vs. the weighted average blocking probability, which is shown in Table 6.1 (Numbers in italic are individual minimum).

Weighted Average	B_1	B_2	B_3	(r_1, r_2, r_3)
0.265702	*0.036159*	0.580163	0.803230	(1,4,5)
0.273409	0.062937	0.526852	0.840663	(1,3,5)
0.290184	0.067968	0.562512	0.792359	(2,4,5)
0.292510	0.112452	0.467300	0.861320	(1,2,5)
0.307262	0.116759	0.485502	0.819291	(1,2,4)
0.309973	0.119625	0.492353	0.824259	(2,3,5)
0.420363	0.571205	*0.284030*	0.565232	(4,1,2)
0.406413	0.319234	0.671934	*0.322844*	(3,5,1)

Table 6.1: Trunk Reservation Parameter Design

Because of the topology symmetry of this example, the same class of traffic encounters approximately the same blocking probability regardless of their source-destination node pair and input rate, although their input rates are counted in calculating the weighted average blocking probability. So the numbers displayed here are only distinguished by their classes but not their associated source-destination node pairs.

As we can see from Table 6.1, the weighted average blocking probability and the blocking probability of class-1 type of traffic achieve their optimum at the same time with trunk reservation parameter choice of 1,4 and 5. The reason is obvious: since class-1 has much smaller trunk reservation requirement than class-2 and 3, together with its lowest bandwidth requirement, it has the highest priority and chances of being admitted into the network. On the other hand, class-2 and 3 are being jeopardized by their high trunk reservation requirement and also high bandwidth requirement. We may also conclude that class-1 type of traffic occupies the greatest amount of the total traffic throughput since the weighted average blocking probability is the lowest while blocking probability of class-1 is the lowest. Class-2 and 3 types of traffic achieve their minimum separately when their trunk reservation parameters are 1 and others' are higher.

6.2 Application Two: Design of Link Capacities

Similar to the first application, link capacities are another set of parameters which is critical in network design. It is natural to expect that we should assign higher capacities to more frequently congested links, and save capacities on rarely used ones. In this example we define the objective as the total carried traffic to be maximized, and the constraints as the total available resource/bandwidth and the capacity bounds on each link. Note that this objective is equivalent to that of the first example (6.1), but gives a different objective value in terms of throughput rather than blocking probability. We use the hierarchical network presented in Chapter 3 for this example with same offered traffic. The network has 30 links.

Design parameters: $\{C_j\}$, all j

$$\begin{aligned} \max \quad & \sum_{(r,s)} \lambda_{rs}(1 - B_{rs}) \quad \text{or} \\ \min \quad & \sum_{(r,s)} \lambda_{rs} B_{rs} \end{aligned}$$

S. t.

$$\begin{aligned} & \sum_j C_j \leq 3700; \\ & 0 \leq C_j \leq 1000 \quad \text{for all} \quad j. \end{aligned}$$

We use CFSQP to solve this problem. The objective function we provide is the reduced load model, and the gradient function for the objective is generated by ADIC with necessary modification as discussed in Chapter 5. We compare the results as follows. Table 6.2 is a random allocation of link capacities similar to what we used for numerical experiments presented in Chapter 5. It is a slight variation of the table given in the Appendix. Tables 6.3 through Table 6.8 are optimization results on link capacities with the traffic load weight being 3, 5 and 7 respectively, and comparison of total carried traffic between optimal and non-optimal capacity allocations. Original results are in floating point type and we present rounded-off numbers here. Entries in Italics indicate cross-group links.

Link (i, j)	$C_{i,j}$	Link (i, j)	$C_{i,j}$	Link (i, j)	$C_{i,j}$
(0,1)	140	(1,2)	120	(1,3)	120
(0,2)	120	(2,4)	100	(3,4)	100
(3,6)	100	(0,6)	100	(4,5)	100
(5,6)	120	(6,7)	120	(5,13)	100
(4,20)	120	(7,8)	120	(7,10)	120
(9,12)	120	(11,12)	140	(8,9)	140
(9,10)	100	(10,11)	140	(12,13)	160
(13,14)	100	(14,15)	140	(14,18)	120
(16,19)	100	(16,18)	160	(15,20)	160
(17,20)	160	(17,18)	120	(19,20)	140

Table 6.2: Link Capacities Used for Comparison with Optimal Allocation

Link (i,j)	$C_{i,j}$	Link (i,j)	$C_{i,j}$	Link (i,j)	$C_{i,j}$
(0,1)	0	(1,2)	111	(1,3)	111
(0,2)	111	(2,4)	138	(3,4)	103
(3,6)	165	(0,6)	111	(4,5)	111
(5,6)	155	(6,7)	211	(5,13)	152
(4,20)	161	(7,8)	111	(7,10)	111
(9,12)	111	(11,12)	131	(8,9)	131
(9,10)	91	(10,11)	131	(12,13)	172
(13,14)	186	(14,15)	131	(14,18)	111
(16,19)	71	(16,18)	71	(15,20)	131
(17,20)	131	(17,18)	111	(19,20)	111

Table 6.3: Optimized Link Capacities with Weight = 3.

	Optimal	Non-optimal
Total Carried Traffic	534.3	501.49767
Total Capacities Allocated	3683	3700

Table 6.4: Throughput Comparison with Weight = 3.

Link (i,j)	$C_{i,j}$	Link (i,j)	$C_{i,j}$	Link (i,j)	$C_{i,j}$
(0,1)	0	(1,2)	67	(1,3)	92
(0,2)	101	(2,4)	184	(3,4)	83
(3,6)	179	(0,6)	106	(4,5)	148
(5,6)	200	(6,7)	279	(5,13)	207
(4,20)	220	(7,8)	101	(7,10)	132
(9,12)	107	(11,12)	90	(8,9)	88
(9,10)	47	(10,11)	86	(12,13)	235
(13,14)	218	(14,15)	122	(14,18)	103
(16,19)	29	(16,18)	38	(15,20)	95
(17,20)	127	(17,18)	102	(19,20)	103

Table 6.5: Optimized Link Capacities with Weight = 5.

	Optimal	Non-optimal
Total Carried Traffic	889.48874	633.50409
Total Capacities Allocated	3689	3700

Table 6.6: Throughput Comparison with Weight = 5.

Link (i,j)	$C_{i,j}$	Link (i,j)	$C_{i,j}$	Link (i,j)	$C_{i,j}$
(0,1)	0	(1,2)	75	(1,3)	106
(0,2)	81	(2,4)	153	(3,4)	72
(3,6)	193	(0,6)	94	(4,5)	110
(5,6)	193	*(6,7)*	*302*	*(5,13)*	*191*
(4,20)	*245*	(7,8)	118	(7,10)	149
(9,12)	108	(11,12)	96	(8,9)	97
(9,10)	55	(10,11)	95	*(12,13)*	*244*
(13,14)	230	(14,15)	114	(14,18)	108
(16,19)	35	(16,18)	36	(15,20)	99
(17,20)	116	(17,18)	92	(19,20)	92

Table 6.7: Optimized Link Capacities with Weight = 7.

	Optimal	Non-optimal
Total Carried Traffic	1074.2879	728.947842
Total Capacities Allocated	3699	3700

Table 6.8: Throughput Comparison with Weight = 7.

6.3 Conclusion

We have shown that the proposed fixed-point approximation scheme can be applied to optimization tools to get trunk reservation parameter design and link capacity design analysis. We can also have various problem formulations according to different optimization purposes, and this scheme is expected to be ready to be applied.

Chapter 7

CONCLUSIONS AND FUTURE WORK

In this thesis we presented our work on the development of connection level mathematical models used for estimating network performance characteristics including throughput, delay and blocking probability. These models are used for data networks using QoS routing policies. We described fast (two to three orders of magnitude faster than discrete event simulation) approximation algorithms, which we have developed for accurately estimating blocking probability in a random topology network, using state-dependent routing, with multiple classes of traffic. We described even faster algorithms based on a hierarchical loss network model we have developed. The latter are well matched to networks that have a natural hierarchical architecture, or which use some form of hierarchical routing to further reduce computational cost. We also developed models for networks using delay-based QoS routing. We then showed how typical design and dimensioning problems can be formulated as a multi-objective constrained optimization problem, using performance estimation network models, and we described our research in the development of a general network design and dimensioning methodology by linking our performance models with Automatic Differentiation and multi-objective optimization algorithms and tools. We examined the applicability of automatic differentiation under the scenario of our interest. We presented examples and applications that demonstrate the speed and versatility of our methodology and algorithms.

Future work remains in the following areas:

- *Initial assumptions.* As we have pointed out in the Introduction chapter, Poisson arrival is the fundamental assumption underlying the Erlang formula. Various later works on single link analysis are also based on this assumption. One important property of the Erlang formula is that it is insensitive to the distribution of the duration of a call. In other words, the distribution of the call duration only shows through its mean value in the blocking probability. Recent statistical analysis of measured network traffic have revealed long holding times (i.e., consistent with infinite variance) not only in data networks, but also in today's voice networks. While the insensitivity property is a very useful one in this scenario, it only applies in "steady state" and it could take infinitely long for a system to reach steady state with the presence of long holding times. It would be interesting to find out how valid the traditional steady state study is, if it still holds, in terms of blocking probabilities and performance analysis.

- *Asymptotic analysis.* We need further asymptotic analysis for the hierarchical model presented in Chapter 3. In addition to asymptotic analysis, it is also of interest to us to investigate if an approximation scheme can generate certain performance bounds. Both are important because they show how valid and useful a model is, under what conditions or within what range is the model most accurate. Simulation is often used to compare with an analytical model. We believe asymptotic analysis and performance bounds are more comprehensive and rigorous ways of assessing a model.

- *Qualitative trade-off study.* We need further numerical experiments for both the non-hierarchical and hierarchical model for validation purposes. Based on the performance and design model we developed, we are interested in extensive trade-off studies, as well as the property of the problem, e.g., convexity, feasible regions, etc.. Trade-off study helps in answering many realistic questions. For example, for a single connection, what is the trade-off between delay and bandwidth requirement? Is there a region where both metrics can be improved, or reduction in one inevitably results in increase in the other? Does increase in the trunk reservation parameter for a certain class of traffic benefit the overall network performance in terms of throughput, and within what range? Given existing network size and configuration, as well as the number of subscribers, what is the range within which it is profitable to increase network size or increase subscription or both? We hope we will be able to give insights to these problems through our study. It's also important to investigate the trade-offs involving networks' ability to handle peak demand or worst case scenario.

Appendix A

TRAFFIC PARAMETERS USED IN EXAMPLES

Node Pair (k,l)	Class s	$\lambda_{k,l}(s)$	Node Pair (k,l)	Class s	$\lambda_{k,l}(s)$
$(0,1)$	1	20.0	$(1,3)$	1	30.0
	2	15.0		2	7.0
	3	10.0		3	17.0
$(0,2)$	1	5.0	$(1,4)$	1	3.0
	2	38.0		2	20.0
	3	9.0		3	20.0
$(0,3)$	1	16.0	$(2,3)$	1	0.0
	2	17.0		2	15.0
	3	16.0		3	20.0
$(0,4)$	1	6.0	$(2,4)$	1	15.0
	2	0.0		2	15.0
	3	32.0		3	20.0
$(1,2)$	1	37.0	$(3,4)$	1	51.0
	2	20.0		2	26.0
	3	5.0		3	0.0

Table A.1: Traffic Rates Used in Example One, Chapter Two

Link (i, j)	$C_{i,j}$	Link (i, j)	$C_{i,j}$	Link (i, j)	$C_{i,j}$
(0,1)	180	(0,2)	120	(0,3)	120
(0,4)	120	(0,5)	180	(0,9)	120
(0,10)	180	(0,11)	180	(0,14)	180
(1,2)	120	(1,4)	120	(1,5)	120
(1,8)	120	(1,9)	120	(1,10)	120
(1,11)	120	(2,5)	120	(2,11)	120
(2,15)	120	(3,10)	120	(5,8)	60
(6,7)	120	(6,13)	120	(6,14)	120
(7,14)	120	(8,10)	120	(9,10)	120
(9,11)	60	(10,11)	120	(11,12)	60
(13,14)	120				

Table A.2: Link Capacities Used in Example Two, Chapter Two

Node Pair (k,l)	$\lambda_{k,l}(1)$	Node Pair (k,l)	$\lambda_{k,l}(1)$	Node Pair (k,l)	$\lambda_{k,l}(1)$
(1,0)	78.65	(2,0)	0.37	(2,1)	11.19
(3,0)	0.96	(5,0)	0.28	(5,1)	6.33
(5,3)	0.001	(5,4)	0.13	(6,0)	0.37
(6,1)	9.82	(6,5)	0.37	(7,1)	0.001
(8,0)	0.01	(8,1)	0.37	(8,2)	0.0001
(8,4)	0.0008	(9,0)	0.38	(9,1)	1.12
(9,2)	0.37	(9,5)	0.37	(9,6)	0.78
(9,8)	0.37	(10,0)	0.06	(10,1)	0.37
(11,1)	1.12	(11,2)	0.007	(11,6)	0.01
(11,8)	0.38	(13,0)	1.31	(13,1)	0.38
(13,2)	0.37	(13,6)	0.0003	(13,8)	0.37
(13,9)	0.75	(13,10)	0.008	(13,11)	0.0001
(14,0)	0.001	(14,1)	0.75	(14,5)	0.07
(14,6)	0.75	(14,13)	0.0003	(15,0)	0.37
(15,1)	15.61	(15,2)	0.37	(15,3)	0.37
(15,4)	0.37	(15,6)	0.37	(15,9)	0.37
(15,13)	0.37				

Table A.3: Arrival rates for class 1 calls, Example Two, Chapter Two.

Node Pair (k,l)	$\lambda_{k,l}(2)$	Node Pair (k,l)	$\lambda_{k,l}(2)$	Node Pair (k,l)	$\lambda_{k,l}(2)$
(1,0)	5.97	(2,0)	6.35	(2,1)	3.36
(3,0)	5.97	(3,1)	0.37	(4,0)	1.49
(4,1)	0.37	(4,2)	0.75	(5,0)	28.90
(5,1)	2.24	(5,2)	22.40	(5,3)	2.24
(5,4)	8.96	(6,0)	1.49	(6,1)	1.49
(6,2)	0.75	(6,3)	0.008	(6,5)	3.73
(8,1)	2.24	(8,2)	1.49	(8,5)	5.97
(8,6)	1.50	(9,0)	4.11	(9,1)	9.71
(9,2)	1.87	(9,3)	0.37	(9,4)	0.37
(9,5)	3.37	(9,6)	3.36	(9,8)	4.48

Table A.4: Arrival rates for class 2 calls, Example Two, Chapter Two, Part I.

Node Pair (k,l)	$\lambda_{k,l}(2)$	Node Pair (k,l)	$\lambda_{k,l}(2)$	Node Pair (k,l)	$\lambda_{k,l}(2)$
(10,0)	5.99	(10,1)	2.24	(10,2)	0.75
(10,3)	0.07	(10,5)	0.09	(10,6)	0.75
(10,9)	8.96	(11,0)	2.24	(11,1)	5.97
(11,2)	1.49	(11,4)	0.75	(11,5)	2.24
(11,6)	0.75	(11,9)	6.35	(13,0)	1.49
(13,1)	1.87	(13,2)	0.37	(13,4)	0.75
(13,6)	0.08	(13,10)	0.75	(14,0)	1.12
(14,1)	1.12	(14,2)	1.49	(14,5)	8.21
(14,6)	0.75	(14,9)	2.24	(14,10)	1.12
(14,13)	1.12	(15,0)	1.49	(15,1)	0.75
(15,2)	3.14	(15,3)	0.75	(15,4)	0.75
(15,6)	0.75	(15,13)	0.75		

Table A.5: Arrival rates for class 2 calls, Example Two, Chapter Two, Part II.

Node Pair (k,l)	$\lambda_{k,l}(3)$	Node Pair (k,l)	$\lambda_{k,l}(3)$	Node Pair (k,l)	$\lambda_{k,l}(3)$
(5,1)	0.37	(6,1)	0.37	(6,5)	0.0005
(9,0)	0.37	(10,1)	0.37	(13,6)	0.37
(14,1)	0.0003				

Table A.6: Arrival rates for class 3 calls, Example Two, Chapter Two.

Node Pair (k,l)	$\lambda_{k,l}(4)$	Node Pair (k,l)	$\lambda_{k,l}(4)$	Node Pair (k,l)	$\lambda_{k,l}(4)$
(1,0)	1.28	(2,0)	0.22	(2,1)	1.60
(3,0)	0.38	(3,1)	1.12	(3,2)	0.75
(4,0)	2.57	(4,1)	2.60	(4,2)	0.37
(5,0)	1.75	(5,1)	1.18	(5,2)	0.13
(5,4)	0.07	(6,0)	1.50	(6,1)	0.52
(6,2)	0.46	(6,3)	0.01	(6,5)	0.01
(7,0)	0.002	(7,2)	0.002	(7,3)	0.001
(7,4)	0.002	(7,6)	0.0001	(8,0)	0.001
(8,1)	0.26	(8,5)	0.008	(9,1)	0.37
(9,3)	0.37	(9,5)	0.003	(10,0)	4.22

Table A.7: Arrival rates for class 4 calls, Example Two, Chapter Two, Part I.

Node Pair (k,l)	$\lambda_{k,l}(4)$	Node Pair (k,l)	$\lambda_{k,l}(4)$	Node Pair (k,l)	$\lambda_{k,l}(4)$
(10,1)	2.20	(10,2)	0.98	(10,3)	1.07
(10,4)	0.40	(10,5)	1.22	(10,6)	1.62
(10,8)	1.65	(10,9)	0.37	(11,0)	0.76
(11,1)	2.57	(11,2)	1.19	(11,3)	1.12
(11,4)	1.50	(11,6)	0.01	(11,8)	0.37
(11,10)	3.30	(13,0)	1.23	(13,1)	2.24
(13,2)	0.43	(13,3)	0.37	(13,5)	0.50
(13,6)	1.03	(13,7)	0.38	(13,9)	0.37
(13,10)	1.26	(14,0)	1.35	(14,1)	0.07
(14,2)	0.38	(14,5)	0.46	(14,6)	0.41
(14,9)	0.37	(14,10)	0.70	(14,11)	0.75
(14,13)	0.29	(15,1)	0.75	(15,8)	0.37
(15,9)	0.0004	(15,10)	0.38	(15,11)	0.0001

Table A.8: Arrival rates for class 4 calls, Example Two, Chapter Two, Part II.

Link (i,j)	$C_{i,j}$	Link (i,j)	$C_{i,j}$
(0,1)	120	(1,2)	120
(1,3)	120	(0,2)	120
(2,4)	80	(3,4)	60
(3,6)	120	(0,6)	120
(4,5)	120	(5,6)	120
(6,7)	160	(5,13)	160
(4,20)	160	(7,8)	120
(7,10)	120	(9,12)	120
(11,12)	140	(8,9)	140
(9,10)	100	(10,11)	140
(12,13)	160	(13,14)	100
(14,15)	140	(14,18)	120
(16,19)	80	(16,18)	80
(15,20)	140	(17,20)	140
(17,18)	120	(19,20)	120

Table A.9: Link Capacities Used in the Hierarhical Network, Chapter Three.

Node Pair r	λ_r	Node Pair r	λ_r	Node Pair r	λ_r
(0,2)	0.2	(0,3)	2.1	(0,5)	1.3
(0,10)	2.3	(0,12)	1.9	(0,13)	2.8
(0,14)	2.2	(0,16)	1.1	(0,18)	2.3
(0,19)	1.7	(0,20)	3.3	(1,4)	0.5
(1,5)	1.3	(1,6)	2.3	(1,7)	7.8
(1,8)	0.2	(1,9)	0.7	(1,10)	0.1
(1,15)	0.1	(1,17)	4.6	(1,20)	3.3
(2,3)	0.7	(2,5)	0.4	(2,8)	5.1
(2,11)	2.3	(2,12)	0.5	(2,13)	0.5
(2,15)	0.5	(2,18)	2.2	(3,5)	1.6
(3,6)	0.9	(3,7)	1.2	(3,9)	3.3
(3,10)	1.4	(3,11)	0.1	(3,12)	0.2
(3,13)	0.5	(3,16)	1.2	(3,17)	0.7
(3,19)	0.5	(3,20)	1.4	(4,5)	1.1
(4,6)	1.6	(4,7)	0.1	(4,8)	0.1

Table A.10: Arrival Rates Between Node Pairs Used in the Hierarchical Network, Chapter Three, Part I.

Node Pair r	λ_r	Node Pair r	λ_r	Node Pair r	λ_r
(4,9)	4.1	(4,10)	5.2	(4,11)	1.1
(4,14)	0.3	(4,15)	1.0	(4,18)	2.3
(4,20)	0.5	(5,7)	0.2	(5,8)	1.4
(5,9)	0.7	(5,10)	2.5	(5,11)	1.8
(5,12)	0.3	(5,13)	0.1	(5,16)	1.2
(5,17)	2.4	(5,19)	0.9	(6,10)	3.3
(6,11)	4.0	(6,12)	2.1	(6.13)	2.5
(6,15)	1.2	(6,16)	0.1	(6.17)	0.3
(6,18)	1.3	(6,19)	0.2	(6,20)	2.5
(7,9)	2.3	(7,10)	0.1	(7,11)	0.4
(7,15)	4.1	(7,17)	0.2	(7,18)	0.2
(7,19)	0.8	(7,20)	0.5	(8,10)	0.1
(8,12)	3.2	(8,13)	0.1	(8,14)	2.6
(8,19)	0.5	(9,11)	1.2	(9,13)	2.3
(9,14)	0.7	(9,15)	0.7	(9,16)	0.3

Table A.11: Arrival Rates Between Node Pairs Used in the Hierarchical Network, Chapter Three, Part II.

Node Pair r	λ_r	Node Pair r	λ_r	Node Pair r	λ_r
(9,17)	0.1	(9,18)	0.5	(9,19)	0.2
(9,20)	1.2	(10,11)	0.3	(10,12)	0.1
(10,13)	0.5	(10,14)	0.5	(10,15)	2.1
(10,16)	0.1	(10,17)	0.3	(10,20)	0.5
(11,12)	1.4	(11,13)	0.1	(11,18)	1.5
(11,19)	1.3	(11,20)	0.4	(12,13)	1.4
(12,15)	1.6	(12,16)	0.4	(12,17)	2.1
(12,18)	0.8	(12,19)	0.6	(12,20)	1.0
(13,14)	3.9	(13,15)	0.1	(13,16)	0.1
(13,17)	0.2	(13,19)	0.5	(13,20)	1.5
(14,15)	0.2	(14,16)	1.3	(14,17)	2.5
(14,18)	2.1	(14,19)	1.4	(14,20)	0.3
(15,18)	0.1	(15,19)	0.8	(15,20)	3.3
(16,17)	0.2	(16,18)	0.7	(16,19)	0.2
(16,20)	0.5	(17,18)	0.3	(17,20)	2.1
(18,19)	1.2	(18,20)	1.0	(19,20)	0.7

Table A.12: Arrival Rates Between Node Pairs Used in the Hierarchical Network, Chapter Three, Part III.

BIBLIOGRAPHY

[1] F. P. Kelly. Notes on effective bandwidth. In *Stochastic Networks Theory and Applications*, F. P. Kelly, S. Zachary and I. Ziedins, pages 141–168. Eds. Oxford, U. K.: Science, 1996.

[2] A. Ephremides and B. Hajek. Information Theory and Communication Network: An Unconsummated Union. *IEEE Trans. Information Theory*, 44(6):2416–2434, 1998.

[3] André Girard. *Routing and Dimensioning in Circuit-Switched Networks.* Addison-Wesley, 1st edition, 1990.

[4] F. P. Kelly. Routing In Circuit-Switched Networks: Optimization, Shadow Prices and Decentralization. *Adv. Appl. Prob.*, 20:112–144, 1988.

[5] A. Girard and B. Liau. Dimensioning of Adaptively Routed Networks. *IEEE Trans. Networking*, 1(4):460–468, 1993.

[6] K. R. Krishnan and T. J. Ott. State-dependant routing for telephone traffic: theory and results. In *Proceedings of 25th Conference on Decision and Control*, volume 10a, December 1986. Athens, Greece.

[7] F. P. Kelly. Loss Networks. *The Annals of Applied Probability*, 1(3):319–378, 1991.

[8] F. P. Kelly. Blocking Probabilities in large Circuit Switched Networks. *Adv. Appl. Prob.*, 18:473–505, 1986.

[9] S. Chung, A. Kashper, and K. W. Ross. Computing Approximate Blocking Probabilities for Large Loss Networks with State-Dependent Routing. *IEEE Trans. Networking*, 1(1):105–115, 1993.

[10] Graham Louth, Michael Mitzenmacher, and Frank Kelly. Computational Complexity of Loss Network. *Theoretical Computer Science*, 125:45–59, 1994.

[11] J. S. Kaufman. Blocking in a Shared Resource Enviornment. *IEEE Trans. Commun.*, 29(8):1474–1481, 1981.

[12] D. Mitra and J. A. Morrison. Erlang Capacity and Uniform Approximations for Shared Unbuffered Resources. *IEEE/ACM Trans. Networking*, 2(6):558–570, 1994.

[13] J. A. Morrison, K. G. Ramakrishnan, and D. Mitra. Refined Asymptotic Approximation to Loss Probabilities and Their Sensitivitiesin Shared Unbuffered Resources. *SIAM J. Appl. Math.*, 59(2):494–513, 1998.

[14] F. P. Kelly. Routing and capacity allocation in networks with trunk reservation. *Mathematics of Operations Research*, 15:771–793, 1990.

[15] D. Mitra, R. Gibbens, and B. D. Huang. State-Dependent Routing on Symmetric Loss Networks with Trunk Reservations – I. *IEEE Trans. Communications*, 41(2):400–411, 1993.

[16] D. Mitra and R. Gibbens. State-Dependent Routing on Symmetric Loss Networks with Trunk Reservations:II:Asymptotics, Optimal Design. *Annals of Operations Research*, 35(1), 1992.

[17] N. G. Bean, R. J. Gibbens, and S. Zachary. The performance of single resource loss systems in multiservice networks. In Jacques Labetoulle and James W. Roberts, editors, *The Fundamental Role of Teletraffic in the Evolution of Telecommunications Networks, Proceedings of the 14th International Teletraffic Congress — ITC 14*, volume 1a of *Teletraffic Science and Engineering*, pages 13–21. Elsevier Science B.V., June 1994. Antibes Juan-les-Pins.

[18] N. B. Bean, R. J. Gibbens, and S. Zachary. Asymptotic Analysis of Single Resource Loss Systems in Heavy Traffic, with Applications to Integrated Networks. *Advances in Applied Probability*, pages 273–292, March 1995.

[19] D. Medhi and I. Sukiman. Admission control and dynamic routing schemes for wide-area broadband networks: Their ineraction and network performance. In *Proceedings IFIP-IEEE Conf. on Broadband Communications*, April 1996. Montreal, Canada.

[20] D. Medhi, A. van de Liefvoort, and C. S. Reece. Performance Analysis of a Digital Link with Heterogeneous Multislot Traffic. *IEEE Trans. Comm.*, 43:968–976, 1995.

[21] K. R. Krishnan and F. Huebner-Szabo de Bucs. Admission control and state-dependent routing for multirate circuit switched traffic. In *Proceedings of 15th International Teletraffic Conference*, pages 1043–1054, June 1997. Washington, D. C.

[22] R. A. Howard. *Dynamic Programming and Markov Processes.* John Wiley, New York, 1st edition, 1960.

[23] A. Dasylva and R. Srikant. Bounds on the performance of admission control and routing policies for general topology networks with multiple call classes. 1999.

[24] S. Chung and K. W. Ross. Reduced Load Approximations for Multirate Loss Networks. *IEEE Trans. Commun.*, 41(8):1222–1231, 1993.

[25] A. G. Greenberg and R. Srikant. Computational Techniques for Accurate Performance Evaluation of Multirate, Multihop Communication Networks. *IEEE/ACM Trans. Networking*, 5(2):266–277, 1997.

[26] G. R. Ash, J. S. Chen, A. E. Frey, and B. D. Huang. Real-time network routing in an integrated network. Technical report, 1991. presented at ITC-13, Copenhagen, Denmark.

[27] A. Girard and M. A. Bell. Blocking evaluation for networks with residual capacity adaptive routing. *IEEE Trans. Commun.*, 37:1372–1380, 1990.

[28] R. J. Gibbens, F. P. Kelly, and R. E. Turner. Dynamic Routing in Multiparented Networks. *IEEE/ACM Trans. Networking*, 1(2):261–269, 1993.

[29] P. J. Hunt and C. N. Laws. Asymptotically Optimal Loss Network Control. *Mathematics of Operations Research*, 18:880–900, 1993.

[30] F. P. Kelly. Bounds on the performance of dynamic routing schemes for highly connected networks. *Mathematics of Operations Research*, 19:1–20, 1994.

[31] S. Chen and K. Nahrstedt. An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions. *IEEE Network*, 1998.

[32] A. Girard and R. Pagé. Dimensioning of Telephone Netwroks with Nonhierarchical Routing and Trunk Reservation. *Proc. of Networks'86*, pages 85–93, 1986.

[33] R. Guérin, H. Ahmadi, and M. Naghshineh. Equivalent Capacity and its Application to Bandwidth Allocation in High-Speed Netwroks. *IEEE J. on Selected Areas in Comm.*, 9:968–981, 1991.

[34] E. Rosenberg. A Nonlinear Programming Heuristic for Computing Optimal Link Capacities in a Multi-hour Alternate Routing Communications Network. *Operations Research*, 35:354–367, 1987.

[35] D. Medhi. Network Dimensioning for a Multi-Service Integrated Digital Network with Dynamic Routing. *Proc. of ACM/IEEE-CS Symposium on Applied Computing*, pages 33–36, 1991.

[36] D. Mitra, J. A. Morrison, and K. G. Ramakrishnan. ATM Network Design and Optimization: A Multirate Loss Network Framework. *IEEE Trans. Networking*, 4(4):531–543, 1996.

[37] D. Mitra, J. A. Morrison, and K. G. Ramakrishnan. Virtual Private Networks: Joint Resource Allocation and Routing Design. *Proc. INFOCOM*, 1(1):480–490, 1999.

[38] M. Liu and J. S. Baras. Fixed point apporoximation for multirate multihop loss networks with adaptive routing. Technical report, Institute for Systems Research, University of Maryland, College Park, MD, 1999. Technical Report TR99-44.

[39] D. Medhi. Multi-Hour, Multi-Traffic Class Network Design for Virtual Path-based Dynamically Reconfigurable Wide-Area ATM Netwroks. *IEEE/ACM Trans. Networking*, 3(6):809–818, 1995.

[40] D. Medhi and S. Guptan. Network Dimensioning and Performance of Multi-Service, Multi-Rate Loss Networks with Dynamic Routing. *IEEE/ACM Trans. Networking*, 5(6):944–957, 1997.

[41] M. Liu, J. S. Baras, and A. Misra. Performance evaluation in multi-rate multi-hop communication network with adaptive routing. *Proc. Annual ATIRP Conferrence*, 1:136–140, February 1998.

[42] D. Bertsekas and R. Gallager. *Data Networks.* Prentice Hall, 2nd edition, 1994.

[43] J. Behrens and J. J. Garcia-Luna-Aceves. Hierarchical Routing Using Link Vectors. *Proc. INFOCOM*, 1998.

[44] M. Montgomery and G. de Veciana. Hierarchical Source Routing Through Clouds. *Proc. INFOCOM*, 1998.

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

[46] R. Guérin and A. Orda. QoS Routing in Networks with Inaccurate Information: Theory and Algorithms. *IEEE/ACM Trans. Networking*, 7(3):350–364, 1999.

[47] A. Orda. Routing with End-to-End QoS Guarantees in Broadband Networks. *IEEE/ACM Trans. Networking*, 7(3):365–374, 1999.

[48] C. Partridge S. Shenker and R. Guerin. Specification of Guaranteed Quality of Service. *IETF RFC 2212*, 1997.

[49] A. Orda R. Guérin and D. Williams. QoS Routing Mechanisms and OSPF Extensions. *2nd IEEE Global Internet Mini-Conference*, November 1997. Phoenix, AZ.

[50] G. Chakraborty C. Pornavalai and N. Shiratori. QoS Based Routing Algorithm in Integrated Services Packet Networks. *Proc. IEEE ICNP*, 1997. Atlanta, GA.

[51] Louis B. Rall and George F. Corliss. An introduction to automatic differentiation. pages 1–18, 1990.

[52] Andreas Griewank, Christian Bischof, George Corliss, Alan Carle, and Karen Williamson. Derivative convergence for iterative equation solvers. *Optimization Methods and Software*, 2:321–355, 1993.

[53] Bruce Dhristianson. Reverse accumulation and attractive fixed points. *Optimization Methods and Software*, 3:311–326, 1994.

[54] Christian Bischof and Lucas Roh. *User's Guide to ADIC 1.0 Revision 1.0 Beta 1*, 1997.

[55] M. K. H. Fan, A. L. Tits, J. Zhou, L. S. Wang, and J. Koninchx. CONSOLE User's Manual. Technical Report TR 87-212r2, Systems Research Center, 1990.

[56] C. Lawrence, J. L. Zhou, and A. L. Tits. User's Guide for CFSQP Version 2.0. Technical Report TR 94-16, Institute for Systems Research, 1994.