Analytical Models for All-Optical Networks Employing Wavelength Division Multiplexing and Wavelength Conversion

by

Christopher Leith

A thesis submitted to the
Department of Mathematics and Statistics
in conformity with the requirements for
the degree of Master of Science

Queen's University
Kingston, Ontario, Canada
September 1999

Abstract

This work studies the effect of wavelength converters at nodes of a wavelength division multiplexed all-optical network. We develop analytical tools to calculate blocking in a network with arbitrary converter placement and a general loading scheme. Exact and approximate models to calculate blocking on short paths are derived and used to build recursions for longer paths. A measure of the network wide performance, in terms of blocking probability, is made by considering all of the routes in turn. In addition to modeling a specified converter placement, we propose heuristics to suggest placement based on given information about the network.

Acknowledgements

I would like to thank several people for their support during my program.

Most importantly, I want to thank my supervisor and friend, Glen Takahara. He was always available and willing to share ideas. I am grateful for his expertise in probability, and his overall guidance throughout my first research endeavour.

I also want to acknowledge everyone at Nortel Networks for their comments on my research, and general insight into the communications industry.

I am further indebted to Jon Davis, a professor in the Department of Mathematics and Statistics at Queen's University, for many impromptu crash courses on a variety of programming topics.

Finally, I would especially like to thank my parents, Steve and Dianne Leith, and my best friend, Erica Hyland, for their constant encouragement over the past two years.

Contents

List of Tables

List of Figures

List of Symbols

K	Number of consecutive links in K link models.
F	Number of wavelengths per fiber on every link of the network.
T	Set of session types.
$t \in T$	Integer string of link numbers denoting a particular session.
λ_t	Arrival rate of type t sessions.
n_t	Number of type t sessions in the system.
x_b	Number of wavelengths free on the links in the integer string b.
Y	Number of NCS blocks in a K link path.
y	NCS block.
$S_{block}^{(y)}$	Set of sets of links used by session types on block y.
$\underline{x}$	Vector of auxiliary variables.
X	Number of auxiliary variables.
$\underline{n}$	Vector of numbers of session types.
N	Number of session types.
$\underline{s}_{exact}$	Exact state.
S_{exact}	Exact state space.
D	Dimension of state space.

Π	Continuous time stationary distribution.
π_i	i^{th} element of Π.
Π^d	Discrete time stationary distribution.
π_i^d	i^{th} element of Π^d.
P	Probability matrix for an embedded discrete time Markov process.
p_{ij}	ij^{th} entry of P; probability of moving from i to j in one step.
q_{ij}	Transition rate from i to j in a continuous time Markov process.
d	Period of a discrete process.
P^*	Limiting matrix of the powers of P.
p_{ij}^*	ij^{th} entry of P^*.
P^{**}	P times the limiting matrix of the powers of P.
p_{ij}^{**}	ij^{th} entry of P^{**}.
$\Pi^{d(even)}$	Discrete time stationary distribution on even steps.
$\pi_i^{d(even)}$	i^{th} element of $\Pi^{d(even)}$.
$\Pi^{d(odd)}$	Discrete time stationary distribution on odd steps.
$\pi_i^{d(odd)}$	i^{th} element of $\Pi^{d(odd)}$.
M	Large number of steps.
$\underline{s}_{approx}$	Approximate state.
S_{approx}	Approximate state space.
$\underline{n}_i^j$	Vector of consecutive elements $n_i, \ldots, n_j$.
I	Number of session types with only 1 link.
$G^{(I+i)}(\underline{n}_{I+i+1}^D)$	Normalization constant for stationary distribution at stage i.
$\gamma_j^{(i)}$	Arrival rate of a type i when there are j type i's in progress.
$R^{(i)}(\underline{n}_1^D)$	Probability of accepting a type i while in state $\underline{n}_1^D$.

$\Pi(\underline{n}_1^i \mid \underline{n}_{i+1}^D)$	Stationary distribution of $\underline{n}_1^i$ given $\underline{n}_{i+1}^D$.
L	Arbitrary number of links for recursion.
l	Stage of recursion.
n_{123f}	Number of wavelengths free on all l links.
n_{3f}	Number of wavelengths free on the last link of an l-hop path.
n_{2f}	Number of wavelengths free on the second last link of an l-hop path.
$\underline{n}_f$	Vector consisting of n_{123f}, n_{3f} and n_{2f}.
x_{123f}	Number of wavelengths free on all $(l-2)$ links.
x_{3f}	Number of wavelengths free on the last link of an $(l-2)$-hop path.
x_{2f}	Number of wavelengths free on the second last link of an $(l-2)$-hop path.
$\underline{x}_f$	Vector consisting of x_{123f}, x_{3f} and x_{2f}.
$B^{(l)}(\underline{n}_f)$	Probability of n_{123f}, n_{3f} and n_{2f} on an l-hop path.
$\underline{s}_{exact}^{(1)}$	Exact state on first identified set of links in recursion.
$\underline{s}_{exact}^{(2)}$	Exact state on second identified set of links in recursion.
$Q^{(L)}$	Probability of blocking an L-hop session on L links.
V	Set of vertices.
E	Set of edges.
G	Graph representing the network topology.
R	Ordered set of routes.
Λ	Ordered set of loads corresponding to the routes in R.
W	Subset of V indicating the nodes with wavelength converters.
$r \in R$	A route.
$\lambda_r \in \Lambda$	Load on route r.

$Q^{(L_r)}(r)$	Probability of blocking an L_r-hop session on route r.
U_{nwb}	Weighted network-wide blocking.
$v \in V$	A node from V.
m_v	Measure of the load at node v.
$\Lambda_i(v)$	Sum of the loads of routes that use v at an intermediate point.
$\Lambda_e(v)$	Sum of the loads of routes that use v at an endpoint.
w	Contribution factor for $\Lambda_e(v)$.
α	Threshold for m_v when calculating converter placement.
β	Threshold for $\Lambda_i(v)$ when calculating converter placement.
C	Maximum number of converters allowed when calculating converter placement.
ρ	Load parameter used in examples.

Chapter 1

Introduction

1.1 Motivation

There is little doubt that high speed networks of the future will be all-optical. Already the world's fastest networks are striving towards this end. The Canadian research network CA*net III, for example, is an almost entirely optical wavelength division multiplexed (WDM) network. It employs 8 wavelengths per fiber, and does rely on some electronic hardware at its nodes. In time, traffic on networks like this will most likely remain entirely in the optical domain. While a WDM optical link has an enormous available bandwidth, currently the electronic equipment for switching, processing and buffering signals results in the node being the focus of congestion [6], and overall the network throughput is restricted.

The demand for network bandwidth continues to grow at a rapid pace. Widespread use of new technologies and applications, such as HDTV and video conferencing, is driving the need for faster and faster networks. The optical domain is the logical choice for satisfying this demand because of the potentially vast amount of bandwidth it can offer. In some systems, for example, a single wavelength is capable of carrying 10 Gb/s for very long distances without significant error. Moreover, many different wavelengths may be used on a single fiber, as in a wavelength division multiplexed network.

Currently, these benefits of the optical domain are not fully exploited. Because there is no practical or feasible way to manipulate data and do switching and routing entirely in the optical domain we are forced to drop to the electrical domain at several points in the transmission path of a session. These drops slow the overall transfer since the electronic hardware has a lower bandwidth capacity than the optical link. With the advent of purely optical switches and other data manipulating hardware we attempt to eliminate these bottlenecks.

Realistically however, while these optical devices are in their infancy they will remain a complex and expensive piece of hardware. For this reason, not every node of a network will be likely to receive such a device. Where we once had electronics at possibly every node, and hence had a form of conversion from one wavelength to another for free, we now only have wavelength conversion at some subset of the nodes. This situation, termed *sparse wavelength conversion* in [22], is our area of study.

In the absence of wavelength changing devices a session is restricted to use the same wavelength on all of the links of its path, which is then called a *lightpath* [6]. If wavelength changers exist at nodes, sessions may use different wavelengths on different hops, if necessary, to avoid blocking (see [22] for a discussion of some intuitive situations where the addition of wavelength changers may not significantly improve blocking performance). It is clear that the probability of a session being blocked is lower when the network employs wavelength changers, but a quantitative analysis of the relative gain is needed. We shall distinguish between the situation where no nodes have wavelength converters and the situation in which all nodes have wavelength changers by referring to them as the non-circuit switched (NCS) case and the circuit switched (CS) case, respectively. Although these are two important extremes, there may be many cases in between when there are a number of nodes in the network. In fact, it is these latter sparse cases that are possibly the most interesting because we have already seen that they are the most likely in reality. Hence, we would like to be able to analyze any case under quite general loading. Moreover, instead of being handed the wavelength converter placement, it would be advantageous to have heuristics to suggest reasonable converter placement based on network information.

1.2 Thesis Overview

Our goal is to analyze the performance of an arbitrary all-optical network in order to determine the potential benefit of wavelength converters. Our measure of the network performance is derived in terms of session blocking. Furthermore, assuming there is a significant advantage to using some converters, we address the issue of optimal or near optimal converter placement. Presently, this is an interesting problem, however, it is not clear how the technology will evolve in this area. For example, the problem could disappear entirely in the future if the cost of wavelength converters does not remain high, or the amount of unused bandwidth per link, available for new sessions, increases significantly (reducing the need for converters). While the latter is not likely given the current growth in demand for network bandwidth, the former is quite possible, but several years away.

We consider future networks in which the amount of bandwidth available is very large. In fact, we measure the amount of bandwidth in terms of the number of wavelengths available per link. Moreover, we assume an ideal wavelength converter that can simultaneously convert any wavelength on any input fiber to any wavelength on any output fiber when placed at a node. In reality this type of functionality would probably require many physical devices working together.

A two phase solution is actually employed. In the first phase we develop tools to analyze a generally loaded network where the arbitrary wavelength converter placement is given. In the second phase we propose heuristics for placing converters. That is, ultimately we use the heuristics, in the second phase, to specify various converter placements, and then analyze each one, using the tools of the first phase, to deduce the placement with the best performance.

In order to accomplish all of this we first develop exact and approximate models for short paths in chapter 3. The next step is to use these models as base units for a recursion in chapter 4 to analyze any length path. Later in chapter 4 we explain how we systematically analyze all of the routes of a network to get a measure of the global network-wide blocking. Once we can analyze a network with arbitrary routes, loading scheme and converter placement, we move on to developing heuristics for best converter placement in chapter 5. Finally, we present several network examples in chapter 6 to verify our calculations. For each example we compare our analytical determination of blocking to computer simulations for validation. Chapter 7 gives a discussion of our conclusions and outlines some future work in this area.

1.3 Contributions

The following is a list of the major contributions of this work.

- Exact K link model for general non CS cases based on appropriate Markov process, for small K.
- Alternative approximate K link model for general non CS cases, for small K.
- Recursion to calculate blocking for L-hop paths of optical networks, for arbitrary L, again with general loading and arbitrary converter placement.
- Network-wide blocking analysis of an optical network with a general loading scheme and arbitrary converter placement.
- Heuristics for converter placement based on network loading information.
- Software tool written in C++ implementing all of the above analytical calculations, as well as a simulation module written in C, to study network examples.

Chapter 2

Literature Review

2.1 General Optical Networking

The use of optical fiber in transmission of communication data is not new. For many years long haul network planners have utilized optical links to build high speed backbones. Many of the beneficial characteristics of optical technology were apparent from the beginning, and ultimately, they are driving the move towards all-optical systems. The motivations for optical networking are nicely summarized in [10].

Most importantly perhaps, there is the obvious demand for bandwidth which seemingly only optics can supply. For example, [10] points out the explosive growth of the World Wide Web. While most users have embraced the access to vast amounts of data on the Internet, the web still falls far short of the audio and video experience it could be. Not only is there already a need for more bandwidth per user due to current and future applications' hungry bandwidth requirements, but there are more and more users every day as homes and offices come online. For now, dedicated copper wire telephone lines and coax cable are providing temporary relief. In [25] it is suggested that terabit per second transmission rates are quite feasible today, although this was not always the case. One of the most significant performance improvements was realized in wavelength division multiplexing. This simply amounts to using many different wavelengths on a single fiber. Operational systems of around 100 Gb/s are now available.

Another key advantage of fiber systems is the lower error rate. In [10] we see that this is because of both the large available bandwidth in the fiber and photon characteristics. The immense amount of bandwidth reduces the need for source coding (compression) which can introduce errors if it is not lossless, and the fact that moving photons have very little interaction means that there is only a slight chance of crosstalk and impulse noise as is the case with copper wire.

Protocol transparency is another benefit of using an optical system. There are many different protocols in use today (there is a concise figure depicting the most common ones in [10]), usually each with its own clearly defined rate of transmission. In the future it seems that more and more a single network will have to carry many of these different protocols simultaneously. The author of [10] points out that protocol variation is one of the largest inhibitors to network flexibility and growth. Transparency is accomplished when the end users need only to worry about the protocol they want to use and are not further forced to obey any imposed network protocol. There is no electronic equipment that needs to be synchronized in an all-optical system, so according to [25], the optical routes can seamlessly transmit signals over thousands of kilometers (with the help of chemical amplifiers) at any protocol rate.

A final advantage of optical networking worth noting is that of reliability and maintenance costs. Because there may be less electronic equipment in the optical system ideally (some optical components can be passive), there is less chance for failure. For example, an optical fiber, chemically doped to amplify the signal, may stretch for thousands of kilometers, whereas to achieve the same distance with copper wire might require several intermediate stations housing complex electronic equipment designed to reshape and amplify the signal.

2.2 Optical Hardware Considerations

With the move towards all-optical networks, hardware designers are faced with new challenges. To build a truly all-optical network would require completely optical switches and other hardware to add and drop sessions at nodes. Some of the aspects of all-optical network nodes are investigated in [21] and [8].

In high performance packet switched networks, some form of queue buffering will need to be done [12]. For this reason, [8] looks at a proposed form of optical buffer called a fiber delay line. Essentially, at the physical level the buffer consists of a specific length of coiled fiber. By forcing the light signal to travel through this optical fiber you effectively delay it for a time depending on the length. In [8] an analytical model for a multi-wavelength packet switch is constructed and used to analyze the number of these fiber delay lines required in different hardware configurations.

Optical networks with minimal electronic domain intervention are being investigated. The authors of [21] propose and experimentally demonstrate a WDM ring topology in which electronics are only used at a node to provide add-drop capabilities. The routing remains entirely in the optical domain by simply assigning each node in the ring a specific wavelength that is generated and maintained by a centralized node. Only signals that originate or terminate at the node need to be dealt with in the electrical domain, while all other sessions are passively routed through the node. The experimental results in [21] evidence the feasibility of a very high speed optical ring system.

2.3 Optical Routing

Another general area of study is that of routing within an optical network. Previously, with electronic equipment at every node, a session was not restricted to use the same channel (wavelength) on an incoming link and outgoing link of a node. That is, having the electronic equipment at the node essentially provided the conversion capabilities. However, in an all-optical network the ability to convert between wavelengths may not exist at every node. This brings up the notions of *lightpaths* and *semilightpaths*. A lightpath is simply an optical data channel established between two nodes by assigning a single wavelength. A semilightpath, as defined in [7], is constructed by chaining several lightpaths together with wavelength conversion at their connection points. That is, a semilightpath establishes a connection between two nodes of a network on possibly many different wavelengths provided there are wavelength converters available at the appropriate conversion points. With this in mind, interesting questions arise. How can we optimally route sessions in a WDM optical network using only lightpaths? How can we optimally route semilightpaths in a WDM optical network with wavelength converters? There are attempts to provide answers to the first question in [6] and [24], and the second question is addressed in [7] .

The routing problem may either be static or dynamic. The *static lightpath establishment* problem, as defined in [6], concerns itself with how to establish a set of given routes simultaneously, while the dynamic problem only deals with the on-the-fly addition of one route to an already running system. In [6] the authors first show that the static routing problem is NP-complete by comparing it to a graph-colourability problem. They then go on to develop lightpath establishment heuristics for both the static and dynamic cases when there are either a bounded or unbounded set of wavelengths available. Ultimately, the heuristics in [6] attempt to minimize the number of wavelengths needed in the static unbounded case, or the ratio of number of lightpaths rejected to lightpaths requested in the static bounded case. In the dynamic situations, the authors also take into account the stability of the network. That is, establishing a lightpath should not lead to a fragmenting of the network or degradation of the network performance at some later time.

Useful bounds that apply to any lightpath routing algorithm have also been discovered. In [24], an upper bound on the carried traffic when there is a fixed number of wavelengths available is presented. The bound is derived by a suitably formulated linear programming problem. Note that this is equivalent to a lower bound on the number of wavelengths required to carry a given traffic load.

A very clever semilightpath routing algorithm is devised in [7]. The goal of the algorithm is to find the minimum cost semilightpath between given source and destination nodes. The basic idea is to first transform a graph representing the physical network topology into an auxiliary graph that contains information about each wavelength on each physical link, and then apply a simple algorithm similar to Dijkstra's algorithm to find the minimum cost route. Roughly speaking, if there are F wavelengths in the network the auxiliary graph is formed by taking F copies of the original graph and linking them together appropriately depending on where the converters are placed in the network. Because of the simplicity and speed of the algorithm, it could be used dynamically provided the link costs in the auxiliary graph were updated to reflect the wavelengths in use on the physical network.

2.4 All-Optical Blocking Analysis

Much of the theory today is based on pioneering work in telecommunications of many years ago. Probability has been applied to communication problems since the work of Erlang, Fry and Molina, for example. Lee credits these researchers in his paper in 1955, [15], and further develops the field. Others built on this work (see [20]), and even today the fundamental probabilistic tools come up again and again when modeling various types of switching networks.

The current models for optical networks have several common features. This similarity arises because certain assumptions are usually made to keep analytical calculations feasible. For example, most models assume that the number of wavelengths available on each link of the network in question is constant, and that sessions require a full wavelength on each link of their paths. Moreover, often the arrival process of the sessions is taken to be Poisson. In [23] however, the authors relax this assumption to some extent. Several papers actually consider load correlation from link to link, while others often assume independence. In some cases a CS model is used to approximate a NCS situation. Call blocking is usually calculated as a measure of network performance.

Some papers, for example [2], model a completely CS network and a completely NCS network and compare the two. This is a valid first step in investigating the usefulness of wavelength converters because it analyzes two extremes. If there is no significant difference between these extremes, there is no need to worry about cases in between. On the other hand, some authors feel that the difference is worth probing further, and recent papers have emerged that attempt to model the sparse wavelength conversion scenario, for instance [22].

In [4] an analytical model is developed and tested with two different routing schemes. The model itself is quite reasonable. It takes into account the difference between the CS case and the NCS case by using a simple combinatorial expression that reflects the probability that the same wavelength is free on every link of a path given the number of wavelengths in use on each link. By comparing to simulation results the author finds that his model performs well using a fixed routing scheme, but he admits that the accuracy of the approximation is not as good under a least loaded routing scheme.

The authors of another paper, [2], derive a more sophisticated analytical model for a path with or without wavelength converters. The model takes into account load correlation from link to link (ie. it does not assume independence from one link to the next) and pays particular attention to quantifiable network characteristics such as average path length and interference length (which they define as the expected number of hops shared by two sessions which share at least one hop). The authors point out that their model accurately predicts network behavior demonstrated by others, so indeed the model performs quite well. In particular, [2] verifies the dependence of blocking probability on the number of hops in a path (blocking rises as path length increases), and concludes that the effect is more pronounced when there are no converters. The authors also suggest that as the interference length increases, there becomes less benefit from wavelength converters. The idea is that networks with a large interference length, for example a ring topology, will require many wavelengths whether or not converters are present.

The model in [2] is quite good but it does not consider sparse wavelength conversion. That is, wavelength converters at some subset of the nodes of a network. This deficiency is addressed in [22] with a new model that ultimately studies networks where the converters are randomly placed according to a binomial distribution with parameters n and q. Here n corresponds to the number of nodes in the network, and q is the probability that a converter exists at a node. The analytical expression to calculate blocking on a given path is derived using a recursion based on a CS 2 link model. Unfortunately, the model does assume that there is a uniform load of sessions to each node. Still, the overall results as compared with simulation are very good in a variety of topology types. The converter density parameter q, as it is defined in [22], is an interesting concept that lets us investigate the complete spectrum from one extreme of completely NCS ($q = 0$) to completely CS ($q = 1$) for the first time. Unfortunately, converters will not be placed randomly in the network in reality. But, by studying fairly symmetric networks with this model the authors are able to conclude that having converters at a relatively small fraction of the network nodes is generally enough to achieve the greatest benefit from the devices. That is, the gains from using more and more wavelength changing devices tend to drop off after a small percentage of nodes have received converters.

Although the assumption that sessions arrive to the network as a Poisson stream is probably quite valid, [23] presents an analysis in which the arrivals are more general. In [23] sessions arrive according to conditional Poisson processes where the arrival rates depend on the state of the system. The reason for modeling different arrival processes to a link is to capture the peakedness, defined in [23] to be the variance of the number of busy circuits on a link divided by the expected number of busy circuits on the link. In more familiar terms, the peakedness is a measure of the burstiness of the traffic. As in other papers, the two cases of CS and NCS are examined, and so this new model studies the effect of peakedness as it relates to the benefit from wavelength converters. The authors conclude that there is less benefit from wavelength converters as the network traffic becomes increasingly bursty.

Chapter 3

Exact and Approximate Models for K Links

3.1 Exact Analysis

In this section we analyze K consecutive links of an all-optical network (AON) exactly. Our goal is to find the probability that the system (after it is in steady-state) is in a given configuration (in terms of the numbers of sessions currently in progress). With this information we can calculate the probability that any session defined entirely on the K links is blocked. In order to accomplish our goal we define an appropriate Markov process on the K links and determine its stationary distribution.

We consider arbitrary wavelength converter placement, but make the following assumptions about the network:

1. AON with nodes capable of transmitting and receiving exactly F wavelengths

on each link.

2. Sessions between source and destination require a full wavelength on every link of their paths.

3. Wavelengths are assigned randomly from the set of available wavelengths.

4. Wavelength converters are capable of converting from all F wavelengths to all F wavelengths.

5. Paths for routing calls are independent of the state of the links.

6. Calls arrive at nodes according to Poisson processes with arbitrary destination.

7. Once accepted, calls have exponential holding times with unit means.

Please note that we are modelling a high speed all-optical network that may exist in the future. We are explicitly interested in networks in which individual sessions require a huge amount of bandwidth, by today's standards, for a finite amount of time. That is, we model sessions (which will most likely be comprised of a number of end-user sessions multiplexed together) that require a full wavelength of bandwidth (ie. the smallest unit of bandwidth in this work), but only for their relatively short duration. Actually, if we consider both time division multiplexing (TDM) as well as WDM we expect the blocking to be lower. TDM within a wavelength allows the possibility for sessions to share a wavelength, even if they have links in common. Even when the assumption that sessions use a full wavelength of bandwidth is not ideal (it depends on how a session is defined), we should obtain an upper bound on the blocking using this model.

On our K links we identify at most $\sum_{i=1}^{K} i = \frac{K(K+1)}{2}$ distinct session types, each with its own arrival rate, corresponding to the links used. We only include sessions that use sequential links. For example, when $K = 3$ there may be sessions of type 1, 2, 3, 12, 23, or 123, but not of type 13. In general, let $T = \{t_1, t_2, \ldots, t_{|T|}\}$ be the set of session types, where each t_i is a string of integers indicating which links the session type uses. Let n_t represent the number of sessions in progress of type $t \in T$, and λ_t the arrival rate of this type. Note that a given session type does not necessarily correspond to only one physical session in the network. For example, session type $12...K$, as viewed on this particular set of K links, may actually represent several different sessions, as viewed at the network level, because all of the different physical sessions happen to have our K links in common. Moreover, the arrival rate (or load, because of assumption 5 above) of session type $12...K$ on our K links is the sum of all the different physical session's arrival rates.

Now, notice that when there are converters at every node, the fully CS case, we can define a process $\underline{n}_{cs} = (n_{t_1}, n_{t_2}, \ldots, n_{t_{|T|}})$ with the following state space:

$$S_{cs} = \{\underline{n}_{cs} \mid \sum_{t \in T:A_1} n_t \leq F, \sum_{t \in T:A_2} n_t \leq F, \ldots \sum_{t \in T:A_K} n_t \leq F\} \tag{3.1}$$

where A_i is the event that session type t uses link i.

The product form stationary distribution of this process is known:

$$\Pi_{cs}(\underline{n}_{cs}) = G \frac{{\lambda_{t_1}}^{n_{t_1}}}{n_{t_1}!} \frac{{\lambda_{t_2}}^{n_{t_2}}}{n_{t_2}!} \cdots \frac{{\lambda_{t_{|T|}}}^{n_{t_{|T|}}}}{n_{t_{|T|}}!} \tag{3.2}$$

where G is a normalization constant.

The state space above is not sufficient to define a Markov process when some nodes lack a converter. With arbitrary converter placement, the model will be developed in two parts. We begin by defining an appropriate Markov process, and then present a feasible method for solving for its stationary distribution.

In addition to the $|T|$ session types, we need to have auxiliary variables to account for the fact that sets of session types may or may not share wavelengths, and this will affect whether a new session can be accepted. For example, consider $F = 2$ and the situation where currently there are only two sessions, one of type 1 and the other of type 2. If no converter exists between link 1 and link 2, a type 12 session will be accepted if the two sessions are on the same wavelength and blocked otherwise. To overcome this problem we must keep track of not only the numbers of each of the session types in the system, but also some information about what wavelengths are available for particular parts of sessions. That is, the K links are naturally divided into Y blocks of consecutive links when there are $Y+1 < K$ wavelength converters at internal nodes (not at the endpoints). Each time a session type uses links from one of these NCS blocks there must be some wavelength available for it. If there is a chance that 2 or more session types may or may not affect the acceptance of a given session type on the block, as in the simple example above, we must know exactly how many wavelengths are free for the given session on the block. Notice that the information we need is entirely within a NCS block.

So, in order to keep track of enough information to determine whether any session type may be accepted, and produce a state space that is as small as possible at the same time, we define auxiliary variables using the following procedure:

1. Consider each NCS block y, for $y = 1, \ldots, Y$, and the session types that use

links in the block. For each subset of this list of sessions that use the same links on the block add an element, say b, to the set $S^{(y)}_{block}$, where b is an integer string of the links used.

2. Finally, for each $b \in S^{(y)}_{block}$ such that there are elements $b', b'' \in S^{(y)}_{block}$ where b' and b'' are disjoint subsets of the links in b, we define an auxiliary variable, x_b, to be the number of wavelengths with all of the links corresponding to b free.

Thus we obtain a set of, say X, auxiliary variables $\{x_{b_1}, x_{b_2}, \ldots, x_{b_X}\}$ and we define $\underline{x} = (x_{b_1}, x_{b_2}, \ldots, x_{b_X})$. x_b represents the number of wavelengths available for any session requiring the links in the integer string b. Together with the set of, say $N = |T|$, variables representing the number of each session type, and the corresponding vector $\underline{n} = (n_{t_1}, n_{t_2}, \ldots, n_{t_N})$ we can define a process $\underline{s}_{exact} = (\underline{n}, \underline{x})$. We describe the state space of the process, S_{exact}, in 2 parts as follows:

<u>Exact State Space Constraints</u>

1. First we describe the constraints on the numbers of session types, since they depend on each other. That is, there are certain session types that have links in common, and therefore can not share wavelengths. Specifically, each member of a set of session types in which any pair of session types have at least 1 link in common must use a unique wavelength. To determine the sets systematically, consider a graph on N nodes constructed as follows: An edge is placed between node i and node j if session types i and j have at least one link in common. A completely connected subgraph is maximal if adding any node to the subgraph results in a graph that is not completely connected. There is a one to one relationship between the number of maximal completely connected subgraphs

and constraint equations in terms of the n_i's. For a given maximal completely connected subgraph on the nodes $i_1, i_2, ..., i_k$ (representing the largest set of pairwise overlapping session types), the corresponding constraint is $\sum_{j=1}^{k} n_{i_j} \leq F$. That is, any state, $\underline{s}_{exact}$, must satisfy this set of constraint equations.

2. In addition to the number of session types satisfying the first condition we must consider the arrangement of the sessions on the K links. We must determine the allowed values of the auxiliary variables, given that an auxiliary variable represents the number of wavelengths free on a set of links, for each tuple $\underline{n}$. That is, a given tuple $\underline{n}$ will correspond to many states in general because of the various values of $\underline{x}$.

So, for a given value of F, we first determine all possible tuples $\underline{n}$, and then for each tuple $\underline{n}$ we determine all of the states based on the allowed values of $\underline{x}$. The actual state space, S_{exact}, is the collection of all states defined this way.

As an example, let's consider the case where we consider all 6 possible session types on 3 links, there are no wavelength converters present, and $F = 3$. From part 1 above we see that one tuple $\underline{n} = (n_1, n_2, n_3, n_{12}, n_{23}, n_{123})$ satisfying the constraints $n_1 + n_{12} + n_{123} \leq F$, $n_2 + n_{12} + n_{23} + n_{123} \leq F$ and $n_3 + n_{23} + n_{123} \leq F$ in this case is $(1, 0, 1, 1, 0, 0)$. Figures 3.1, 3.2 and 3.3 depict 3 states that arise from this $\underline{n}$, corresponding to 3 different values for $\underline{x} = (x_{12}, x_{23}, x_{123})$ (recall how the auxiliary variables are defined).

In each case the rows in the figure represent the $F = 3$ different wavelengths, labeled 0, 1 and 2, and therefore we can picture which wavelengths each session type uses on any link. We see that figures 3.1, 3.2 and 3.3 correspond to the 3 states,

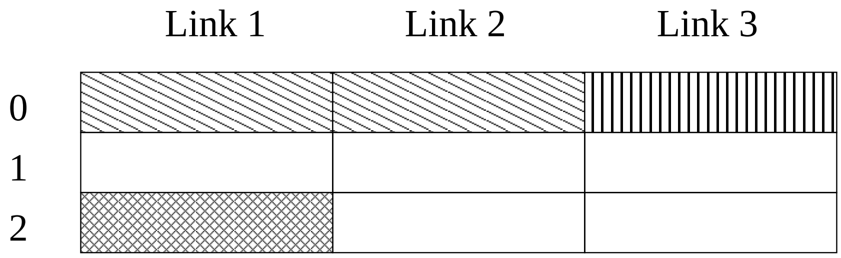

Figure 3.1: Exact State $(1, 0, 1, 1, 0, 0, 1, 2, 1)$

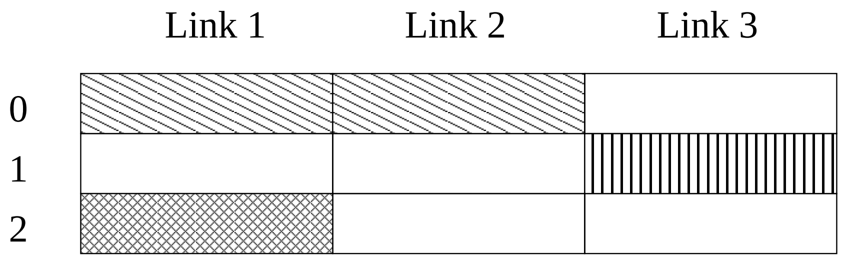

Figure 3.2: Exact State $(1, 0, 1, 1, 0, 0, 1, 1, 0)$

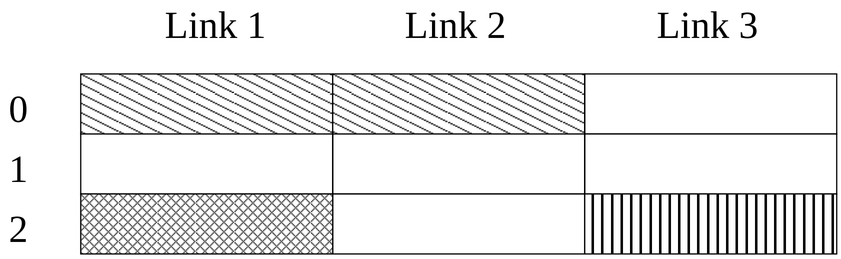

Figure 3.3: Exact State $(1, 0, 1, 1, 0, 0, 1, 1, 1)$

$(1, 0, 1, 1, 0, 0, 1, 2, 1)$, $(1, 0, 1, 1, 0, 0, 1, 1, 0)$ and $(1, 0, 1, 1, 0, 0, 1, 1, 1)$ respectively.

We have described the state space of a continuous time Markov process, $\underline{s}_{exact}$. We see that the dimension of this process is $D = N + X$. The rates at which the process moves from one state to another are completely determined because we have been careful in our design to include enough information to conclude what sessions may arrive and what sessions may depart when the system is in a given state. We will solve for the continuous time stationary distribution, say $\Pi(\underline{s}_{exact})$, by forming an embedded discrete time chain, and calculating its stationary distribution, say Π^d. Once we have $\Pi^d = (\pi_1^d, \pi_2^d, \ldots, \pi_{|S_{exact}|}^d)$ we can calculate $\Pi = (\pi_1, \pi_2, \ldots, \pi_{|S_{exact}|})$ using the relation

$$\pi_i = \frac{\pi_i^d}{\sum\limits_{k \in S_{exact}} q_{ik}}. \tag{3.3}$$

The epochs of the embedded chain will be the points in time when a session arrives or departs. We will use the characteristic probability matrix, P, of the discrete time chain. The ij^{th} element of P, say p_{ij}, is the one step transition probability of moving from state i to state j. With q_{ij} as the rate at which you move from state i to state j in the continuous chain, p_{ij}, can be easily determined as follows:

$$p_{ij} = \frac{q_{ij}}{\sum\limits_{k \in S_{exact}} q_{ik}}. \tag{3.4}$$

There will be many states, even for moderate values of F, so a direct calculation of Π^d from the system of linear equations $\Pi^d = \Pi^d P$ is not feasible. Instead, we use the following algorithm to determine Π^d:

Exact Iterative Algorithm

1. Take a guess for Π_0^d (for example uniform), and set $k = 1$.
2. Calculate $\Pi_k^d = \Pi_{k-1}^d P$ and $\Pi_{k+1}^d = \Pi_k^d P$.
3. If $|\Pi_{k+1}^d - \Pi_{k-1}^d| < E$, where E is a predefined allowed error, goto 4, else set $k = k + 1$ and go to 2.
4. Calculate Π^d as the componentwise average of Π_{k+1}^d and Π_k^d.

Before we prove the correctness of this algorithm let us recall an important limit theorem:

Theorem 3.1 *[Grimmett and Stirzaker, Theorem (21), p. 217 [11]]*

For an aperiodic state j of a Markov chain $p_{jj}(n) \to \frac{1}{\mu_j}$ as $n \to \infty$. Furthermore, if i is any other state $p_{ij}(n) \to \frac{1}{\mu_j} f_{ij}$ as $n \to \infty$.

Here μ_j is the mean recurrence time of state j, and f_{ij} is the probability of ever reaching j having started in i.

Corollary 3.2 *[Grimmett and Stirzaker, Corollary (22), p. 217 [11]]*

With $\tau_{ij}(n) = \dfrac{1}{n} \sum\limits_{m=1}^{n} p_{ij}(m)$ as the mean proportion of elapsed time up to the nth step during which the chain was in state j, given that it started in i, we have $\tau_{ij}(n) \to \frac{1}{\mu_j} f_{ij}$ as $n \to \infty$.

Notice that the above theorem states that for an aperiodic chain with transition matrix P, as $n \to \infty$, P^n tends to a matrix where the ijth entry is the long-run proportion of time the chain spends in j given that it started in i.

Now we can present the following theory:

Theorem 3.3

The Exact Iterative Algorithm converges to the true stationary distribution of the process we constructed in this section.

Proof

First we note that our process is of period $d = 2$.

The key observation is that for every transition that occurs, exactly one of the n's changes by exactly one. That is, we move from state i to state j because one and only one of a particular session type either arrives or departs.

Imagine beginning in state i, leaving i, and returning to i in k steps.

$i \to j_1 \to j_2 \to \ldots \to j_{k-1} \to i$

We move from i to j_1 because some session type has arrived/departed. At some later step one of these session types must depart/arrive or else we contradict our return to i. Similarly, we move from j_1 to j_2 because some session type has arrived/departed, and again, one of that session type must later depart/arrive. In fact, this pairing continues such that there will be $\frac{k}{2}$ (an integer) arrivals matched with $\frac{k}{2}$ departures. Hence, starting from i we always return to i in an even number of steps. In particular, beginning from any state i we can always leave i and return again in 2 steps if some session type arrives/departs and then it immediately departs/arrives. Therefore the period, d, which by definition is the greatest common divisor of the set of all number of steps in which you can leave a state and return again, is 2.

Note that our process is irreducible, but not aperiodic. In order to use the above theorem we need to think of decomposing the discrete chain embedded in our process into its even and odd steps. We get 2 aperiodic discrete processes, not necessarily irreducible since we are using the original state space and in general not all states will be reachable at every other step for a given starting point. If our original chain has probability transition matrix P, then both the aperiodic chain on the odd steps and the aperiodic chain on the even steps have probability transition matrix P^2. Moreover, by the above theorem we have a limiting matrix, say P^*, as $n \to \infty$. That is, $P^* = \lim_{n\to\infty}(P^2)^n$.

The algorithm produces a sequence of distributions Π_0^d, $\Pi_0^d P$, $\Pi_0^d P^2$, $\Pi_0^d P^3$, $\Pi_0^d P^4$, $\Pi_0^d P^5$, ... and eventually converges to an alternating sequence of $\Pi^{d^{(even)}}$ and $\Pi^{d^{(odd)}}$ where $\Pi^{d^{(even)}} = \Pi_0^d P^*$ and $\Pi^{d^{(odd)}} = \Pi_0^d P P^*$.

Thinking of Π_0^d as the probability distribution of the state we start in we see that:

$\pi_j^{d^{(even)}}$ = long run proportion of even times in which system is in state $j = \lim_{M\to\infty} P($system is in j after 2^M steps$) = \sum_i \lim_{M\to\infty} P($system is in j after 2^M steps $|$started in $i)P($started in $i) = \sum_i p_{ij}^* \pi_{0\,i}^d$.

Similarly, with $P^{**} = PP^*$ and hence $p_{ij}^{**} = \sum_k p_{ik} p_{kj}^* = \sum_k \lim_{M\to\infty} P($system is in j after 2^M steps $|$started in $k)P($go from i to k on the first step$) = \lim_{M\to\infty} Pr\{$system is in j after $2^M + 1$ steps $|$started in $i\}$ we have $\pi_j^{d^{(odd)}}$ = long run proportion of odd times system is in $j = \sum_i p_{ij}^{**} \pi_{0\,i}^d$.

Finally, $\Pi^d = (\pi_1^d, \pi_2^d, \ldots, \pi_{|S_{exact}|}^d)$ is the componentwise average because the system is in an odd time half of the time and in an even time half the time. That is $\pi_j^d = \frac{1}{2}\pi_j^{d^{(even)}} + \frac{1}{2}\pi_j^{d^{(odd)}}$. $\diamond$

REMARKS:

1. Although $\pi_j^d = \frac{p^*_{jj}}{2}$ since the original process was irreducible, this algorithm is preferred for practical computational reasons because it avoids several $|S_{exact}| \times |S_{exact}|$ by $|S_{exact}| \times |S_{exact}|$ matrix multiplications (to calculate $P^* = \lim_{n\to\infty}(P^2)^n$). In fact, the algorithm above only ever requires a vector - matrix multiplication.

2. Storing P and calculating $\Pi_k^d P$ is quite feasible and efficient since P, corresponding to our process, is sparse.

3. In general, if the period of the process is d, we can decompose into d aperiodic discrete time processes. The algorithm (with the obvious modification that you must check every d^{th} distribution in the sequence for convergence) produces d distributions and the true stationary distribution is similarly the average of these.

The algorithm runs quite fast, with its main limitation being storage for the matrix P. The number of non zero entries in P determines whether or not it is feasible to calculate the stationary distribution of a given process, not the square of the size of the process's state space. In our application however, the number of non zero entries of P grows at approximately the same rate as the size of the state space. The number of possible states reachable from a given state (ie. the number of possible non zero entries in any row of P) is always some fixed constant number for a given K. For example, with $K = 3$ there are at most 44 non zero entries in any row of P, regardless of the value of F. We can use the algorithm successfully on the Markov process defined earlier in this section with $K = 3$ and up to about $F = 10$. This corresponds to a state space on the order of 10^5 and a probability matrix P where the number of non zero entries are on the order of 10^6.

3.2 Approximate Analysis

The previous section reveals that the size of the state space grows rapidly with K and F, especially in the absence of wavelength converters. For this reason, we describe an approximate analysis of K consecutive links using a process with a smaller state space. This process can be used to analyze longer paths (for reasonable values of F) than the exact process developed in section 3.1. As before, this model not only handles both the CS and NCS cases, it is general enough to cover situations in between where there are wavelength converters at some subset of the nodes in the K hop path. The assumptions about the network are the same as in section 3.1.

The first step is to fix K and decide which session types we will consider. Note again that there is a maximum of $\sum_{i=1}^{K} i = \frac{K(K+1)}{2}$ distinct types that use consecutive links. This time, however, we do not concern ourselves with auxiliary variables, and the dimension of our state space is simply $D = |T|$, the number of session types we are considering. We sacrifice exactness in order to reduce the number of states. As before we let n_t represent the number of sessions of type t, and we write a given state as $\underline{s}_{approx} = \underline{n} = (n_{t_1}, n_{t_2}, ..., n_{t_D})$. The order of the n_t's becomes important this time. There is a restriction that n_{t_1} precedes n_{t_2} if session type t_1 has fewer links than session type t_2. That is, the elements are grouped according to their session length and placed in increasing order. Let the number of session types using only 1 link (ie. in the first group) be I. Now we will fix the order and write $\underline{s}_{approx} = (n_{t_1}, n_{t_2}, ..., n_{t_D})$ as $\underline{s}_{approx} = (n_1, n_2, ..., n_D)$ so that the indices on the elements are easier to work with. Let $\underline{n}_i^j$ represent the vector consisting of the consecutive elements $n_i, n_{i+1}, ..., n_j$. The state space, S_{approx}, for the process $\underline{s}_{approx}$ is identical to the exact case from section 3.1 above except that there are no auxiliary variables, so we ignore the second part of the exact state space definition.

Given $\underline{n}_{I+1}^{D}$ we approximate the conditional stationary distribution of the types $1, 2, ..., I$ as product form:

$$\Pi(\underline{n}_1^I \mid \underline{n}_{I+1}^D) = G^{(0)}(\underline{n}_{I+1}^D)\frac{\lambda_1{}^{n_1}}{n_1!}\frac{\lambda_2{}^{n_2}}{n_2!}...\frac{\lambda_I{}^{n_I}}{n_I!}, \tag{3.5}$$

where $G^{(0)}(\underline{n}_{I+1}^D)$ is a normalization constant. Notice that the normalizing constant depends on $\underline{n}_{I+1}^D$. Next, we consider each of the $D - I$ remaining session types, in turn, as a birth-death process, and proceed to calculate $\Pi(\underline{n}_1^D)$ as:

$$\Pi(\underline{n}_1^D) = \Pi(\underline{n}_1^I \mid \underline{n}_{I+1}^D)\Pi(n_{I+1} \mid \underline{n}_{I+2}^D)...\Pi(n_{I+i} \mid \underline{n}_{I+i+1}^D)...\Pi(n_{D-1} \mid \underline{n}_D^D)\Pi(n_D) \tag{3.6}$$

in $D - I$ stages. At stage i we consider session type $I + i$. Given $\underline{n}_{I+i+1}^D$, we can determine both the birth rate and death rate when there are currently $n_{I+i} = j$ sessions in progress. The death rate is simply j, and the arrival rate can be calculated as:

$$\gamma_j^{(I+i)} = \lambda_{I+i}P(A_j^{(I+i)}) = \lambda_{I+i}\sum_{\underline{n}_1^{I+i-1}} R^{(I+i)}(\underline{n}_1^D)\Pi(\underline{n}_1^{I+i-1} \mid \underline{n}_{I+i}^D) \tag{3.7}$$

where $A_j^{(I+i)} = \{$type $I + i$ session is accepted given there are j type $I + i$ sessions in progress and $\underline{n}_{I+i+1}^D\}$ and $R^{(I+i)}(\underline{n}_1^D) = P($type $I + i$ session is accepted $\mid \underline{n}_1^D)$. The latter is a straightforward combinatorial expression [18]. Note that it is this expression that changes depending on the number and placement of wavelength converters. For example, if $\underline{n}_1^D = (n_1, n_2, n_3, n_{12}, n_{23}, n_{123})$ and there are no converters we have

$$R^{(4)}(\underline{n}_1^D) = 1 - \frac{\binom{n_{23}+n_2}{F-n_{123}-n_{12}-n_1}}{\binom{F-n_{123}-n_{12}}{n_1}}. \tag{3.8}$$

$\Pi(\underline{n}_1^{I+i-1} \mid \underline{n}_{I+i}^D)$ was calculated in the previous stage. This yields

$$\Pi(\underline{n}_{I+i} \mid \underline{n}_{I+i+1}^D) = G^{(I+i)}(\underline{n}_{I+i+1}^D)\frac{\gamma_0^{(I+i)}\gamma_1^{(I+i)}\cdots\gamma_{n_{I+i}-1}^{(I+i)}}{n_{I+i}!}, \tag{3.9}$$

as the stationary distribution of the birth-death process[11] at stage i, and hence

$$\Pi(\underline{n}_1^{I+i} \mid \underline{n}_{I+i+1}^D) = \Pi(\underline{n}_1^{I+i-1} \mid \underline{n}_{I+i}^D)\Pi(\underline{n}_{I+i} \mid \underline{n}_{I+i+1}^D) \tag{3.10}$$

to be used at stage $i+1$.

Theorem 3.4

The approximate method above is exact when applied to the CS case.

Proof

First we notice a key fact:

Observation:

The CS case is special because we can take any set of session types, ie. any subset $T' = \{t'_1, t'_2, \ldots, t'_{|T'|}\}$ of T, form a Markov process, ie. the process $\underline{n}'_{cs} = (n'_{t'_1}, n'_{t'_2}, \ldots, n'_{t'_{|T'|}})$ where the capacities on each of the K links are specified, and we immediately know that the stationary distribution is product form, ie.:

$$\Pi'_{cs}(\underline{n}'_{cs}) \propto \frac{\lambda_{t'_1}^{n'_{t'_1}}}{n'_{t'_1}!}\frac{\lambda_{t'_2}^{n'_{t'_2}}}{n'_{t'_2}!}\cdots\frac{\lambda_{t'_{|T'|}}^{n'_{t'_{|T'|}}}}{n'_{t'_{|T'|}}!}. \tag{3.11}$$

Now, recall that we calculate $\Pi(\underline{n}_1^D)$ using equation (3.6), and consider the terms of the product on the right hand side.

Our observation above justifies the product form of the process $(n_1, n_2, \dots, n_I)$ where the link capacities are suitably reduced by the number of session types $I+1, \dots, D$. Since

$$\Pi(\underline{n}_1^I \mid \underline{n}_{I+1}^D) = \frac{\Pi(\underline{n}_1^D)}{\Pi(\underline{n}_{I+1}^D)} \propto \frac{\frac{\lambda_1{}^{n_1}}{n_1!} \cdots \frac{\lambda_D{}^{n_D}}{n_D!}}{\frac{\lambda_{I+1}{}^{n_{I+1}}}{n_{I+1}!} \cdots \frac{\lambda_D{}^{n_D}}{n_D!}} = \frac{\lambda_1{}^{n_1}}{n_1!} \cdots \frac{\lambda_I{}^{n_I}}{n_I!} \propto \Pi(\underline{n}_1^I) \tag{3.12}$$

we are correct to assume in the approximate method that the first term is product form in the CS case, suitably normalized given that the capacities of the links are effectively reduced by the numbers of the session types $I+1, \dots, D$.

For the remaining terms of the form $\Pi(n_i \mid n_{i+1}, \dots, n_D)$ our observation also tells us that

$$\Pi(n_i \mid \underline{n}_{i+1}^D) = \frac{\Pi(\underline{n}_i^D)}{\Pi(\underline{n}_{i+1}^D)} \propto \frac{\frac{\lambda_i{}^{n_i}}{n_i!} \cdots \frac{\lambda_D{}^{n_D}}{n_D!}}{\frac{\lambda_{i+1}{}^{n_{i+1}}}{n_{i+1}!} \cdots \frac{\lambda_D{}^{n_D}}{n_D!}} = \frac{\lambda_i^{n_i}}{n_i!} \propto \Pi(n_i). \tag{3.13}$$

That is, the conditional process $(n_i \mid n_{i+1}, \dots, n_D)$ has the same stationary distribution as the Markov process (n_i) where the capacities of the links are correctly adjusted because of the numbers of the session types $I+i+1, \dots, D$. At this point we see that it is sufficient to calculate (n_i), for $i = I+1, \dots, D-1$, and similarly (n_D) in the last step.

Now, although we know the exact form of the stationary distribution of (n_i) for $i = I+1, \dots, D$, we choose to calculate it in a less efficient way. We do this because the less efficient way can be generalized to non CS cases. In the approximate method we actually calculate (n_i) as the marginal distribution of $(\underline{n}_1^i)$ by conditioning on $\Pi(\underline{n}_1^{i-1})$. Note that we actually use $\Pi(\underline{n}_1^{i-1} \mid \underline{n}_i^D)$ in place of $\Pi(\underline{n}_1^{i-1})$, but once again this is equivalent by our observation. Indeed (n_i) is correctly modeled by a birth-death process where the birth and death rates are determined by conditioning on the values of $n_1, \ldots, n_{i-1}$. $\diamond$

Note that the above proof relies heavily on one key observation that is only true in the CS case. In fact, there are many counter examples to the exactness of the approximate method with the state space of dimension $D = |T|$ in the other cases. In any non CS case you can numerically verify that $\Pi_{approx}(\underline{n}_1^D) \neq \Pi_{exact}(\underline{n}_1^D) = \sum_{\underline{x}} \Pi_{exact}(\underline{n}_1^D, \underline{x})$, where Π_{approx} and Π_{exact} are the stationary distributions of the approximate and exact models of sections 3.2 and 3.1 respectively.

Without the aid of the following iterative algorithm, the above calculations are actually infeasible for even moderate values of F.

<u>Approximate Iterative Algorithm</u>

1. Take a guess for Π_0 (for example uniform), and set $k = 1$.
2. Calculate all conditional distributions from Π_{k-1} needed in equation (3.7) for determining the birth rates in stages 2 to $D - I$ of the approximate method.
3. Using the conditional distributions from the previous step in equation (3.7) when necessary, calculate a new approximation, Π_k, from the equations in the approximate method.
4. If $|\Pi_k - \Pi_{k-1}| < E$, where E is a predefined allowed error, stop, else set $k = k+1$ and go to 2.

This algorithm appears to converge for the processes we have defined in this application. Moreover, it runs very quickly; five to ten iterations are usually enough to ensure five decimal places or more accuracy.

Overall, the approximate method is quite good. At lower loads, resulting in lower and more realistic blocking values for the session types, the approximate method is often comparable to the exact method. It is only when the loading scheme on the K links becomes very heavy that the approximation begins to break down.

Chapter 4

Network Level Analysis

4.1 A Recursive Algorithm for L-Hop Blocking

At some point it will not be possible to analyze long paths, either exactly or approximately, using the models of the previous chapter. For that reason we develop a recursive expression, based on an exact $K = 3$ link model with the 6 distinct session types, to approximate the blocking. It should be noted that this recursion is an analogous extension of the recursion in [22] in some respects. There are, however, significant differences. The recursion presented here is based on a 3 link base unit, as opposed to a 2 link base unit in [22], and so the correlation of load on consecutive links is better represented. Moreover, at each stage of this recursion the base unit is exact, taking into account both a general loading scheme and the converter placement, while in [22], a CS 2 link base unit with symmetric loading is used.

At each stage in the recursion we will view an l-hop path in 2 separate ways.

On one hand, our goal is to calculate the blocking on all links, so the recursion at

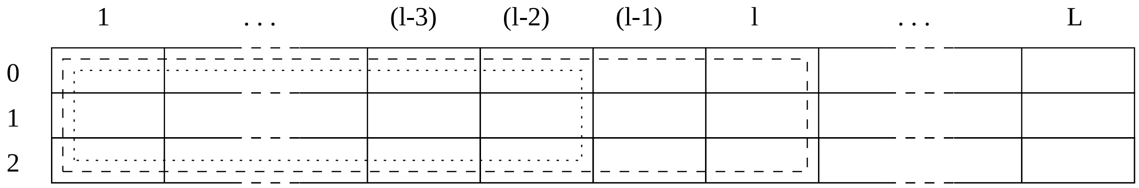

Figure 4.1: First View of an L-Hop Path

stage l is defined in terms of events on the first l links. We let $B^{(l)}(n_{123f}, n_{3f}, n_{2f})$ be the probability on an l-hop path that there are n_{123f} wavelengths free on all links, n_{3f} wavelengths free on the last link, and n_{2f} wavelengths free on the second last link. Similarly x_{123f} will refer to the number of wavelengths free on the first $(l-2)$ links, x_{3f} will be the number of wavelengths free on the $(l-2)$nd link, and x_{2f} the number of wavelengths free on the $(l-3)$rd link. We calculate $B^{(l)}(n_{123f}, n_{3f}, n_{2f})$ recursively by conditioning on the status of the first $(l-2)$ links. Figure 4.1 depicts the recursion at stage l. The dashed line shows the first l links together, while the dotted line points out the first $(l-2)$ links that are conditioned on.

On the other hand, we have to employ our base $K = 3$ exact model to calculate $B^{(l)}(n_{123f}, n_{3f}, n_{2f})$, and in order to do this we need to select 3 links from the path and determine the arrival rates (equivalently, the loads) of the 6 session types as in section 3.1. We will see below that it will be appropriate to use 2 different sets of 3 links, depending on what we want to calculate. The 2 sets are highlighted in figure 4.2. We will define the exact processes on the first set and the second set to be $\underline{s}_{exact}^{(1)}$ and $\underline{s}_{exact}^{(2)}$ respectively.

Now, using $\underline{n}_f = (n_{123f}, n_{3f}, n_{2f})$ and $\underline{x}_f = (x_{123f}, x_{3f}, x_{2f})$ we write

$$B^{(l)}(\underline{n}_f) = \sum_{\underline{x}_f} P(\underline{n}_f \mid \underline{x}_f) B^{(l-2)}(\underline{x}_f). \tag{4.1}$$

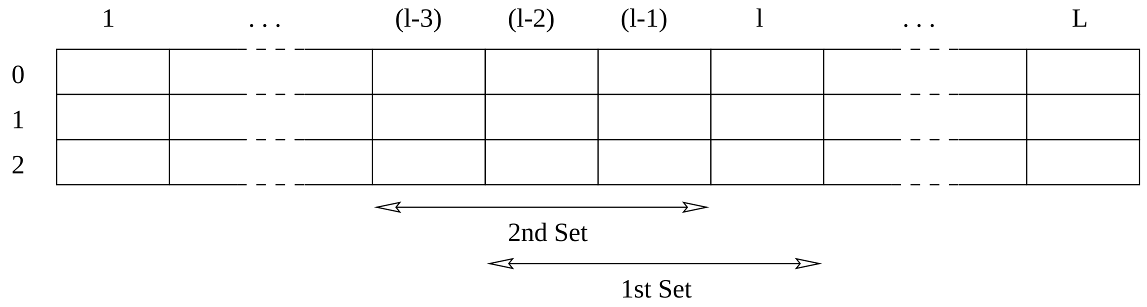

Figure 4.2: Second View of an L-Hop Path

where

$$
\begin{aligned}
P(\underline{n}_f \mid \underline{x}_f) &= P(n_{123f} \mid n_{3f}, n_{2f}, x_{123f}, x_{3f}, x_{2f}) \\
&\quad \times P(n_{3f} \mid n_{2f}, x_{123f}, x_{3f}, x_{2f}) P(n_{2f} \mid x_{123f}, x_{3f}, x_{2f}). \qquad (4.2)
\end{aligned}
$$

We calculate $P(\underline{n}_f \mid \underline{x}_f)$ by approximating each of the 3 terms in the product above.

$$
\begin{aligned}
P(n_{123f} \mid n_{3f}, n_{2f}, x_{123f}, x_{3f}, x_{2f}) &\approx P(n_{123f} \mid n_{3f}, n_{2f}, x_{123f}) \qquad (4.3) \\
&= \sum_{n_{12}, n_{23}, n_{123}} \Big[P(n_{123f} \mid n_{3f}, n_{2f}, x_{123f}, n_{12}, n_{23}, n_{123}) \\
&\quad \times P(n_{12}, n_{23}, n_{123} \mid n_{3f}, n_{2f}, x_{123f}) \Big] \qquad (4.4)
\end{aligned}
$$

Depending on the converter placement in the path, $P(n_{123f} \mid n_{3f}, n_{2f}, x_{123f})$ may be calculated directly, as in equation (4.3), or by further conditioning, as in equation (4.4). In either case, we find $P(n_{123f} \mid n_{3f}, n_{2f}, x_{123f})$ using $(\underline{s}^{(1)}_{exact})$ by assuming the first $(l-2)$ links of the path are accounted for by the first link of the process. If $\underline{s}^{(1)}_{exact}$ contains an auxiliary variable x_{123} we can simply use $\Pi(\underline{s}^{(1)}_{exact})$ directly since x_{123} is equivalent to n_{123f} (also n_{3f} is equivalent to $F - n_{123} - n_{23} - n_3$, n_{2f} is equivalent to $F - n_{123} - n_{23} - n_{12} - n_2$ and x_{123f} is equivalent to $F - n_{123} - n_{12} - n_1$). Otherwise, we must further condition on the values of n_{12}, n_{23} and n_{123}. In the later case, $P(n_{123f} \mid n_{3f}, n_{2f}, x_{123f}, n_{12}, n_{23}, n_{123})$ reduces to a combinatorial expression and $P(n_{12}, n_{23}, n_{123} \mid n_{3f}, n_{2f}, x_{123f})$ can be found using $\Pi(\underline{s}^{(1)}_{exact})$.

Similarly

$$P(n_{3f} \mid n_{2f}, x_{123f}, x_{3f}, x_{2f}) \approx P(n_{3f} \mid n_{2f}, x_{3f}) \tag{4.5}$$

can be calculated from $\Pi(\underline{s}^{(1)}_{exact})$ assuming n_{3f} is equivalent to $F - n_{123} - n_{23} - n_3$, n_{2f} is equivalent to $F - n_{123} - n_{23} - n_{12} - n_2$ and x_{3f} is equivalent to $F - n_{123} - n_{12} - n_1$.

Finally,

$$P(n_{2f} \mid x_{123f}, x_{3f}, x_{2f}) \approx P(n_{2f} \mid x_{3f}, x_{2f}) \tag{4.6}$$

can be calculated from $\Pi(\underline{s}^{(2)}_{exact})$ directly assuming n_{2f} is equivalent to $F - n_{123} - n_{23} - n_3$, x_{3f} is equivalent to $F - n_{123} - n_{23} - n_{12} - n_2$ and x_{2f} is equivalent to $F - n_{123} - n_{12} - n_1$.

For this second order recursion we need to specify 2 starting values.

$$B^{(3)}(n_{123f}, n_{3f}, n_{2f}) = \sum_{\underline{s}^{(1)}_{exact}} P(n_{123f}, n_{3f}, n_{2f} \mid \underline{s}^{(1)}_{exact}) \Pi(\underline{s}^{(1)}_{exact}) \tag{4.7}$$

where $P(n_{123f}, n_{2f}, n_{1f} \mid \underline{s}^{(1)}_{exact}) = P(n_{123f}$ wavelengths free on the first three links n_{3f} wavelengths free on the third link, n_{2f} wavelengths free on the second link $\mid$ currently in state $\underline{s}^{(1)}_{exact})$. $B^{(3)}(n_{123f}, n_{3f}, n_{2f})$ can be found in a similar manner as $P(n_{123f} \mid n_{3f}, n_{2f}, x_{123f})$. That is, it may be found directly from $\Pi(\underline{s}^{(1)}_{exact})$ if x_{123} is defined, or we may need to further condition on the state of the links as indicated in equation (4.7). In the latter case, the probability $P(n_{123f}, n_{3f}, n_{2f} \mid \underline{s}_{exact}^{(1)})$ will be a combinatorial expression.

$$B^{(2)}(n_{12f}, n_{2f}, n_{1f}) = \sum_{\underline{s}_{exact}^{(1)}} P\{n_{12f}, n_{2f}, n_{1f} \mid \underline{s}_{exact}^{(1)}\}\Pi(\underline{s}_{exact}^{(1)}) \tag{4.8}$$

where $P(n_{12f}, n_{2f}, n_{1f} \mid \underline{s}_{exact}^{(1)}) = P(n_{12f}$ wavelengths free on the first two links n_{2f} wavelengths free on the second link, n_{1f} wavelengths free on the first link | currently in state $\underline{s}_{exact}^{(1)})$. This time if $(\underline{s}_{exact}^{(1)})$ contains an auxiliary variable x_{12} we can simply use $\Pi(\underline{s}_{exact}^{(1)})$ directly to find $P(n_{123f}, n_{3f}, n_{2f} \mid \underline{s}_{exact}^{(1)})$ since x_{12} is equivalent to n_{12f} (also n_{2f} is equivalent to $F - n_{123} - n_{23} - n_{12} - n_2$ and n_{1f} is equivalent to $F - n_{123} - n_{12} - n_1$), otherwise, we must further condition as indicated in equation (4.8). Once again, in the latter case, the probability $P(n_{12f}, n_{2f}, n_{1f} \mid \underline{s}_{exact}^{(1)})$ will be a combinatorial expression.

Now, using $B^{(L)}(\underline{n}_f)$ we know the probability of blocking, say $Q^{(L)}$, for a session of length L:

$$Q^{(L)} = \sum_{n_{3f}=0}^{F} \sum_{n_{2f}=0}^{F} B^{(L)}(0, n_{3f}, n_{2f}). \tag{4.9}$$

4.2 Network-Wide Blocking

The theory developed in sections 3.1, 3.2, and especially 4.1 enables us to look at a more global picture of the network. Now that we can determine the blocking on arbitrary paths, we are in a position to analyze various network topologies. Given information about physical connectivity, routes, loading and wavelength converter placement, we can systematically analyze each route and determine an overall measure of the network blocking. We assume that all sessions considered have a prescribed route, and that the static average load for a given session (or equivalently a given route) is known. More precisely, we require the following information:

1. A graph, $G = (V, E)$, representing the network topology.
2. An ordered set of network routes, R. (Every route with non zero load listed.)
3. An ordered set of loads, Λ. (Ordered to correspond a load with a route from R. Note that the load to a route is simply the sum of arrival rates of all sessions that use the route since the session holding times are always assumed to have unit means.)
4. A subset W of V representing those nodes in G with wavelength converters.

We will proceed to calculate the blocking for each route in turn using the models from sections 3.1 or section 3.2 when possible, and the recursion from section 4.1 when otherwise necessary. The network loading, given by R and Λ, together with the wavelength converter positions, given by W, allows us to determine the appropriate parameters for the models or base units of the recursion. For each route $r \in R$ with length L_r we can determine the probability of blocking for that route, say $Q^{(L_r)}(r)$, where $Q^{(L_r)}(r) = Q^{(L_r)}$ applied to route r. With this set $Q = \{Q^{(L_r)}(r)\}$ of route blocking values, we can define a weighted network-wide blocking:

$$U_{nwb} = \frac{\sum_r \lambda_r Q^{(L_r)}(r)}{\sum_r \lambda_r} \tag{4.10}$$

where $\lambda_r \in \Lambda$ is the load on route r.

This provides us with a measure of the overall network performance that will allow us to gauge the effects of changing loads and converter placement.

Note that this analysis overlooks the actual dependence between Λ and Q. That is, we use the offered load Λ to determine Q and implicitly assume when calculating $Q^{(L_r)}(r)$ that no sessions are blocked on routes outside of r. In reality, when blocking occurs on links outside of r it reduces the load on route r, and this in turn affects the blocking $Q^{(L_r)}(r)$. An exact analysis would have to account for this relationship and perhaps iteratively converge to the equilibrium point for blocking as a function of load. In most of the examples of interest, however, the blocking is very small, and hence, would result in an insignificant adjustment to the load Λ. For this reason we ignore this relationship and calculate a highly accurate result based on Λ, the offered load.

Because the wavelength converter placement, W, is not needed until after we already have a complete picture of the network loading through G, R, and Λ, we can do more than just input the placement of wavelength changing devices. We can actually use the given information to propose W. That is, W may be a function of G, R and Λ. This idea will be explored in the next chapter.

Chapter 5

Converter Placement Heuristics

Although we can explicitly specify which nodes are capable of converting from some wavelength to another, as we have seen in section 4.2, it is more interesting to create algorithms that assign converters to some subset of the nodes in the network based on our intuition about where these devices would benefit most. In [22] the authors assign a converter to a given node with probability q. Although this does attempt to parameterize the total number of converters in the network, it does not realistically reflect how the converters would be placed. A better approach is to base the decision on the overall load at the node. Because a wavelength changer is most useful when a node is carrying a lot of traffic, we simply argue that the devices should be placed at the heaviest loaded nodes in the network. To determine which nodes are the heaviest loaded we calculate a measure of the load, say m_v for each node $v \in V$, as follows:

<u>Calculating m_v</u>:

1. For each route $r \in R$, determine if v is a member of the route.

2. If v belongs to r and is not either a source or destination node, add λ_r to m_v.

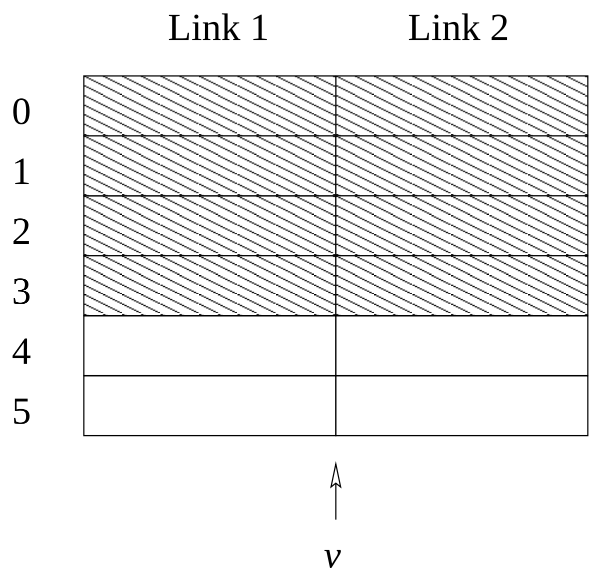

Figure 5.1: 4 Sessions Using Node v Not at Endpoints

3. If v belongs to r and is either a source or destination node, add $w\lambda_r$ to m_v, where $0 < w < 1$.

The idea behind the above algorithm for calculating m_v is based on the average number of resources (in terms of load) used by routes that contain v. A route that uses v not at an end point certainly reduces the number of free wavelengths for additional sessions, and hence, increases the need for a wavelength changer. A route that uses v at an endpoint, as a source/destination, may or may not reduce the number of free wavelengths because a session on that route may or may not share a wavelength with some other session, using v as a destination/source. That is, 2 sessions that use v at endpoints, one as a source and the other as a destination, may use the same wavelength and thus together these routes should not contribute as much as the sum of their loads to the score. To see this more clearly, compare figures 5.1 and 5.2.

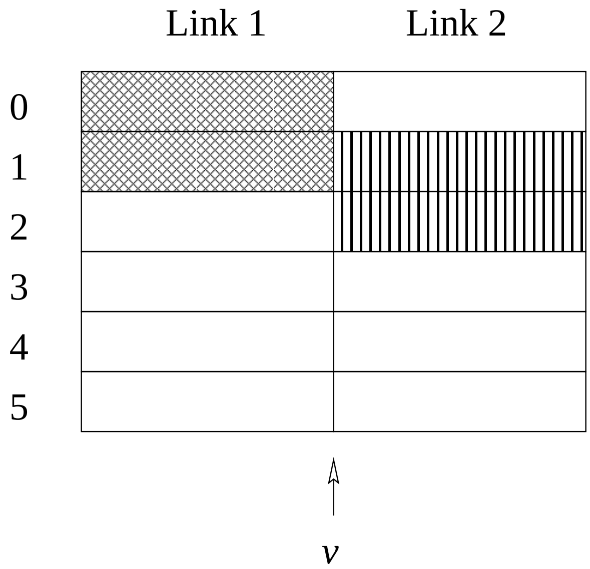

Figure 5.2: 4 Sessions Using Node v at Endpoints

Both pictures show 4 sessions in progress on two links adjacent to v with $F = 6$. In figure 5.1 all of the sessions must use a wavelength, for a total of 4, since they use v not at an endpoint of their path. In contrast, figure 5.2 shows us that when the sessions use v at endpoints they may take fewer than 4 wavelengths total, saving resources for other sessions. Therefore the relative contributions of these two types will be 1 and w, respectively, where $0 < w < 1$. A reasonable value for w is $\frac{3}{4}$ by the following argument. Imagine there are two routes that use v at an endpoint and have similar load. On average, one of the routes will use v as a source and the other will use v as a destination, and so they may share the same wavelength. Hence, the 2 routes together use either 1 or 2 wavelengths, and on average they require $\frac{3}{2}$ wavelengths. Therefore each route should contribute $w = \frac{1}{2}(\frac{3}{2}) = \frac{3}{4}$ to m_v.

With $\Lambda_i(v)$ as the sum of the loads of routes that use v at an intermediate point (not an endpoint) and $\Lambda_e(v)$ as the sum of the loads of routes that use v as an endpoint we have simply:

$$m_v = \Lambda_i(v) + w\Lambda_e(v). \tag{5.1}$$

Note that many other measures based on the available information, and especially G, R and Λ, can be defined.

Now, once we have established a score, m_v, for each node, we can decide on W with various things in mind. Most obvious, since the score was attempting to measure the need for a converter in terms of resources used, we can assign converters only to nodes whose score is above some threshold value, say $\alpha \geq 0$. We will assume α is a percentage, where F represents 100%. We may further only assign a converter when $\Lambda_i(v)$ is above another threshold value, say $\beta \geq 0$, also as a percent, to ensure that converters are not assigned to nodes that only belong to routes at endpoints. Alternatively, we may decide that we have a fixed number of converters, say C, to place (perhaps because that is the number of devices we can afford) and so we simply take the C nodes with the largest scores. Most generally, our criteria may insist that the nodes pass all of these filters simultaneously. That is, we may only select nodes with scores that pass the specified thresholds α and β, and then only up to the C nodes with largest scores.

Once W has been established we can analyze the network as in section 4.2 of chapter 4 and check its performance. In fact, it is this feature that makes this model useful when it comes to sparse wavelength conversion and converter placement. The proposed heuristics are barely more than common sense, but it is important that we can analyze the placement once it is established using the analytical model. Moreover, newer and more elaborate heuristics can be used as well. A network designer may compare several different policies for converter placement on a given network, corresponding to the different filter parameters α, β and C or their own intuition, to produce an optimal performing configuration.

Chapter 6

Examples

In this chapter we present some numerical results to illustrate the accuracy of the model in a variety of situations. We begin with some simple paths in section 6.1 to highlight the 3 link model and the recursion. Next we consider a highly symmetric, yet very common, ring network in section 6.2. Finally, in section 6.3 we demonstrate the network analysis and wavelength converter placement heuristics using Canada's high speed research network, CA*net II.

The examples in each section use $F = 5$. Other parameters will be specified as necessary. Recall that we assumed in chapter 3 that sessions have holding times with unit means, so the load for a given session is equal to the value of its arrival rate.

The analytical calculations in this chapter were performed by computer, implemented by the author using the C++ programming language. In each example the analytical solution is compared to a computer simulation. The network simulation code was written in C by Glen Takahara. Simulating the network takes considerably longer, and it is done here strictly to validate the accuracy of the analytical model. It was found that one longer simulation run is better than the average of several shorter ones, so in the results that follow the simulated data value is from a single, yet lengthy run. For example one simulation point for the CA*net II network with moderate loading (it takes longer for lighter loading because there are fewer blocking events) takes about 3 or 4 days running on an Intel based 300MHz computer. In any sparse wavelength converter cases the converter placement was first determined using the analytical solution, and then fed into the simulation program for verification. This was necessary because the simulation code does not implement the heuristics for converter placement as described in chapter 5. The simulation was written (with a fixed input seed) so that the results are reproducible if it runs for the same length of time. However, the stopping criteria that we used was not automatic. We simply waited until the blocking values on each of the routes considered had converged in enough significant digits (usually 3 or 4) for the given loading, and then manually halted the simulation. For more detailed information about the source code of either the analytical calculations or the simulation please refer to the Appendix.

6.1 3 and 5 Hop Path Examples

This first set of examples is intended to illustrate the accuracy of our 3 link models and recursion before we move on to more complicated networks.

Figure 6.1 shows the simple 3 hop path we analyze considering all 6 possible routes. That is, there are 3 one hop routes, 2 two hop routes and 1 three hop route. To show symmetry, we take a uniform load of 0.2 on each route. Tables 6.1, 6.2 and 6.3 give the blocking data under various converter placements of NCS, Sparse, and CS

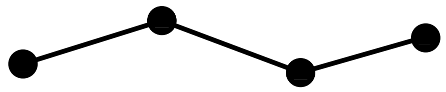

Figure 6.1: Simple 3 Hop Path with 6 Possible Routes

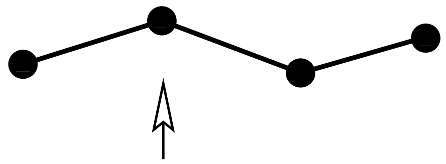

Figure 6.2: 3 Hop Path Showing Example Converter Placement

respectively. The arrow in figure 6.2 shows the placement of the single converter in the Sparse case. In each case we show both our exact and approximate model results, and compare them to those obtained by simulation.

The 3 hop path results confirm the correctness of our exact model, as well as the usefulness of the approximate model given its accuracy. Furthermore, these results give us confidence in the simulation for later examples when we will no longer have the luxury of an exact calculation for comparison.

Figure 6.3 shows the simple 5 hop path in which we analyze all 15 possible routes. Once again we use a uniform load, this time of 0.1, to each route. As before we

Type	Exact	Simulation	Approximate
Session 1	0.000295013	0.00029659963	0.000325479
Session 2	0.001118670	0.00111837996	0.001175130
Session 3	0.000295013	0.00029559328	0.000325429
Session 12	0.002069840	0.00205988815	0.002053620
Session 23	0.002069840	0.00206139671	0.002053740
Session 123	0.003422510	0.00341570202	0.003295540
Network-Wide	0.001545150	0.00154121946	0.001538160

Table 6.1: 3 Hop NCS Blocking Data

Type	Exact	Simulation	Approximate
Session 1	0.000328567	0.00032240322	0.000330371
Session 2	0.001161910	0.00115820276	0.001197190
Session 3	0.000301817	0.00030311470	0.000326549
Session 12	0.001424090	0.00142467720	0.001459480
Session 23	0.002128800	0.00213249675	0.002078180
Session 123	0.002380890	0.00237987302	0.002330720
Network-Wide	0.001287680	0.00128678833	0.001287081

Table 6.2: 3 Hop Sparse Blocking Data

Type	Exact	Simulation	Approximate
Session 1	0.000331312	0.00032936206	0.000331310
Session 2	0.001205840	0.00120607581	0.001205830
Session 3	0.000331312	0.00033084540	0.000331310
Session 12	0.001468250	0.00146707762	0.001468240
Session 23	0.001468250	0.00146690118	0.001468240
Session 123	0.001728740	0.00172946990	0.001728730
Network-Wide	0.001088950	0.00108828987	0.001088943

Table 6.3: 3 Hop CS Blocking Data

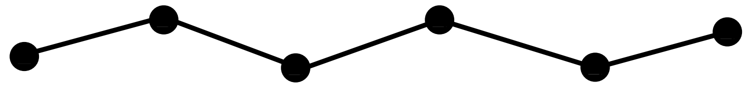

Figure 6.3: Simple 5 Hop Path with 15 Possible Routes

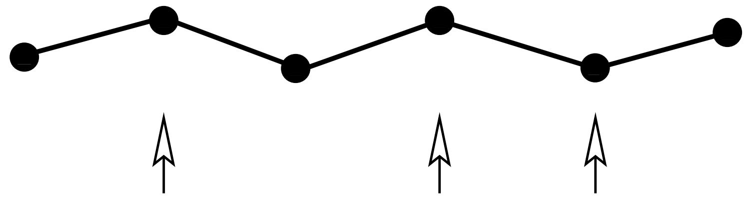

Figure 6.4: 5 Hop Path Showing Example Converter Placement

consider three cases: NCS, Sparse, CS, and the blocking results are given in tables 6.4, 6.5 and 6.6 respectively. The converter placement in the Sparse case is depicted by the arrows in figure 6.4. Notice that this time the tables only compare the recursion, based on an exact base unit, to the simulation.

The results for the 5 hop path in tables 6.4, 6.5 and 6.6 validate the recursion in a quite general state of loading and converter placement. In each of the NCS, Sparse and CS cases the calculated network-wide blocking values compare extremely well to the simulated values. Moreover, the longer hop session's calculated blocking is very close to the respective simulated blocking. It is also worth noting that even the individual trends for each session type as the number of converters increases are captured very well with the analytical model. With the exception of session types 1 and 5, which could be attributed to simulation error since the differences are so small, the calculated blocking for a given session type increases or decreases in parallel to the simulation as we move from NCS to Sparse to CS.

Type	Recursion	Simulation
Session 1	0.000112735	0.00010984277
Session 2	0.001029110	0.00097244056
Session 3	0.001828180	0.00167062068
Session 4	0.001029110	0.00097500232
Session 5	0.000112735	0.00010938944
Session 12	0.001411490	0.00134777366
Session 23	0.003531760	0.00332244067
Session 34	0.003531760	0.00330434413
Session 45	0.001411490	0.00135560057
Session 123	0.004280110	0.00404616642
Session 234	0.005679640	0.00555257769
Session 345	0.004280110	0.00402263461
Session 1234	0.006447970	0.00656805054
Session 2345	0.006159330	0.00655479752
Session 12345	0.007224420	0.00766544927
Network-Wide	0.003204660	0.00317171260

Table 6.4: 5 Hop NCS Blocking Data

Type	Recursion	Simulation
Session 1	0.000128066	0.00013155317
Session 2	0.001049580	0.00103290536
Session 3	0.001848560	0.00179226425
Session 4	0.001128680	0.00110008576
Session 5	0.000131821	0.00013108589
Session 12	0.001131430	0.00112017144
Session 23	0.003575140	0.00351517148
Session 34	0.002735090	0.00257302985
Session 45	0.001210790	0.00117482726
Session 123	0.003647760	0.00358198224
Session 234	0.004267220	0.00426248196
Session 345	0.002812350	0.00264819357
Session 1234	0.004424930	0.00432518676
Session 2345	0.004342250	0.00432507371
Session 12345	0.004470550	0.00440467823
Network-Wide	0.002460280	0.00240794656

Table 6.5: 5 Hop Sparse Blocking Data

Type	Recursion	Simulation
Session 1	0.000131821	0.00013131914
Session 2	0.001128680	0.00111023351
Session 3	0.001940010	0.00188157742
Session 4	0.001128680	0.00110834014
Session 5	0.000131821	0.00012917264
Session 12	0.001210790	0.00118474431
Session 23	0.002735080	0.00265975486
Session 34	0.002735080	0.00268337786
Session 45	0.001210790	0.00119251108
Session 123	0.002812350	0.00273382964
Session 234	0.003457610	0.00343887230
Session 345	0.002812350	0.00274781822
Session 1234	0.003590850	0.00351274361
Session 2345	0.003533840	0.00352908072
Session 12345	0.003667100	0.00359448698
Network-Wide	0.002148460	0.00210916459

Table 6.6: 5 Hop CS Blocking Data

6.2 Symmetric 8 Node Ring Example

The next example, an 8 node ring depicted in figure 6.5, is included because it is simple (highly symmetric), yet very common in practice. We assume the ring is bidirectional and shortest path routing is employed. We define 32 routes to accommodate all $\binom{8}{2}$ source destination pairs in the following way to balance load:

- 8 bidirectional one hop routes; minimum necessary
- 8 bidirectional two hop routes; minimum necessary
- 8 bidirectional three hop routes; minimum necessary
- 8 unidirectional four hop routes; only 4 bidirectional routes are actually needed, but we consider 8 clockwise routes to balance the load and retain symmetry within the network

In order to examine how the blocking changes as the load varies in a simple way, we assume all routes have the same load, ρ. The result is plotted in figure 6.6.

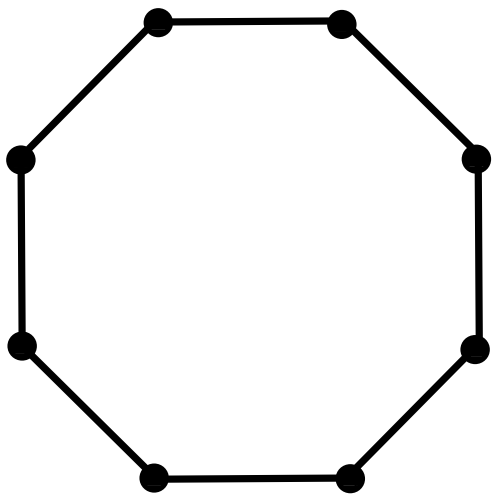

Figure 6.5: 8 Node Ring

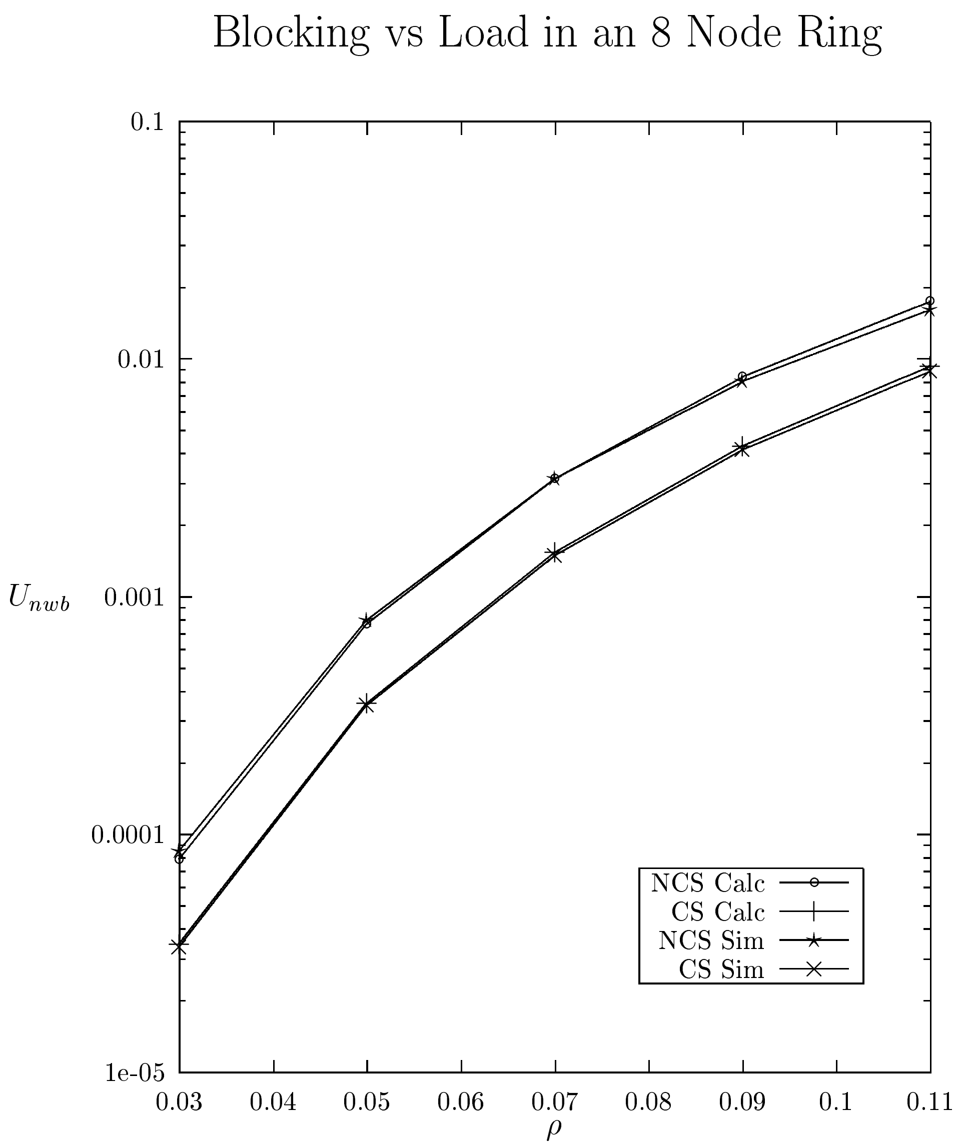

Figure 6.6: Blocking vs Load in an 8 Node Ring

The 8 node ring is our first look at a more connected and realistic network. For simplicity in presentation only, it too had a very symmetric loading scheme. The calculated and simulated values, as plotted in figure 6.6, are in excellent accordance once again, further supporting the accuracy of our analytical solution. Although only the NCS and CS cases were tested, they provide the most interesting data since we know that with all other cases of converter placement, the network-wide blocking will fall between these two extremes. That is, the NCS and CS values show the widest range in blocking among all converter placements, and therefore represent the maximum gain to be achieved by using converters in the network. In this particular example we see that the NCS blocking is approximately 2 times the CS blocking, and that the benefit increases as the load increases.

6.3 CA*net II General Network Examples

In this last section we consider a much more general network topology based on CA*net II. An illustration of the CA*net II's backbone is given in figure 6.7, while figure 6.8 shows the same topology expanded somewhat and as a planar graph.

In addition to analyzing network blocking versus load, as in section 6.2, we add another dimension in the examples to follow. We consider two different routing schemes to further compare different ways to view the network. That is, one philosophy considers the topology as strictly mesh-like, and routes between source and destination follow the shortest possible path, while the other views the network as consisting of rings. In the rings case, sessions may only pass from one ring to another at specified hand off nodes. While this tends to increase the length of routes, as compared to the

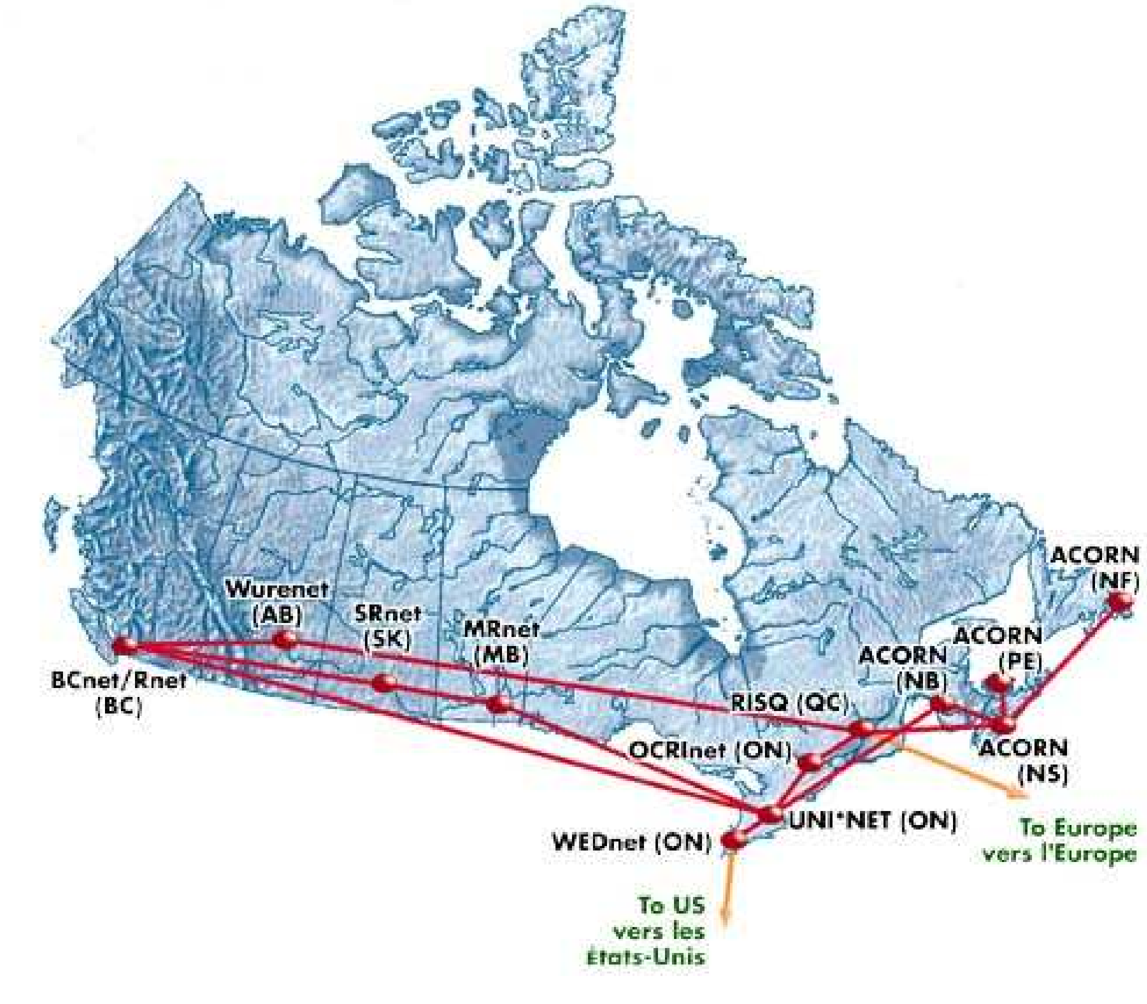

Figure 6.7: CA*net II Map

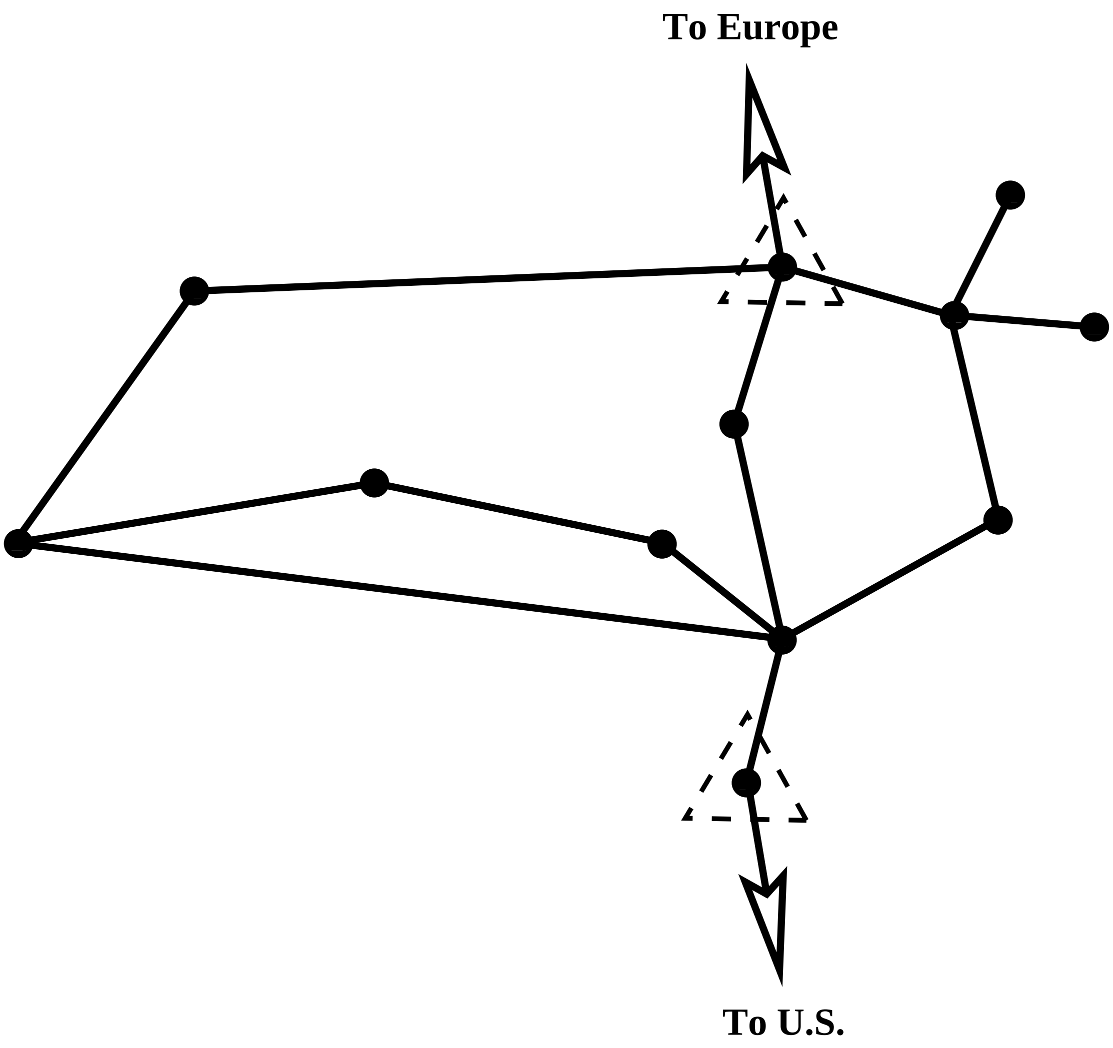

Figure 6.8: CA*net II Mesh Topology

mesh view, it is often done in practice for survivability (protection and restoration of routes).

The term *mesh routing* will refer to figure 6.8, where the $\binom{12}{2} = 66$ routes we consider are simply the shortest paths between the pair of nodes. In the event of a tie for the shortest number of hops, the physically shortest route geographically was used (refer to figure 6.7).

The term *rings routing* will refer to figure 6.9, where the $\binom{12}{2} = 66$ routes are determined from the indicated rings (thin solid lines) and specified hand off nodes (hatched boxes) in the diagram. This time the route between any pair of nodes is defined to be the shortest path using the least number of hand off nodes to go between rings. For instance, in figure 6.10 links 1 and 6 would be used as a route between node B and node G in the mesh case, whereas links 1, 2, 4 and 5 would have to be used in the rings routing case. Once again ties are broken by taking the physically shortest route as determined by the geographic picture of the network in figure 6.7.

As before, we parameterize the network load using a single variable, ρ, in order to produce simple plots of blocking versus load. In order to do this we recognize 4 different classes of routes and define a relative load weight for each class. Please notice the 2 hatched triangles in either figure 6.8 or 6.9. These triangles highlight 2 special nodes that act as link up points to other high speed networks in Europe and the United States, and will form the basis for our class divisions. Let's temporarily call the special node that provides the link up to Europe node 1, and the special link up node to the States node 2.

Class 1 consists of all routes on the graph between node 1 and 2 (there is actually only 1). Routes in this class have relative weight 4. They carry sessions between nodes 1

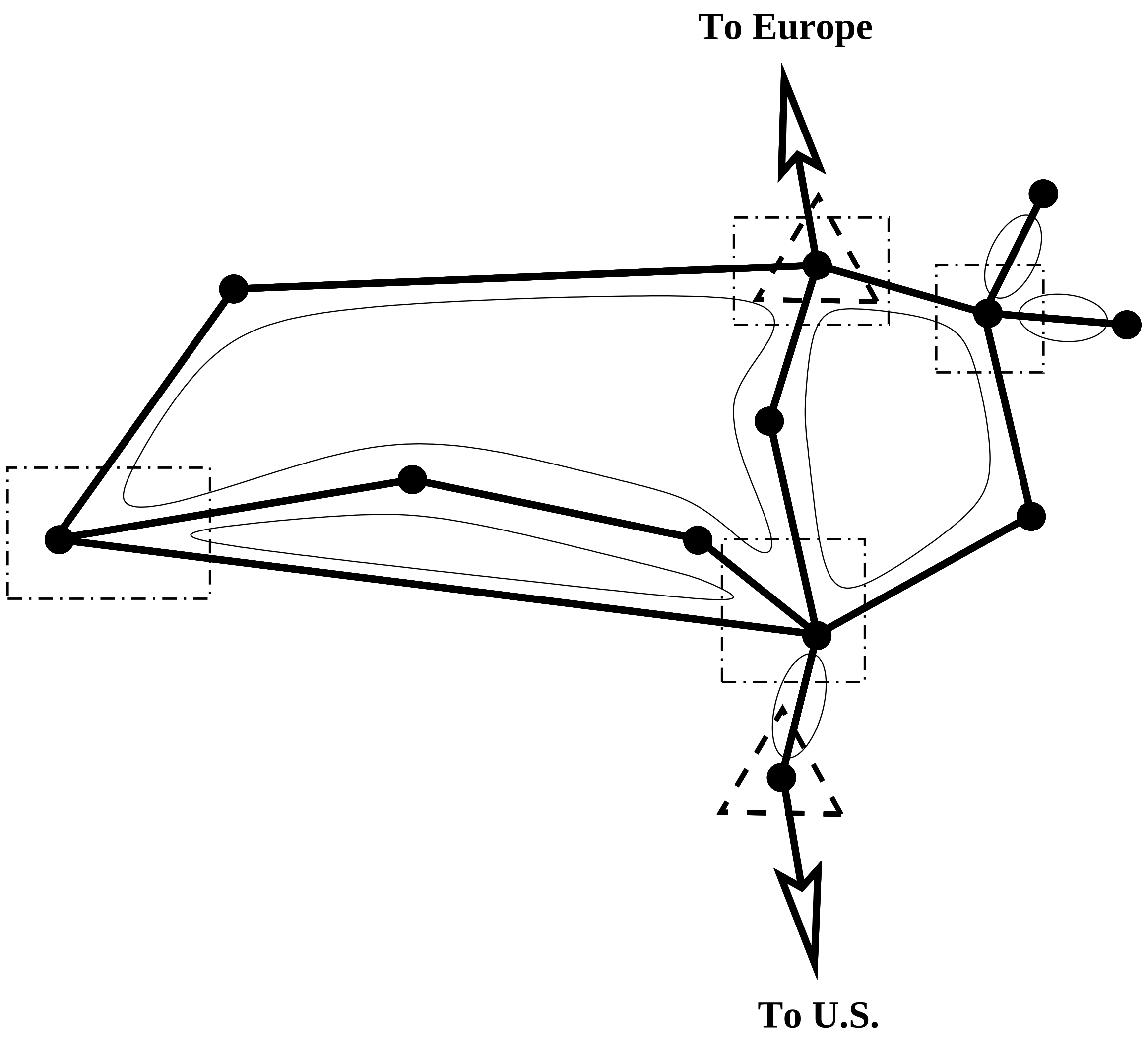

Figure 6.9: CA*net II Rings Topology

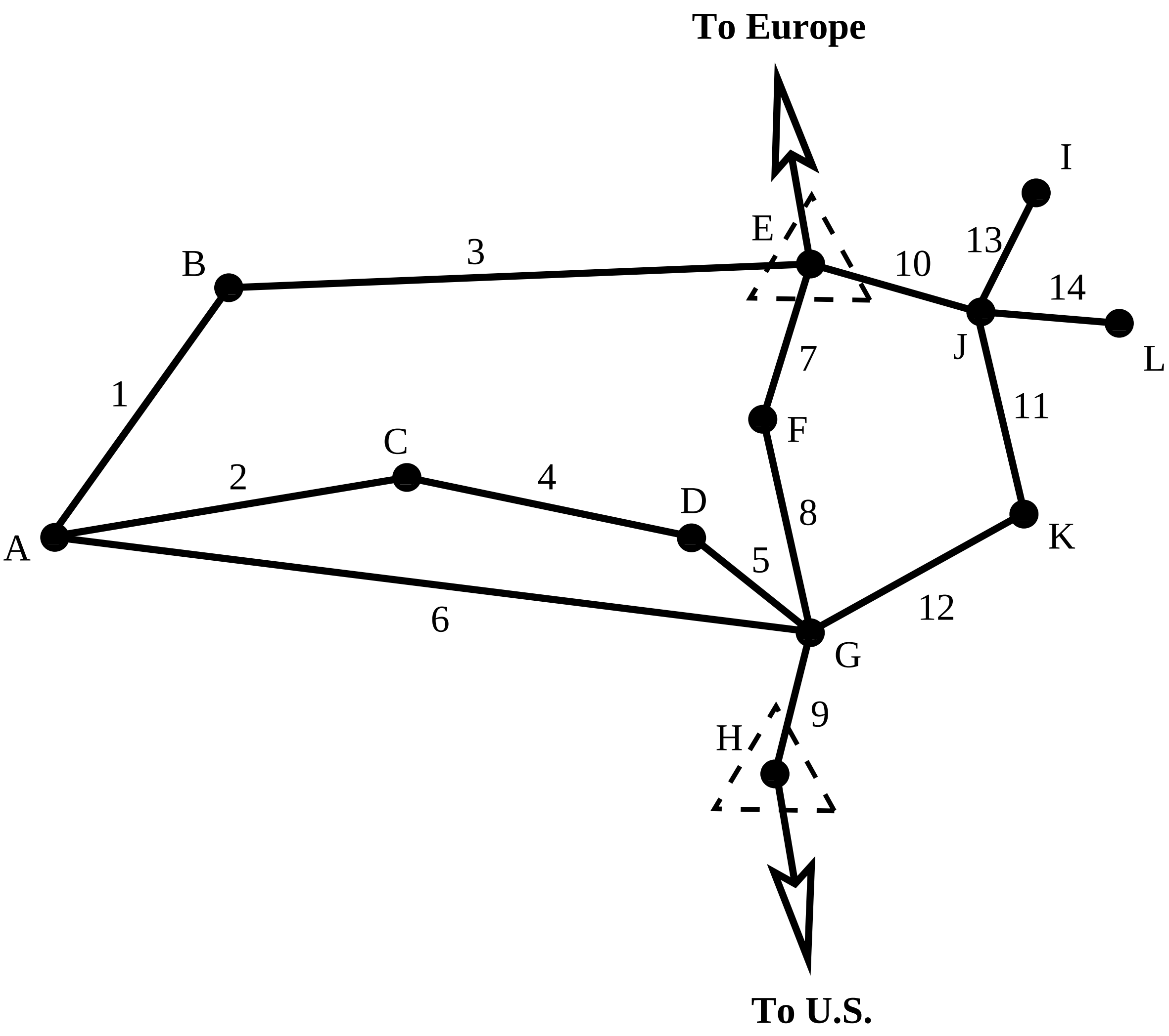

Figure 6.10: CA*net II Node and Link Labels

and 2, sessions that originate at node 1 and have a destination in the United States, sessions that originate at node 2 and have a destination in Europe, and possibly some sessions that originate in Europe and have a destination in the States and vice versa.

Class 2 consists of all remaining routes which terminate at node 2 on the graph (there are 10 of them). Routes in this class have relative weight 3. They consist of sessions that actually terminate at node 2 and sessions that have any destination in the States.

Class 3 consists of all remaining routes which terminate at node 1 on the graph (there are 10 of them). Routes in this class have relative weight 2. They consist of sessions that actually terminate at node 2 and sessions that have any destination in Europe. Notice that class 2 routes have a higher relative weight than class 3 routes because we assume there is more traffic to the United States than to Europe.

Class 4 consists of all other remaining routes (there are 45 of them). Routes in this class have relative weight 1. They consist of sessions that have both source and destination strictly within the CA*net II backbone and do not involve either the special link up node to Europe or the United States.

Now, the load associated with a given route is taken to be the product of the relative weight and the parameter ρ.

This allows us to plot the network blocking versus the parameterized load, ρ, in both the mesh and rings routing schemes. The plots are given in figures 6.11 and 6.12 respectively.

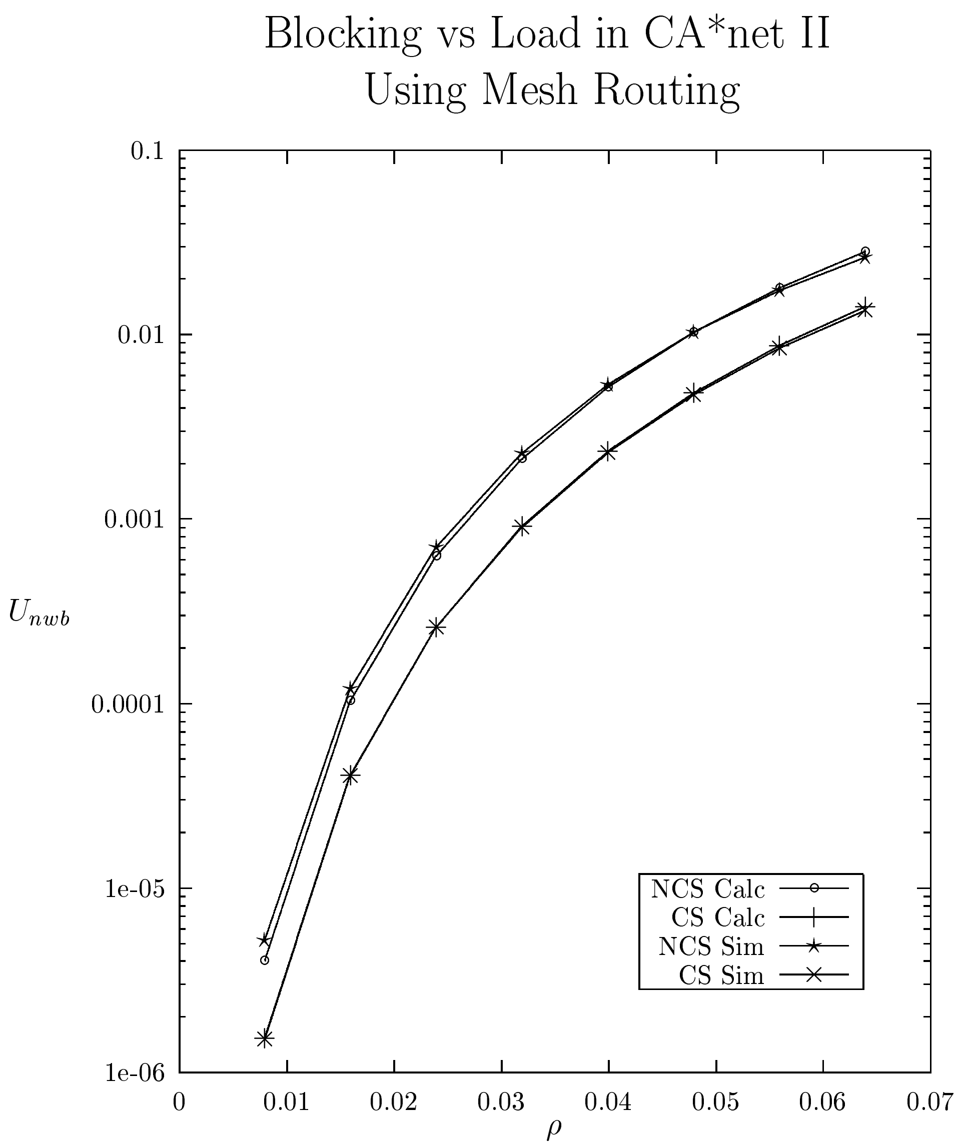

Figure 6.11: Blocking vs Load in CA*net II using Mesh Routing

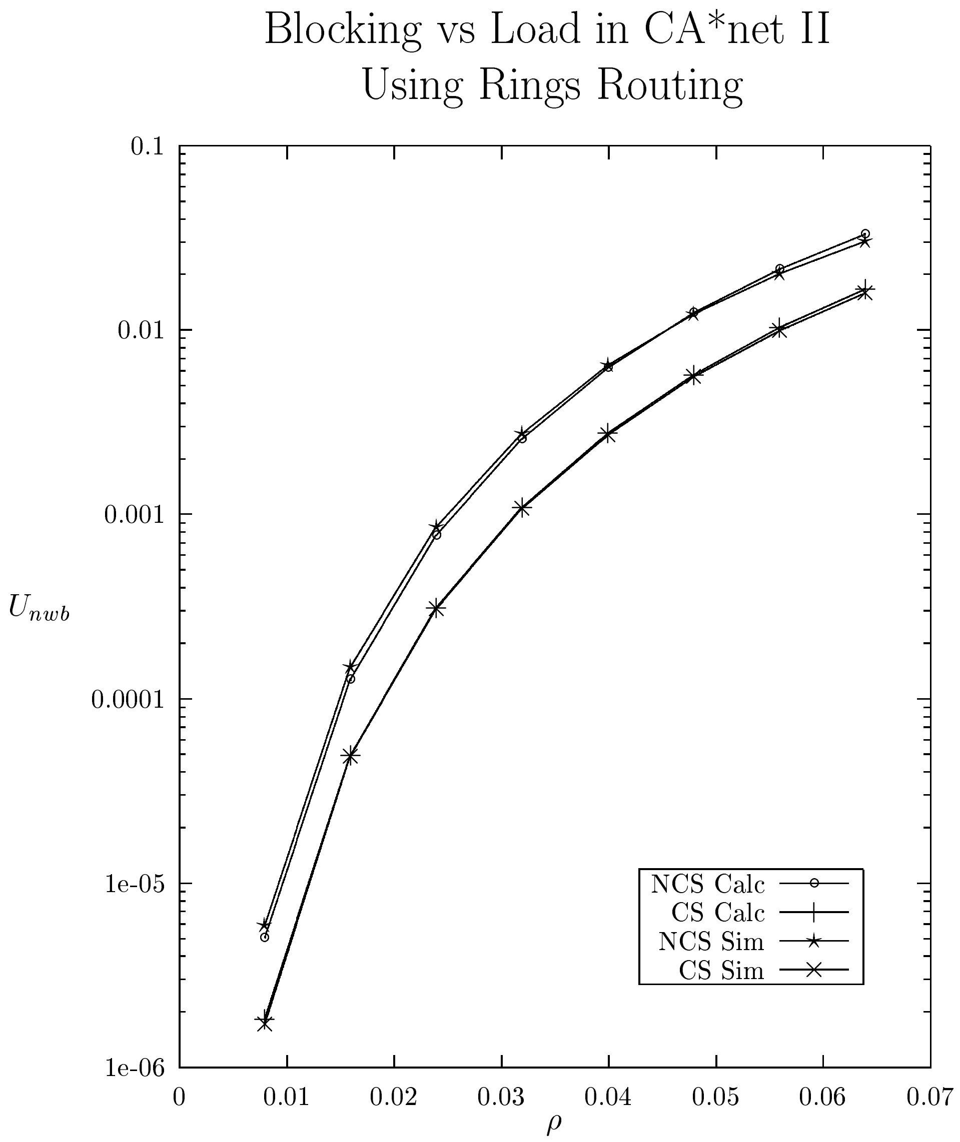

Figure 6.12: Blocking vs Load in CA*net II using Rings Routing

These examples based on the CA*net II backbone are interesting for a number of reasons:

- They are based on an actual high speed research network.
- The topology is not symmetric, as in the previous examples.
- The loading is taken to be reasonably general.
- The routing is defined in 2 separate ways, mesh and rings, and compared below.

Immediately we see that despite the generality of the previous two examples, our analytical solution models either routing situation very well. The calculated values and the simulated values are almost coincident in figures 6.11 and 6.12. Once again we can see the maximum spread between the NCS and CS cases in these figures. With either type of routing, the NCS blocking is greater than the CS blocking by a factor of about 2 to 4, depending on ρ. That is, as the load increases (or similarly if F decreases) there is a greater benefit from converters. Indeed this is not surprising since if there were an infinite supply of wavelengths available per link, and a finite amount of traffic in the network, there would never be a need for a wavelength converter. Finally, if we look closely and compare the blocking in the mesh case to the blocking in the rings case, we see that the rings routing produces blocking that is slightly higher than the mesh routing, which is what we expected given that we know the routes are longer on average in the rings routing case.

Another interesting plot is created by fixing the loading scheme and varying the number of converters in the network. This is easily accomplished by setting α to 0 (so that potentially any node can receive a converter), β to a very small but positive number (so that nodes that are only members of routes at endpoints are not assigned converters), and controlling C, the maximum number of converters allowed. Figure 6.13 shows the result under both mesh and rings routing when we take $w = \frac{3}{4}$ to calculate the score (refer to chapter 5) and use a load, parameterized as before, of $\rho = 0.024$. The heuristic assigned converters to the nodes, as labeled in 6.10, in the following order in each case:

- Mesh: G, J, E, K, F, B, D, A, C
- Rings: G, E, J, K, F, B, D, A, C

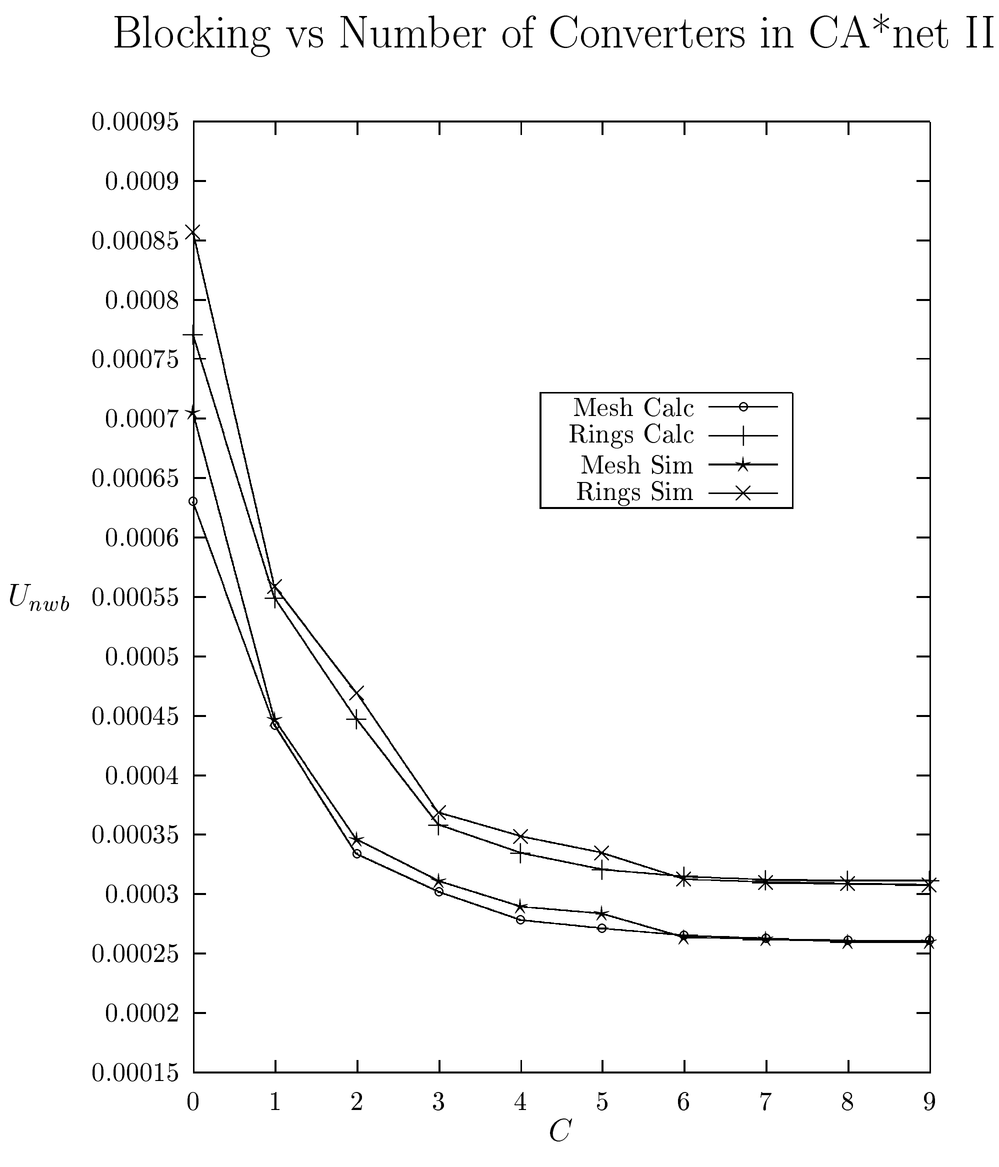

Figure 6.13: Blocking vs Number of Converters in CA*net II

Although the blocking in 6.13 is over a much smaller range, as compared with the plots from previous examples, we notice that once again the analytical solution very closely matches the simulated values. More importantly, we see that based on a heuristic with the specified parameters, the benefit of adding additional converters, in terms of reducing the network-wide blocking, is great at first but then quickly drops off. That is, figure 6.13 suggests that after 3 or 4 wavelength converters have been placed, in either the mesh or rings case, we see only slight improvement in blocking as more wavelength changing devices are added. More examples could certainly by studied to see if there is always a diminishing returns trend in general.

Chapter 7

Conclusions

7.1 Discussion of Contributions

This research was motivated by a specific application to communication networks. The analytical model was developed to study paths of an all-optical network with wavelength division multiplexing and sparse wavelength conversion. To this end, the proposed theory works very well. We can calculate blocking for a generally loaded network in a fraction of the time it takes to simulate the same network and obtain results. Moreover, in addition to analyzing a network in which the nodes with wavelength converters are given, we have developed heuristics to suggest converter placement based on topology, routing, and loading.

Ultimately, we have produced a mathematical tool that network planners could use to study networks and develop intuition about the benefit of wavelength converters in various situations. Given information about the network, in terms of the parameters introduced in chapters 3 and 4, ie. F, G, R, Λ and W, the tool can calculate the network-wide blocking, U_{nwb}. The results in chapter 6 confirm the accuracy of the analytical model. In a variety of loading schemes and wavelength converter placements the model compares very well to computer simulation. By further supplying w, α, β and C, introduced in chapter 5, a network planner may ask the tool to recalculate W and then solve for U_{nwb}. Unfortunately, because of the length of time necessary to obtain a single simulation data point, the converter placement heuristics have not been validated properly at this time. On the surface, the suggested placements seem reasonable, but a more careful evaluation is needed to determine whether the heuristics consistently predict optimal or near optimal placement.

However, with this tool at their disposal, network planners can study any of the above parameters and discover the effects on the network. The examples in chapter 6 illustrate some, but definitely not all, of the possibilities. In chapter 6 we have tried different routing schemes (varying R), and different numbers of wavelength converters (varying C), for instance, but we have not analyzed the combined effects of several parameters together.

The actual solution hinges on our K hop models presented in chapter 3, and it is these pieces that are of more mathematical significance. The stochastic processes presented there can be used to model a much more general situation than simply the specific communications application above. In order to see this clearly, we need to rephrase the work in more general terms. Let us think of the K links now as K stages that our client must pass through to be serviced. At each stage the client is confronted with a panel of F labeled servers, previously the F wavelengths. If the current panel has *extended capabilities*, the client may choose any free server for the next stage, otherwise, the client may only proceed if the same labeled server they had at the last stage is available for the next stage. Of course the *extended capabilities* compare to the notion of a wavelength converter being present at a node in the previous language of the problem. We see that this type of stochastic process is quite general, and recall that it even reduces to the CS case (every panel has *extended capabilities*) in which the client only needs some server for each stage. We have presented an exact solution to this problem for modest values of K and F by defining a suitable Markov chain. Furthermore, for larger values of these two fundamental parameters we have demonstrated an approximate solution that is also quite accurate.

7.2 Future Work

Currently, the major limiting factors to this work are the size of K and F. As each of these parameters grows, so do the sizes of our state spaces (either exact or approximate), and eventually the calculations become infeasible. Although the presented iterative algorithms run very quickly, so far we have been restricted to about $K = 3$ and $F \leq 10$ in the exact case, and $K = 4$ and $F \leq 5$ in the approximate case. We can calculate an exact $K = 2$ model up to about $F = 40$ or 50, but of course with a smaller value of K the model becomes less sensitive to the correlation of load on different links. If we need to investigate values of F that are greater than 10, however, a recursion with a 2 link exact base unit could be developed and it should perform reasonably well. In the future, it would be better to have a model that is less restricted by the values of K and F.

It would also be nice to relax some of the assumptions about the network under consideration. In particular, it would be interesting to investigate alternate routes used when the network becomes congested. In order to do this however, we would need to remove the restriction that sessions arrive to every route according to a Poisson process, as this does not properly model the bursty behavior of the traffic on the alternative paths.

Another area of investigation arises from the possibility of modifying the approximate method to produce an exact solution in non CS cases. This would most likely involve a clever state space and removing the assumption that the conditional distribution of the one hop long session types given the rest is product form. Unfortunately, our probing of this question confirms that the exact state space together with modifying the method to look at each dimension of the process in turn is not sufficient. Of course if some modification exists, we would have another efficient way of calculating the steady state distribution for any converter placement on K links. Furthermore, we believe this idea is likely general and applies to a broader range of multi-dimensional Markov processes.

Bibliography

[1] K. Bala, E. Bouillet, and G. Ellinas, "Benefits of "minimal" wavelength interchange in WDM rings", *Optical Fiber Communication Conference*, Vol. 6, OSA Technical Digest Series, 1997, pp. 120-121.

[2] R. A. Barry and P. A. Humblet, "Models of Blocking Probability in All-Optical Networks With and Without Wavelength Changers", *IEEE Journal on Selected Areas of Commununication*, Vol.14, no.5, June 1996, pp. 858-867.

[3] D. Bertsekas and R. Gallager, *Data Networks, 2nd ed.*, Prentice Hall, New Jersey, 1992.

[4] A. Birman, "Computing Approximate Blocking Probabilities for a Class of All-Optical Networks", *IEEE Journal on Selected Areas of Communication*, Vol. 14, No. 5, June 1996, pp. 852-857.

[5] R. Burden and D. Faires, *Numerical Analysis, 6th ed.*, Brooks/Cole Publishing Company, Pacific Grove, CA, 1997.

[6] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath Communications: An Approach to High Bandwidth Optical WAN's", *IEEE Transanctions on Communications*, Vol.40, no.7, July 1992, pp.1171-1182.

[7] Imrich Chlamtac, András Faragó, and Tao Zhang, "Lightpath (Wavelength) Routing in Large WDM Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 14, No. 5, June 1996, pp. 909-913.

[8] Soeren Lykke Danielsen, Benny Mikkelsen, Carsten Joergensen, Terfi Durhuus, and Kristian E. Stubkjaer, "WDM Packet Switch Architectures and Analysis of the Influence of Tuneable Wavelength Converters on the Performance", *Journal of Lightwave Technology*, Vol. 15, No. 2, February 1997, pp. 219-226.

[9] O. Gerstel, R. Ramaswami, G. Sasaki, "Benefits of Limited Wavelength Conversion in WDM Ring Networks", *Optical Fiber Communication Conference*, Vol. 6, OSA Technical Digest Series, 1997, pp. 119-120.

[10] Paul E. Green Jr., "Optical Networking Update", *IEEE Journal on Selected Areas of Communication*, Vol. 14, No. 5, June 1996, pp. 764-779.

[11] G. R. Grimmett and D. R. Stirzaker, *Probability and Random Processes, 2nd ed.*, Clarendon Press, Oxford, 1993.

[12] Michael Hluchyj and Mark Karol, "Queueing in High Performance Packet Switching", *IEEE Journal on Selected Areas of Communication*, Vol. 6, No. 9, December 1988, pp. 1587-1597.

[13] E. Kao, *An Introduction to Stochastic Processes*, Duxbury Press, Belmont, CA, 1997.

[14] M. Kovacevic and A. Acampora, "Benefits of Wavelength Translation in All-Optical Clear-Channel Networks", *IEEE Journal on Selected Areas of Communication*, Vol. 14, No. 5, June 1996, pp. 868-880.

[15] C. Y. Lee, "Analysis of Switching Networks", *Bell Systems Technical Journal*, Vol. 34, November 1955, pp. 1287-1315.

[16] Chris Leith, "A Brief Introduction to the Wavelength Changing Debate in All-Optical Networks", November 1997, available at www.mast.queensu.ca/˜cleith.

[17] Chris Leith, "Summary of All-Optical Networks with Sparse Wavelength Conversion", December 1997, available at www.mast.queensu.ca/˜cleith.

[18] C. Leith, G. Takahara, and Z. Mansourati, "Blocking For All-Optical Networks Based on Exact and Near Exact Blocking in a 3-Hop Path", CISS '99, Baltimore, United States, March 17-19, 1999, Proceedings pp. 219-224.

[19] C. Leith, G. Takahara, and Z. Mansourati, "Exact, Approximate, and Recursive Blocking in All-Optical Networks", GLOBECOM '99, Symposium on High Speed Networking, Rio de Janeiro, Brazil, December 6-10, 1999, to appear in Proceedings.

[20] Nicholas Pippenger, "On Crossbar Switching Networks", *IEEE Transactions on Communications*, Vol. COM-23, June 1975, pp. 646-659.

[21] Manish Sharma, Takeshi Ozeki, and Hiroynki Ibe, "WDM Ring Network Using a Centralized Multiwavelength Light Source and Add-Drop Multiplexing Filters", *IEEE Journal of Lightwave Technology*, Vol. 15, No. 6, June 1997, pp. 917-928.

[22] S. Subramaniam, M. Azizoğlu, and A. Somani, "All-Optical Networks with Sparse Wavelength Conversion", *IEEE Transactions on Networking*, Vol. 4, August 1996, pp.544-557.

[23] S. Subramaniam, M. Azizoğlu, and Richard A. Barry, "The Benefits of Wavelength Conversion in WDM Networks with Non-Poisson Traffic", *IEEE Communications Letters*, Vol. 3, March 1999, pp. 81-83.

[24] Rajiv Ramaswami and Kumar Sivarajan, "Optimal Routing and Wavelength Assignment in All-Optical Networks", *IEEE INFOCOM '94 Proceedings*, June 1994, pp. 970-979.

[25] Allan Willner, "Mining the Optical Bandwidth for a Tb/s", *IEEE Spectrum*, April 1997, pp. 32-41.

Appendix

Computers were employed extensively to obtain the results in chapter 6. For each example network studied there were two main computations that had to be performed. First, the solution to the analytical model had to be calculated. This was accomplished using a C++ application, written by the author, on an Intel based PC. Second, a network simulation program, written in C by Glen Takahara, was used to determine blocking values for comparison purposes. Because of the length of time required for a single network simulation, several different computers were utilized. These were a mixture of Intel based machines and Sun work stations. Actually, neither pieces of source code are hardware specific, and thus should compile and run with very few modifications on any machine.

The analytical solution code, called *netwide.cc*, essentially follows the development of the analytical model. It inputs F, G, R, Λ, W, w, α, β and C, and outputs W' (assuming you want it to calculate a new converter placement W'), $Q = \{Q^{(L_r)}(r)\}$ and U_{nwb}. After getting all of the input information (including some initial converter placement W) it first determines if the user wants to use a heuristic to calculate W'. If so, it proceeds to calculate all node scores and deduce a new converter placement, which it stores by replacing the old. Next the code simply calculates the blocking on each of the user specified routes in order to produce the network-wide blocking value. Note that the user can specify which subset of the routes in R are to have their blocking calculated. This is very useful in the case of a symmetric network example where it would be redundant to calculate blocking for several routes with identical loading. There is a function that calculates an exact stationary distribution for any $K = 3$ links, and another that uses that function to implement the L-hop recursion. Blocking on a given route is determined and used to calculate the network-wide blocking value.

The code is not fully optimized, but considerable thought did go into the design in order to reduce memory requirements and computation time. The most significant design characteristic is the use of input files to speed up the calculation of an arbitrary 3 link stationary distribution. This is the focus for optimization because several 3 link stationary distributions need to be calculated. Recall from 3 that we need to deduce P in order to apply our Exact Iterative Algorithm. Although P does depend on F, the converter placement on the 3 links and the arrival rates to each of the 6 session types, up to a certain point we do not need to recalculate it's structure each time. We choose to store the non zero entries of P, up to the values of the 6 arrival rates, in a file that can be read in whenever necessary. There is an input file for each combination of F and the converter placement (there are only 4 possible converter placements with 3 links). Any time we need to calculate the steady-state distribution of 3 links with some converter placement, and for some value of F, we simply read in the appropriate file and quickly produce P using the current values of the session arrival rates. This results in a considerable savings in both memory and computation time. Storing only the non zero entries, and their structure of course, uses far less memory than would be required to store all of the $|S_{exact}| \times |S_{exact}|$ actual elements of P. Moreover, because the structure of P is contained in the input files, the code does not have perform the tedious calculations that determine all states that are reachable from a given state. This method seems to work very well. Once the input files are created once, using another piece of code called *createin.cc* that is called by *netwide.cc* when necessary (ie. an input file that does not already exist is needed), they are stored for future use. The code (*netwide.cc*) keeps track of which input files have already been created and other information in a file called *numstatesandps*. If there are no input files to be created, the network-wide blocking of a modest sized example can be calculated quickly.

The simulation code, called *networkgen.c*, is reasonably straight forward. It inputs the same information about the network, F, G, R, Λ, W, and eventually outputs the blocking on each route as well as the overall network-wide blocking. The simulation is event driven in nature. A random number generator is used to create a Poisson arrival stream for each route at the specified rate, and also the exponential holding time of the session. The random number generator is fed a fixed seed so that the results are reproducible if necessary. Sessions of a given type that are blocked on arrival, because of the current state of the network links, are recorded, as are the total number of arrivals of that type. This allows a simple calculation of the probability that sessions on a given route are blocked, and ultimately, the network-wide blocking is just the weighted average of these. Unfortunately, it takes a very long time for several blocking events to be recorded, especially at very low loads. The simulation must run long enough to reach a steady state (where the blocking on routes changes only slightly), so that we can trust the results as a true measure of the stationary blocking in the network. The code outputs the blocking on each route, the network-wide blocking and the maximum absolute difference in blocking among symmetrical sessions at user specified intervals. It is then up to the user to realize when the simulation has converged to the true stationary blocking by monitoring the output. That is, the user decides that the simulation may be stopped once the blocking values no longer change (in a certain number of decimal places that they determine is acceptable) from the output of one interval to the next. Of course the maximum absolute error within a class of sessions with the same blocking aids in the decision.

Vita

Name	Christopher S. D. Leith
Date of birth	May 1st, 1974
Place of birth	Kitchener Ontario, Canada
Education	Bachelor of Science (Honours Mathematics), 1997,
	Queen's University, Kingston Ontario, Canada
	Master's of Science (Mathematics), 1999,
	Queen's University, Kingston Ontario, Canada
Awards	NSERC IPS-1 Scholarship, 1997-1999
	Canada Scholarship, 1993-1997
	Bell Canada Scholarship, 1993-1997
	Queen's Honours Matriculation Scholarship, 1993-1995
	Annie-Lillie-Bentley Prize, 1994
	Governor General's Medal, 1993
Publications	1. Blocking For All-Optical Networks Based on Exact and Near Exact Blocking in a 3-Hop Path, *CISS '99 Proceedings*
	2. Exact, Approximate, and Recursive Blocking in All-Optical Networks, *GLOBECOM '99 Proceedings*