ABSTRACT

ILIA BALDINE. Dynamic Reconfiguration in WDM Networks. (Under the direction of Dr. George N. Rouskas and Dr. Yannis Viniotis.)

In this research we study the problems associated with dynamic reconfiguration of broadcast WDM networks. Adaptability to the changing traffic conditions is viewed as one of the key features of multiwavelength optical networks, and this is the first comprehensive in-depth study of this problem area. Our contribution consists of identifying the three main questions related to network reconfiguration: a) how to balance the load across multiple wavelengths; b) deciding when it is best to reconfigure the network and c) performing the actual reconfiguration in an efficient manner, that minimizes cell losses. We provide novel solutions to each of these problems. Our solutions consist of an algorithm we call GLPT, which balances the cell load across wavelengths, an optimal reconfiguration policy, derived from representing the problem as a Markovian Decision Process, and a class of retuning strategies that allow us to reconfigure the system. In addition, we perform a simulated comparison of static and dynamically reconfigurable networks in order to verify the validity of our approach. The simulation also provides us with valuable insights into the behavior of an adaptable optical network.

DYNAMIC RECONFIGURATION IN BROADCAST WDM NETWORKS

by

ILIA BALDINE

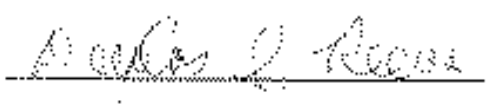
A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

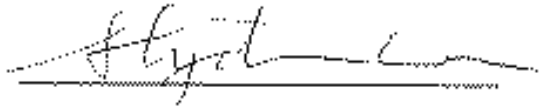
COMPUTER SCIENCE

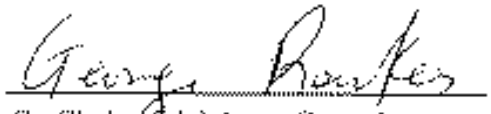
Raleigh

1998

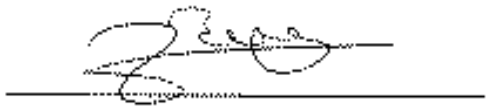
APPROVED BY:







Co-Chair of Advisory Committee



Co-Chair of Advisory Committee

BIOGRAPHY

Ilia Baldine was born August 7, 1972 in Dubna, Moscow Region, Russia. He received his B.S. in Computer Science in 1993 from the Illinois Institute of Technology and his M.S. in Computer Science in 1995 from North Carolina State University.

ACKNOWLEDGEMENTS

First of all I would like to thank my parents for their unconditional love and support, without which this work would have been impossible. I also would like to thank the members of my PhD committee for their comments and guidance, particularly Dr.George Rouskas. Finally, I would like to extend my deepest gratitude to all my teachers for making me who I am.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this work we propose a novel architecture for all-optical networks that allows them to reconfigure in response to changes in the traffic pattern. Having communications networks dynamically adjust themselves to the predominant traffic conditions has been an outstanding problem for a number of years. Such capability offers the potential for a much more efficient use of network resources and, as a result, a higher overall network throughput. Given that currently the new bandwidth is consumed as fast as it is offered, novel architectures capable of using it more efficiently promise to alleviate the congestion that the present-day networks are experiencing.

In this work we discuss the problems associated with the dynamic reconfiguration of single-hop WDM networks in response to changes in the traffic pattern. These problems include

1. Deciding when the change in the traffic pattern warrants a reconfiguration or what we call a *Reconfiguration Policy*.

2. Balancing the packet load across wavelengths by using reconfigurable elements, such as retunable optical receivers.

3. Performing the actual reconfiguration in the most efficient manner possible with the use of a *Retuning Strategy*.

The following sections will briefly summarize our work on each of the problems.

1.1 Reconfiguration Policy

Our solution to this problem is based on Markovian Decision Processes and the associated optimal policies. We first present the problem as an MDP and then use the existing algorithm for calculating such optimal policies to come up with a *Reconfiguration Policy* which can be used in a real network.

1.2 Load Balancing

At the heart of this problem lies the need to create assignments of nodes to channels in a way that the packet load is spread evenly across the channels. We also wish to create such an assignment based on the previous assignment in a way that the the reconfiguration phase (problem area 3 above) is simplified.

To this end we present an algorithm called Generalized LPT (or GLPT), which allows us to create assignments of nodes to optical channels based on both the current traffic information and the preceding assignment of nodes to channels. The algorithm also allows us to gauge the goal of balancing the load across channels as perfectly as possible against the complexity of the following reconfiguration phase.

1.3 Retuning Strategy

We assume that the network has made the decision to reconfigure according to the *Reconfiguration Policy* and have calculated the new assignment of nodes to channels. What remains is to perform the actual reconfiguration of the network in a way that the packet losses during that phase are minimized. As a solution we present a greedy heuristic.

1.4 Thesis Organization

Chapter 2 will describe the history of optical networks and introduce the subject of all-optical WDM networks. Chapter 3 will describe some of the previous work done in the area of dynamic reconfiguration of optical networks. Chapter 4 will present a system model and introduce the novel architecture we propose. Chapters 5,6 and 7 will be dedicated to the three problem areas described above. They will each introduce the problem and present

our solution to it. Finally Chapter 8 will present our conclusions and some of the directions this work can be taken in the future.

Chapter 2

Optical WDM Networks

2.1 First-Generation Optical Networks

The need for higher and higher capacities in voice and data networks has been the single driving force behind network design since their early emergence. What started as a need to transmit analog voice signal over long distances became transformed into the need to transmit huge amounts of digital information that includes voice and video in real-time in addition to many kinds of other data. The merging of telephone and data networks in the early 90's spurred the need not only for protocols that could successfully combine data of different types with different quality of service requirements (e.g. ATM) but also for the networks that could carry these increasingly large amounts of data. Traditional copper-wire telephone networks gave way to optical fiber, which due to the width of the optical spectrum can carry thousand times more data with low error-rates over long distances. Phone companies were the first to utilize this property of the optical fiber, and up until recently there was no need to deliver large bandwidth directly to the desktop, so the use of fiber was limited to the long-distance telephone networks. However the emergence of inexpensive fast and powerful PCs and their potential applications in everyday life created a need to move large amounts of data between them, and this ever-growing need can no longer be met by the traditional copper-based LAN technologies. In response to that computer network designers started looking at the possibility of using optical fiber as the medium for delivering these large amounts of information that modern-day PCs are able to process.

The first optical architectures were adaptations of well-known copper-based protocols. For instance FDDI (Fiber Distributed Data Interface) was an extension to the optical

domain of a commonly used token-ring protocol Later, as the need for high-throughput protocols arose, fiber-specific protocols started to appear, such as SONET/SDH. They were both adopted in order to allow the multiplexing of multiple data and/or voice streams over a single fiber at speeds ranging from 51Mb/s (STS-1) to 9.953Gb/s (STS-192). The addition of direct mapping of ATM (Asynchronous Transfer Mode) to Sonet/SDH made this technology especially important for data networking, as ATM is seen by many as the dominant Layer 2 and 3 (Link and Network Layers) protocol for the near future.

2.2 WDM Networks

Since the time they first appeared it was clear that optical networks were the solution to the bandwidth crunch problem that the telecommunications industry was headed for. The low-attenuation regions of the optical fiber (the regions of light wavelengths, where the optical signal experiences the least amount of attenuation, while traveling through the fiber) contain a total of 50THz of bandwidth. However the initial applications of this technology did not take full advantage of the properties of the optical signals and the colossal bandwidth. First-generation optical networks transmitted and received data on a single wavelength, which although providing a significant amount of bandwidth compared to the copper-based networks, in reality used only a fraction of the bandwidth that the fiber offered. Due to the properties of the propagation of optical signals in the fiber it is possible, in fact, to transmit several signals at the same time on different wavelengths with low interference. At the present state of optical technology it is possible to have about 30 separate wavelength channels in the fiber and transmit data through them at rates of several gigabits per second, which is equivalent to the bandwidth extracted out of the entire fiber by the state-of-the-art first-generation systems. In other words, using the same fiber it is now possible to transmit 30-times more data. This technique of subdividing the optical spectrum into channels is similar to the technique of Frequency-Division Multiplexing used in wireless and copper-based signals, and because of that it is called Wavelength-Division Multiplexing or WDM.

WDM is achieved by using lasers and optical filters that can transmit and receive optical signals only on a particular wavelength or channel. At the present state of technology these channels have to be spaced 1 to 2nm apart to avoid crosstalk. Early examples of the use of this technology can already be seen in several products currently on the market. One

of them is an IBM 9729 Optical Wavelength Division Multiplexer, which provides up to 10 channels on each individual fiber up to 155Mb/s each. The channels are spaced 1nm apart in the 1540-1559nm wavelength region. This product however is quite simple compared to the systems currently under development at other companies. It provides a point-to-point connection for two locations up to 30 miles apart. It's purpose is to help companies that at present use optical fiber technology multiply the bandwidth in their point-to-point connections without significant investments in the infrastructure (i.e. get more bandwidth out of the existing fiber).

In addition to increasing the amount of the utilized bandwidth in the fiber, WDM also opens a way to the use of fiber-optics in Local Area Networks. It does so by breaking up the huge available bandwidth into smaller pieces that can be "digested" by the desktop computers, since no PC can at present hope to utilize the entire bandwidth offered by the optical fiber. Several competing architectures have emerged that attempt to solve some of the logistical problems associated with allowing multiple network nodes communicate over the same fiber.

Most of the problems have to do with the need for the nodes to be able to transmit and receive on several of the available channels, in order to communicate with one another freely. If the nodes can talk and listen on a single channel only, than the network will essentially be broken up into several subnetworks, one per channel, unable to communicate with one another. This problem can be solved in a multitude of ways. Early solutions proposed having arrays of lasers or optical filters on each of the nodes, with each laser or filter tuned to a different channel. This way a node can communicate with other nodes by choosing one of the lasers or filters in the array, and thus choosing a channel for transmission or reception of data. This solution however has a significant drawback of being expensive and non-scalable. It is foreseen that the number of channels available on the fiber will increase as the technology matures, and this means that these arrays of lasers or filters would have to grow as the number of channels grows.

As a result of the extensive research into the physics of lasers and optical filters, it is possible today to have these devices "tunable". That means a single device can utilize several optical channels by modifying its physical characteristics. This possibility gave rise to several competing architectures for WDM networks that are at present under study in the industry and in academia.

2.3 Wavelength-Routed Networks

One such network architecture is intended to be used in long-haul WANs and is called "wavelength-routed" [19]. The main idea behind the wavelength-routed networks is that of a "light-path" which carries data from one node to another. A light-path is realized by allocating bandwidth on links connecting the origin and the destination nodes *on a single wavelength.* A network may consist of many such light-paths. The main difference from the traditional switching networks is that the data does not leave the optical domain until it arrives to the destination node. In other words, if in traditional networks employing optical fibers, the signal at every node has to be transformed from the optical form to the electrical form to be processed and routed, so that it can be again transformed back into the optical form and sent on the outgoing fiber, in the wavelength-routed networks the signal remains in the optical form until it reaches its destination. This is achieved by routing the optical signal based on its wavelength, hence the name "wavelength-routed". This removes a significant bottleneck which the electro-optical interfaces create in the present-day optical networks, where the top speed of the the network is limited to the speed at which the electro-optical interface can process the signals.

The other two competing architectures are more suited to be used in LANs and MANs and will be described in more detail in the following section.

2.4 Broadcast-and-Select WDM Networks

In these types of networks a node is equipped with one fixed and one tunable optical device device (a transmitter or a receiver). Each node is associated with a particular channel, to which its fixed device is tuned to. The first of these architectures assumes that a tunable device (a laser or a filter) on a node can only tune to some of the available channels but not all of them. This assumption is made based on the research that shows that as the spectrum available to the device grows (it can tune to more channels) so does its cost. This stipulation means that not every node is capable of communicating with every other node, since some of them may "listen" or be capable of tuning to wavelengths unavailable to others. The other architecture takes a more optimistic approach and assumes that a tunable device can access all of the available channels, which means that any node can in principle reach any other node. The networks of the first type were given the name "Multi-Hop",

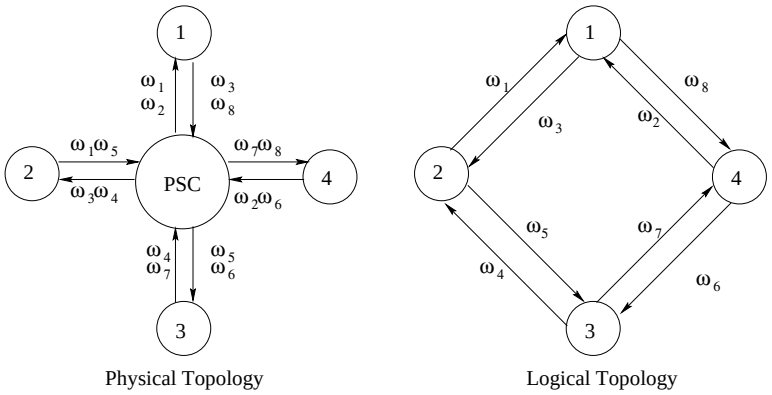


Figure 2.1: Physical vs. Logical Topology in a Multi-Hop Network

since in order for node A to reach node B , with which it has no common wavelength, it might have to communicate through node C, which has available to it wavelengths common to both A and B. The second type were given the name "Single-Hop" since any node can communicate with any other node.

Within these two broad types of network architectures two different subtypes of WDM networks are defined, depending on whether the lasers or the optical filters are tunable. Networks in which the lasers are tunable and the filters are fixed are called TT-FR or "Tunable Transmitters - Fixed Receivers". Their counterpart is called FT-TR or "Fixed Transmitters- Tunable Receivers". Regardless of the type, both architectures have a star physical topology with multiple fibers connected to a device called "Passive Star Coupler" (PSC) [24] which works as a multiplexer for every incoming link by splitting the optical signal into all of the outgoing links and, in essence, broadcasting any input to all the outputs, hence the name Boradcast-and-Select. As with wavelength-routed networks the signal does not leave the optical domain until it reaches its destination, since the PSC is an all-optical device and does not do any signal processing or routing of its own.

The chief difference between the architectures lies in the logical topologies they impose on the network. The topology of a single-hop network can be represented by a complete graph, since any node can communicate with any other node. On the other hand, the topology of a multi-hop network depends entirely on the assignment of the devices on the fixed side (transmitters or receivers) to particular wavelengths (see Figure 2.1).

For Single-Hop networks the particular assignment of the fixed devices to wavelengths is no less important than it is for the Multi-Hop networks, since each node needs

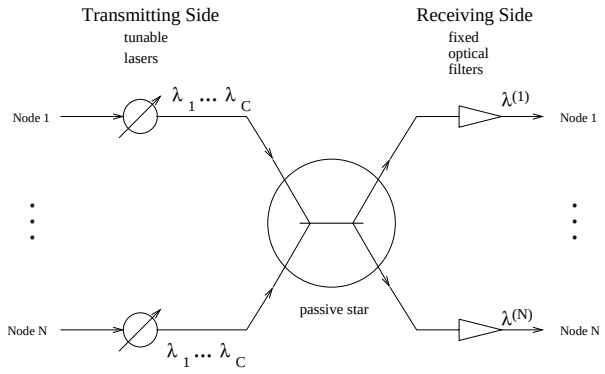


Figure 2.2: A Single-Hop TT-FR network.

to know which channel the other nodes can transmit or listen to, in order to communicate with them (see Figure 2.2)

In this discussion we have not at all touched on the subject of the medium-access (MAC) procedures in single- and multi-hop optical networks, nor did we mention the time scale at which the retuning of the optical devices takes place. These two questions lie at the heart of the research into the WDM networks and will be given special attention here.

When talking about "tunable" devices in the context of Single- and Multi-Hop optical networks, what is usually meant is that the device is capable of switching from one wavelength to another *within the time comparable to the transmission time of a single packet*. This stipulation is very important for the MAC procedures, since if the "tunable" device takes a very long time to retune, compared to a single packet transmission time, then we cannot hope for high link utilization in such a network, since the devices will spend a significant proportion of time retuning, without transmitting or receiving packets. For this reason in the following chapters we will refer to these devices as "fast-tunable" to emphasize the fact that they are capable of switching from one wavelength to another within a short time compared to packet transmission time. Such devices exist today, however their cost rises dramatically as one tries to shorten the tuning time further and further.

Most MAC protocols proposed for use in WDM networks make an assumption that "fast-tunable" devices are in use. Their main goal is to utilize the optical channels as much as possible by masking the time spent on tuning through scheduling techniques. Despite this common goal, these protocols vary drastically depending on the type of the architecture in question. For instance, in multi-hop optical networks the medium-access protocol must be

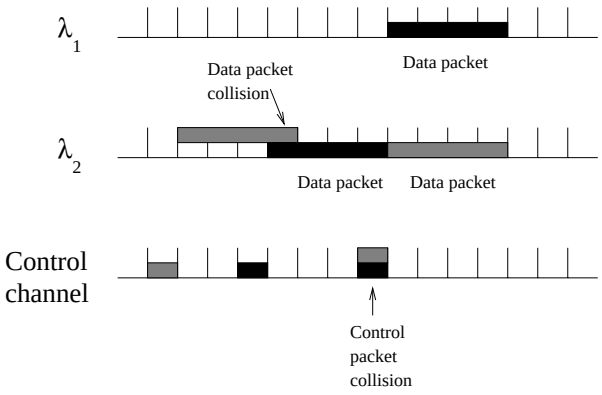


Figure 2.3: Collisions in SA/SA operation.

able to communicate to the nodes the topology of the network to facilitate data transmission, while in single-hop optical networks the topology is not in question. Also, depending on whether the architecture is TT-FR or FT-TR, the method of disseminating the necessary information between nodes will be different - it can be done either in broadcast mode or on a specified control channel. Since this work is only concerned with single-hop WDM networks, the following section will more closely examine several of the existing MAC protocols for this architecture.

2.5 MAC Protocols in Single-Hop WDM Networks

Early MAC protocols for WDM networks were created as extensions of already existing MAC protocols for broadcast media, i.e. Aloha, Slotted Aloha. Many of them made assumptions that did not allow to make use of the properties that make the WDM architectures distinct from the existing network architectures. For instance in many early protocols the number of nodes was assumed to be equal to the number of available channels. Given their significant bandwidth, each channel is, in fact, capable of accommodating multiple nodes, so granting a single node exclusive use of a channel is wasteful from the point of view of bandwidth utilization.

Among these protocols were SA/SA and DT-WDMA. The former was a variant of a well-known Slotted Aloha protocol, up to this point used in satellite communications. For a network with N nodes it needs $N + 1$ channels - 1 data channel for each of the nodes with an additional control channel. The discipline used on both the data and the control

channels is Slotted Aloha, hence the name Slotted Aloha/Slotted Aloha or SA/SA. The basic operation of the protocol can be described as follows - whenever a node has a packet to send, it sends a short control packet on the control channel and the data packet is sent on the node's data channel immediately following the control packet. The delivery of either the control or the data packet is not guaranteed - if another node starts transmitting in the same slot, then both packets are corrupted and lost and the nodes have to retransmit them (see Figure 2.3). If the control packet is not corrupted in transit, it is received by all the nodes, including the destination node, which then knows to expect a data packet on its preassigned channel. This scheme is intended for use with TT-FR networks.

The second MAC protocol - DT-WDMA was specifically developed for the WDM networks, but it bears close resemblance to the SA/SA. It also uses $N + 1$ channels - N data and a control channel, however, unlike SA/SA the data slots cannot overlap, and the size of the data packet is N times the size of the control packet. Thus N control packets are synchronized to a single data packet and each node has a preassigned slot among these N available control slots. Whenever a node has a packet to send, it waits for its slot on the control channel and transmits a control packet in it, with the data packet following immediately in the next data slot. Each node listens on the control channel and upon receiving a control packet indicating it is about to receive a packet, it can retune its receiver to the wavelength indicated by the slot, in which the control packet was received, and receive the data packet. In this scheme the collision of both data and control packets is impossible, since control packets can only be transmitted by nodes in their preassigned slots, and the data packets are transmitted on different channels - the network architecture used is presumed to be FT-TR (see Figure 2.4). Packet loss occurs when two or more nodes send their packets to the same destination node in the same data slot - since a node can only listen to one of the data channels within a single data slot, all other data packets are lost.

Both these protocols suffer from low throughput (especially SA/SA) and high packet losses under high loads (see [19]). Both these qualities are due to the *tell-and-go* property they share - they attempt to transmit a data packet as soon as they have one. Thus they give up on any possibility of scheduling packets for transmission themselves, which might result in fewer collisions, and consequently in higher throughput. They also make an assumption that the number of nodes and available data channels is the same, which severely limits the number of nodes a network can have using current technologies

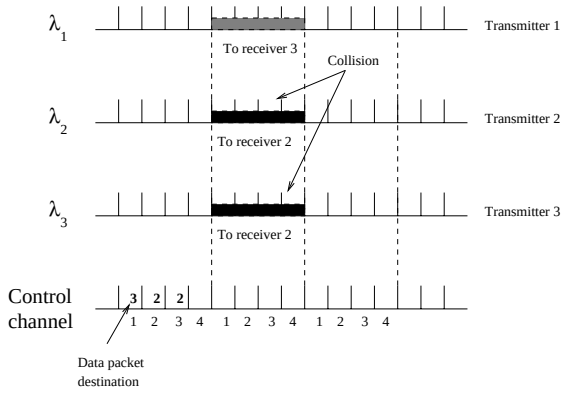


Figure 2.4: Collisions in DT-WDMA operation.

(as mentioned earlier, presently the number of channels is limited to 10-20).

Other protocols attempt to circumvent these difficulties by buffering the arriving packets, until a significant number of them accumulates and then collectively generating collision-free transmission schedules. This group of protocols is called "reservation protocols" because through the use of schedules they reserve bandwidth for the packets they have ready for transmission ahead of time. Such protocols use the notion of a "Traffic" or "Backlog" matrix T. This matrix is dimensioned NxN, with N being the number of nodes, whose element t_{ij} denotes the number of packets node i has for node j ready for transmission. This matrix is used for calculating the transmission schedules. Packets are usually buffered in the queues on a per-wavelength basis. Some of the protocols consider only the head-of-line packets in the queues (FatMAC [25]), others consider multiple packets for transmission per node to improve the overall utilization (DQMW [15]). Most however share a common deficiency - they schedule each packet independently and therefore incur significant tuning delays, since in the worst case a transmitter might need to be retuned for each of the outgoing packets (in a TT-FR architecture).

One of the recently proposed schemes overcomes all of these difficulties by rearranging the outgoing packets and grouping them by the outgoing wavelength. It also takes into account and masks some of the necessary tuning delays. This protocol is called HiPeR-l and it will be discussed in detail in the following section.

2.6 Description of HiPeR-l

HiPeR-l was proposed in [26] and is based on the scheduling algorithm proposed by the same authors earlier ([20]). The protocol has few restrictions - it allows for multiple nodes per wavelength and there is no need for a special control channel. It achieves the highest utilization when the tuning time of the transmitters or receivers (depending on whether it is an FT-TR or a TT-FR configuration) is close to the packet transmission time, although the latter is not necessary for the operation of the protocol. Packets are all assumed to be of equal length and not dissimilar to ATM cells in length and structure. The protocol works as follows:

- Control packets are sent periodically on each of the wavelengths in order to update the traffic matrix T maintained by each of the nodes
- Each node posesses a copy of the same scheduling algorithm. Using this algorithm a node calculates a transmission schedule, and since all the nodes use identical algorithms on the identical traffic matrices, they all use same transmission schedules and thus avoid collisions

HiPeR-l's distinctive features are the lack of a control channel and its distributive nature which allow for easy scalability. It can be used in both FT-TR and TT-FR configurations with minor modifications, from now on, however, the TT-FR (Tunable Transmitter-Fixed Receiver) configuration will be assumed. It can also successfully hide the tuning latencies of the transmitters or the receivers by overlapping the tuning time at one node with the transmission of a number of packets by other nodes. The underlying scheduling algorithm can construct schedules of near optimal length for a given traffic (or backlog) matrix T, so a high degree of throughput can be achieved by transmitting a number of packets back-to-back on the same channel before retuning to the next. Also, the amount of data sent in the control packets is small - it amounts to simply the number of packets each node has waiting in each of its queues associated with the available channels. This keeps the operational overhead of the protocol low.

Most of the work presented here is based on the assumption that the Single-Hop network architecture we are studying employs HiPeR-l as its MAC protocol One of the notions derived from the use of HiPeR-l, which will be used repeatedly here is that of a "frame" - a collection of slots accommodating a single schedule. A typical frame length

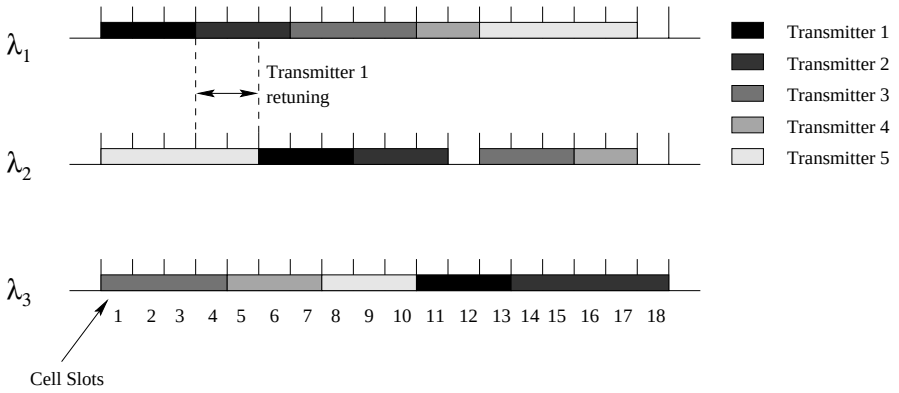


Figure 2.5: An example of a HiPeR-l frame for 5 nodes and 3 wavelengths.

under moderate loads can be 400-500 cell slots. In Figure 2.5 we see a single frame of length 18, created for 5 nodes and 3 wavelengths. The transmitter tuning latency is equivalent to 2 cell slots. The notion of a frame is important because every time a frame is transmitted, the traffic matrix T is renewed, so the new schedule for the next frame needs to be calculated.

The performance analysis of HiPeR-l has resulted in several important observations regarding the utilization of an optical network which employs HiPeR-l or a similar *gated* protocol. These findings and their consequences for this work will be presented in the following section.

2.7 Load Balancing in Single-Hop WDM Networks

We have emphasized from the start that increasing the utilization of an optical network is the goal of this work. At this point we will provide the main justification for our research.

In the performance analysis of HiPeR-l two parameters of importance were introduced:

- *Degree of load balancing* ϵ_b, which characterizes the upper bound on the number of cells carried by a single channel, compared to the lower bound, when the cells are distributed evenly across wavelengths.
- *Scheduling guarantee* ϵ_s, which characterizes the actual constructed schedule length compared to the lower bound.

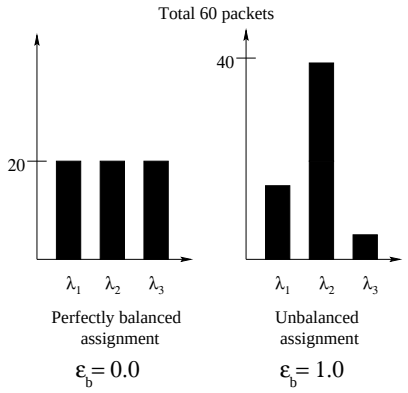


Figure 2.6: Examples of a balanced and an unbalanced assignment of receivers to wavelengths.

Based on these two parameters the following stability condition for HiPeR-l was arrived at (see [26]):

$$\gamma < \frac{C}{(1+\epsilon_b)(1+\epsilon_s)} \tag{2.1}$$

where γ is the total cell arrival rate to the network and C is the number of available channels.

What this expression shows, is that minimizing the ϵ_b and the ϵ_s quantities will have the effect of raising the arrival rate in the network, thus increasing the throughput. And while ϵ_s depends purely on the scheduling algorithm used, and in the future will be considered constant, ϵ_b depends on the number of cells accumulated for transmission on each channel, and is affected by the particular assignment of the fixed receivers to the channels. For instance, if the receivers are assigned to the wavelengths such that the load is distributed evenly between wavelengths, we will see something similar to the diagram on the left of Figure 2.6. If, however, several heavily-used servers are assigned to wavelength 2, while the other two wavelengths remain underutilized, we will see something similar to the diagram on the right of Figure 2.6.

Clearly the degree of load balancing depends not only on the assignment of the receivers to the wavelengths, but also on the number of packets that arrive for each of the destinations in each node during a frame. This, in turn, will be governed by the predominant traffic pattern currently in effect. If we could find a way to lower the degree of load balancing for the different traffic patterns by dynamically reconfiguring the network, this would result in the higher total cell arrival rate, and, in turn, in a higher throughput. To the best of our knowledge this work is the first to consider this problem in Single-Hop optical networks.

It has been, however, considered for the Multi-Hop networks and some of the proposed solutions will be described in the following chapter.

Chapter 3

Previous Work

3.1 Dynamic Reconfiguration in Multi-Hop Networks

As was mentioned in the beginning of the previous chapter, Multi-Hop networks differ from Single-Hop in that the assignment of the fixed receivers or transmitters to the particular channels dictates the logical topology of the network. If this assignment is allowed to be changed dynamically, this property can be exploited in order to improve network performance and reliability. Such dynamic networks hold a great promise over their static counterparts in operating at higher utilization and having self-healing capabilities to help cope with potential failures. This feature allows us to adapt the network topology to the different optimization goals we might be faced with. Two of the most popular goals are:

- Throughput Optimization - minimizing the maximum flow on each link, a problem well-known in the world of circuit-switched networks.
- Delay Optimization - minimizing the maximum number of hops a packet has to traverse to reach its destination.

Once the optimization goal is chosen, two following problems must be addressed:

1. Determine the best new topology for the new traffic conditions given the optimization goal (*Traffic Optimization* see [14]).
2. Find a way to reconfigure the network from its original topology to the new one (*Reconfiguration Management* see [14]).

Furthermore, when determining the new best topologies we can be limited to a set of predetermined fixed topologies, or be able to use any arbitrary regular topology.

The following two sections will describe in more detail approaches used in solving the two problems.

3.2 Optimizing the Topology

Both formulations of the problem depend on the state of the traffic in the network which is usually described as a NxN matrix

$$\Lambda^T = \{\lambda_{st}^T\}_{1 \leq s,t \leq N} \tag{3.1}$$

where λ_{st}^T is a measure of the amount of traffic node s has for node t. Another parameter of importance is the matrix representing propagation delays in the network

$$\Delta^T = \{\delta_{ij}^T\}_{1 \leq i,j \leq N} \tag{3.2}$$

As mentioned above, the optimization can be done both over an arbitrary ([12], [6]) and a fixed set of topologies ([21], [22]). Examples of fixed topologies could be the Perfect Shuffle, de Bruijn Graphs, uni- and bi-directional rings, Manhattan Street Network and hypercube. We will examine the two approaches separately below.

3.2.1 Optimization over Arbitrary Topologies

In this formulation it is important to describe the amount of traffic, present on each link due to a particular source-destination pair in order to come up with the best topology. This is done by defining a number of flow variables f_{ij}^{st} each describing the traffic on a link connecting node i to node j due to source-destination pair (s,t). To describe the topology a set of connectivity variables z_{ij} is introduced. The values of z_{ij} are restricted to $\{0,1\}$ and their values represent the actual connections in the network - 1 for when a link connects node i to node j and 0 when there is no direct link.

These definitions allow us to represent the problem of optimizing the topology for a particular goal as an integer programming problem. For instance, throughput maximization can be done by minimizing the largest flow in the network (the so called min-max problem) and can be formally presented as

$$F = max_{(i,j)}\{\sum_{s,t} f_{ij}^{st}\} \tag{3.3}$$

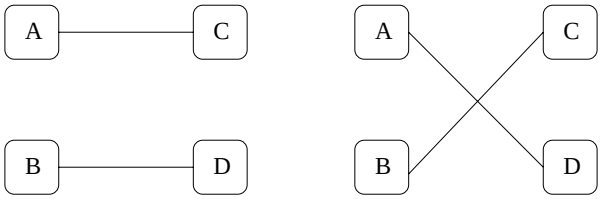


Figure 3.1: An example of a Branch-Exchange Operation

$$\sum_{s,t} f_{ij}^{st} \leq z_{ij} C \tag{3.4}$$

where C is the link capacity. The resulting integer programming problem can be solved by a variety of existing techniques like the simplex method, the Tabu search [7], or simulated annealing. Other approaches include heuristic algorithms that attempt to improve the existing topology through localized changes called *branch-exchange operations* (see Figure 3.1).

3.2.2 Optimization over Fixed Topologies

Since in this approach the topology is already known, the problem then becomes that of assigning particular nodes to locations in the fixed topology. To that end the problem formulation now includes assignment variables X_{si}, whose values are limited to $\{0, 1\}$. They represent the assignment of nodes to locations - 1 if node is s is assigned to location i and 0 otherwise. Also, for delay optimizations, yet another set of variables η_{ij} can be added to represent the shortest propagation delay between two nodes for a given topology.

As with arbitrary topologies, a variety of techniques exist that tackle this problem. Some of them are based on integer programming techniques, while others attempt to reduce the problem to other problems such as *minimum cut linear arrangement* and *optimal linear arrangement problem* which happen to be NP-complete [23].

3.3 Reconfiguration Management

Once the new topology, that is optimized to some set of parameters is determined, the problem becomes that of transitioning the network from the initial topology to the new one. Here the extremes are taking the network off-line and performing the reconfiguration in one step, but sustaining high packet losses. Alternatively, we can perform the reconfigura-

tion in small steps, thus minimizing the packet losses, but possibly degrading the long-term performance due to the duration of the reconfiguration period.

Authors of [13] consider the problem of constructing a sequence of previously mentioned *branch-exchange operations* that transforms the network from the initial topology into the new one. They propose polynomial-time algorithms that find minimal sequences of such operations, so as to keep the reconfiguration phase short.

This concludes the description of the previous work in the area of dynamic reconfiguration of WDM networks. In the next chapter we will present our reconfigurable architecture for single-hop networks, as well as the system model.

Chapter 4

System Model

4.1 Reconfigurable Single-Hop WDM Networks

In the previous chapters we have defined the need for the dynamic reconfiguration capability in WDM networks. In short it means higher throughput and more optimal use of network resources. In this section we will define the architecture that posesses this capability.

Traditional Single-Hop WDM networks, as mentioned before, have tunable device only on one side - either the receivers or the transmitters (FT-TR vs. TT-FR). In the case of a TT-FR (Tunable Transmitter- Fixed Receiver) configuration, in order to communicate with another node, a node must know the destination's receiver wavelength, so it can tune its transmitter to it and send the cells. This architecture is incapable of reconfiguration, since its only dynamic aspect - the tunable transmitter, is used to provide for network connectivity with a scheduling protocol such as HiPeR-l. What we require, is an architecture where both the transmitter and the receiver are tunable, and an appropriate MAC protocol capable of communicating changes in the configuration to the nodes in the network. Earlier we spoke of fast-tunable devices, as being traditionally used in WDM networks, and how expensive they were. For this reason an architecture where each node is equipped with a fast-tunable receiver and a fast-tunable transmitter will be prohibitively expensive, and also unnecessary. The need for reconfiguration will not be as frequent as the need to retune from one wavelength to another while transmitting cells - it will happen on a significantly longer time-scale, and therefore a fast-tunable device is not needed. For this reason we propose a novel architecture, in which the transmitters are fast- or rapidly-tunable, while

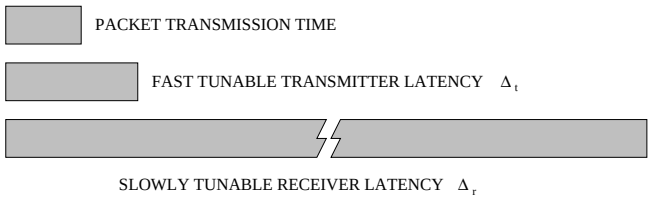


Figure 4.1: Cell transmission time vs. transmitter and receiver tuning latencies

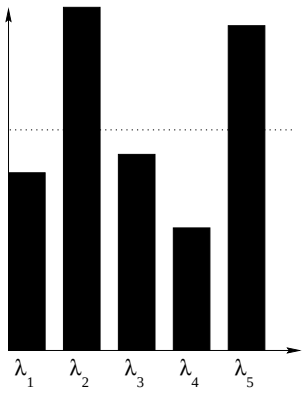


Figure 4.2: Unbalanced load

the receivers are *slowly-tunable* or as we call it $RTT - STR$. Figure 4.1 shows the difference in times scales of a single cell transmission time, the time it takes a transmitter to retune or its *tuning latency* and the time it takes a receiver to retune.

This architecture retains the characteristics that make it suitable for use with an efficient scheduling protocol such as HiPeR-l, while allowing for dynamic reconfiguration at a moderate cost. Let's look at how its reconfiguration capability can be put to our advantage. Presume, that the traffic conditions have changed so drastically that the cell load across the wavelengths is very unbalanced as in Figure 4.2. If now, based on the changed traffic conditions we come up with a new assignment of nodes to channels, which allows for a better balanced cell load as in Figure 4.3, and by using the retuning capability of the receivers we implement that new assignment, the network should function more efficiently according to expression 2.1.

While in theory this looks very promising, there are several problems that need to be faced:

1. We need to decide when the reconfiguration is advantageous from the point of view

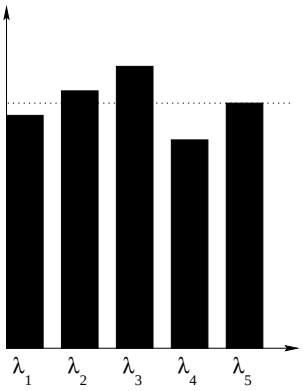


Figure 4.3: Balanced Load

of utilization and cell losses. While the network is reconfiguring, some cell losses are inevitable as receivers become unavailable while they retune.

2. We need to be able to determine what the new balanced assignment of nodes to wavelengths is.

3. Having the initial and the new assignment of nodes to wavelengths we need a way to efficiently reconfigure the network from one to the other with minimal cell losses.

These are the problems (and the solutions to them) that we will be presenting in the following chapters. However before we proceed, we must introduce several concepts that define our system model, which will be used throughout this work.

4.2 Network Model and Operation

We consider a packet-switched single-hop lightwave network with N nodes, and one transmitter-receiver pair per node. The nodes are physically connected to a passive broadcast optical medium that supports $C < N$ wavelengths, $\lambda_1, \cdots, \lambda_C$. Both the transmitter and the receiver at each node are tunable over the entire range of available wavelengths. However, the transmitters are *rapidly tunable*, while the receivers are *slowly tunable* (RTT-STR).

Let Δ_t (Δ_r) denote the normalized tuning latency of transmitters (receivers), expressed in units of cell transmission time. In the RTT-STR system under consideration, we have that $\Delta_r \gg \Delta_t \geq 1$, where Δ_t is a small integer, while Δ_r takes values that may be

significantly greater than Δ_t. The main motivation for employing slowly tunable receivers vs. fast tunable ones is the significant savings in cost that can be realized.

We distinguish two levels of network operation, differing mainly in the time scales at which they take place. At the *cell scheduling* level, connectivity among the network nodes is provided by reservation protocol such as HiPeR-ℓ [26] that requires tunability only at the transmitting end. The protocol schedules cells for transmission by employing scheduling algorithms that can effectively mask the tuning latency of tunable transmitters [16, 20, 2, 4, 18]. Since the receiver latency Δ_r is significantly long and cannot be overlapped with cell transmissions, at this level of operation the receivers are considered to be fixed tuned to a particular wavelength. Let $\lambda(j) \in \{\lambda_1, \cdots, \lambda_C\}$ be the wavelength currently assigned to receiver j. We define an assignment of wavelengths to receivers as a partition $\mathcal{R} = \{R_c,\ c = 1, \cdots, C\}$ of the set $\mathcal{N} = \{1, \cdots, N\}$ of nodes, such that R_c is the subset of nodes currently receiving on wavelength λ_c:

$$R_c \quad = \quad \{j \ \mid \ \lambda(j) = \lambda_c\} \qquad c = 1, \cdots, C \tag{4.1}$$

In the future we will be referring to receiver wavelength assignment as WLA.

The ability of receivers to tune, albeit slowly, is invoked only at the *resource allocation* level; in this work, the shared resource of interest is bandwidth. We note that a partition $\mathcal{R} = \{R_c\}$ in (4.1) implies an allocation of the available bandwidth to the various receivers. The availability of tunable receivers allows this allocation to be optimized to prevailing traffic conditions. As the traffic varies, a new assignment of receive wavelengths may be sought that satisfies some optimality criteria. We will use the term "reconfiguration" to refer to the reallocation of bandwidth to receivers. Since this variation in traffic will more likely take place over larger scales in time, reconfiguration is expected to be a relatively infrequent event, and each assignment of receive wavelengths will be long lived relative to the scheduling of cell transmissions by the media access protocol. Consequently, receivers with a tuning time Δ_r significantly larger than the cell transmission time, will be acceptable at the resource allocation level as long as Δ_r is small compared to the mean time between successive reconfiguration events.

Intuitively, receive wavelengths should be assigned so that the traffic load be balanced across the C channels. In Chapter 2 we mentioned parameter ϵ_b - the degree of load balancing which was introduced as part of the stability condition for HiPeR-l 2.1. We formally define it such that no channel carries more than $\frac{(1+\epsilon_b)}{C}$ times the total traffic offered to

the network. In other words, ϵ_b is a measure of the *degree of load balancing* of the network; under perfect load balancing, $\epsilon_b = 0$.

We represent the bandwidth requirements of source-destination pairs by a traffic demand matrix $\mathbf{T}= [t_{ij}]$. Quantity t_{ij} could be a measure of the average traffic originating at node i and terminating at node j, or it could be the effective bandwidth [17] of the traffic from i to j. Given matrix $\mathbf{T}$, we can compute the total bandwidth requirement b_j of receiver j as the sum of the elements of the j-th column of $\mathbf{T}$:

$$b_j \quad = \quad \sum_{i=1}^{N} t_{ij} \qquad j = 1, \cdots, N \tag{4.2}$$

As mentioned in the beginning of this chapter, we distinguish three separate problems that need to be addressed in order for a network to go through the reconfiguration phase efficiently:

1. Determining the balanced WLA for the new traffic conditions (Chapter 5).

2. Determining when it is optimal to reconfigure the network by gauging the packet losses associated with the reconfiguration against the improvement in network efficiency achieved through a new WLA (Chapter 6).

3. Finding a way to optimally reconfigure the network from one WLA to another with minimal packet losses (Chapter 7).

Even though in the previous section the problem of finding the balanced WLA was stated as number 2, we actually need to find a solution for it *before* we can address the other two problems, since they both depend on the newly calculated balanced WLA.

The following chapters will discuss our proposed solutions to each of these problems.

Chapter 5

Load Balancing in Single-Hop WDM Networks

Based on our observations regarding load balancing, our objective is to assign the receivers to the available channels such that the total bandwidth used in each channel is approximately the same among different channels. This problem is equivalent to the multiprocessor scheduling problem [8], where given a set of tasks with *a priori* known processing times and a number of processing units, the objective is to allocate the tasks to the processors such that the overall finish time is minimized. (This implies that the total processing time of the various processors differs as little as possible.) In our case the channels take the place of the processors, the receivers replace the tasks and the bandwidth requirements b_j replace the processing times.

The multiprocessor scheduling problem is $\mathcal{NP}$-complete [9]. Two approximation algorithms for this problem are *MULTIFIT* [5], with an absolute performance ratio of 1.22, and *LPT* [10], with an absolute performance ratio of 1.33. Either of these two algorithms may be used to obtain an assignment of receive wavelengths based on the receiver bandwidth requirements b_j, $j = 1, \cdots, N$, such that traffic is spread across the various channels as evenly as possible. We now proceed to discuss what happens when, due to changes in the traffic pattern, the current wavelength assignment becomes suboptimal.

5.1 The Transition Phase

Let $\mathcal{R}$ be an assignment of receive wavelengths based on traffic matrix $\mathbf{T}$ and the corresponding bandwidth requirements $\{b_j\}$ in (4.2). As traffic varies over time, the elements of matrix $\mathbf{T}$, as well as the column sums $\{b_j\}$, will change. Let $\mathbf{T}'$ be a new traffic matrix, and $\{b'_j\}$ be the new receiver bandwidth requirements. If, due to these traffic changes, assignment $\mathcal{R}$ is no longer successful in balancing the load across the channels, two actions are taken: a new assignment $\mathcal{R}'$ is obtained, optimized for the new bandwidth requirements $\{b'_j\}$, and a number of receivers are tuned to new wavelengths as specified by $\mathcal{R}'$.

In [13] it was assumed that the traffic pattern is slowly and predictably changing over time. In this case, an assignment of receive wavelengths may be precomputed for the expected new traffic conditions. If changes in the traffic pattern are not predictable, the network nodes (or a special node dedicated to managing the network) may monitor packet transmissions on the various channels, and apply statistical techniques to determine whether the overall conditions have changed in a way that significantly affects the optimality of the current wavelength assignment. The problem of determining *when* the wavelength assignment needs to be updated will be considered in the following chapter. Instead here we concentrate on the issues arising once a decision to reconfigure the network has been taken based on a new traffic matrix $\mathbf{T}'$.

The reconfiguration phase will take the network from the current assignment $\mathcal{R}$ to some new assignment $\mathcal{R}'$. We define the distance $\mathcal{D}$ between two wavelength assignments $\mathcal{R}$ and $\mathcal{R}'$ as follows:

$$\mathcal{D}(\mathcal{R}, \mathcal{R}') \quad = \quad N \; - \; \sum_{c=1}^{C} \mid R_c \cap R'_c \mid \tag{5.1}$$

The distance $\mathcal{D}(\mathcal{R}, \mathcal{R}')$ represents the number of receivers that would need to be retuned in order to take the network from wavelength assignment $\mathcal{R}$ to the new assignment $\mathcal{R}'$.

There is a wide range of policies for reconfiguring the network, mainly differing in the tradeoff between the length of the transition period and the portion of the network that becomes unavailable during this period (see [13] for a discussion on similar issues arising in multihop networks). One extreme approach would be to simultaneously retune all the receivers that are assigned new channels under $\mathcal{R}'$. The duration of the transition period is minimized under this approach (it becomes equal to Δ_r), but a significant fraction of

the network may be unusable during this time. At the other extreme, an approach that retunes one receiver at a time minimizes the portion of the network unavailable at any given instant during the transition phase, but maximizes the length of this phase (which now becomes $\mathcal{D}(\mathcal{R}, \mathcal{R}')\Delta_r$). Between these policies at the two ends of the spectrum lie a range of approaches in which two or more receivers are retuned simultaneously.

Let us define a *step* in the reconfiguration phase as an interval of length Δ_r during which one or more receivers are retuned. Let $k(p)$ be the number of steps required under policy p, and let $x_n(p),\ n = 1, \cdots, k(p)$, be the number of receivers retuned during step n for this policy. During the transition period, the network incurs some cost in terms of packet delay, packet loss, packet desequencing, and the control resources involved in receiver retuning. This cost is directly proportional to both the portion of the network that becomes unavailable and the length of the transition period. A measure of this cost that accounts for both these factors is the network unavailable fraction-unavailability length product, which can be obtained as the sum $\sum_{n=1}^{k(p)} \left(\Delta_r \ \frac{x_n(p)}{N}\right)$. But, for any reconfiguration policy p, this sum is equal to:

$$\sum_{n=1}^{k(p)} \left(\Delta_r \ \frac{x_n(p)}{N}\right) \quad = \quad \Delta_r \ \frac{\mathcal{D}(\mathcal{R}, \mathcal{R}')}{N} \qquad \forall\ p \tag{5.2}$$

Thus, regardless of the policy used, the number of retuning operations $\mathcal{D}(\mathcal{R}, \mathcal{R}')$ emerges as an important parameter, one that determines the impact of the reconfiguration phase on the traffic carried by the network.

The rest of the chapter considers the problem of minimizing the number of retuning operations given an initial assignment $\mathcal{R}$ and a new traffic matrix $\mathbf{T}'$. As in [13], we also ignore network specific issues such as how to coordinate the individual steps of the transition phase and inform the nodes of which receivers to retune and when. Instead, we concentrate on an abstract model that hides the details of operation but is applicable to a wide range of network environments.

5.2 Determining the New Wavelength Assignment

Consider a network operating under wavelength assignment $\mathcal{R}$ optimized for traffic matrix $\mathbf{T}$. As traffic varies over time, the matrix is updated to reflect the changes in the traffic pattern. Let $\mathbf{T}'$ be the traffic matrix at the instant reconfiguration is triggered. Our objective is to obtain a new wavelength assignment $\mathcal{R}'$ such that:

1. the new traffic load, as specified by matrix $\mathbf{T}'$ is evenly spread across the C channels, and

2. the number of retunings required to take the network from assignment $\mathcal{R}$ to assignment $\mathcal{R}'$ is as small as possible.

We note that these requirements on $\mathcal{R}'$ represent two conflicting objectives: minimizing the number of retunings alone would result in $\mathcal{R}'$ being the same as $\mathcal{R}$, which may be suboptimal in terms of load balancing; while optimally balancing the load across the C channels might produce a new assignment such that the distance in (5.1) be large.

We distinguish two approaches in constructing a new assignment $\mathcal{R}'$, differing mainly in whether the optimization procedure attempts to satisfy both objectives simultaneously, or one at a time:

- The first approach consists of two steps. The first step is to partition the set of receivers by solving the load balancing problem on matrix $\mathbf{T}'$ *independently* of the initial assignment $\mathcal{R}$. The second step assigns the new subsets of receivers to wavelengths so as to minimize the number of retunings required starting from $\mathcal{R}$. This gives rise to the *Channel Assignment* problem discussed in the next subsection.

- The second approach attempts to solve the load balancing problem on matrix $\mathbf{T}'$, while at the same time minimizing the number of retunings that have to be performed. We will call this the *Constrained Load Balancing* problem.

We now study the two problems, starting with the channel assignment problem.

5.2.1 The Channel Assignment Problem

We consider an initial wavelength assignment $\mathcal{R}$ and a new traffic matrix $\mathbf{T}'$. The first step in the reconfiguration process is to run an approximation algorithm (such as *MULTIFIT* or *LPT*) to obtain a partition $\mathcal{S}' = \{S'_c\}$ of the set of receivers into C sets S'_c, $c = 1, \cdots, C$. This partition $\mathcal{S}'$ is such that the bandwidth requirements (as defined by matrix $\mathbf{T}'$) of the receivers in each set S'_c is approximately the same among the C sets. We note that the approximation algorithm does not distinguish among the various channels. Thus, the output of the algorithm is simply a partition $\mathcal{S}'$ of the set of receivers, *not* a wavelength assignment as defined in (4.1); in other words, there is no association among the receiver subsets S'_c and the available wavelengths.

From $\mathcal{S}'$ we may obtain a new wavelength assignment $\mathcal{R}'$ by mapping each subset S'_c to one of the wavelengths, such that no two subsets map to the same wavelength. Since our objective is to minimize the number of retuning operations during the reconfiguration, the problem of selecting a mapping that results in the least number of retunings arises. We will call this the *Channel Assignment (CA)* problem; it can be formally stated as:

Problem 5.2.1 (CA) *Given an initial wavelength assignment $\mathcal{R} = \{R_c\}$, and a new partition $\mathcal{S}' = \{S'_c\}$ of the set of receivers, find a permutation $(\pi_1, \pi_2, \cdots, \pi_C)$ of $\{1, \cdots, C\}$ such that for the new wavelength assignment $\mathcal{R}' = \{R'_c\}$ with $R'_c = S'_{\pi_c}$, $c = 1, \cdots, C$, the distance $\mathcal{D}(\mathcal{R}, \mathcal{R}')$ is minimum over all possible permutations.*

Problem *CA* is an example of a *bipartite weighted matching* or *assignment* problem [1], when given a weighted bipartite network it is required to find a perfect matching of minimum weight. Several polynomial-time algorithms exist for the assignment problem [1]. The following lemma emphasizes the importance of employing an optimal algorithm for the *CA* problem, by stating that using a simple scheme such as the identity permutation (i.e., letting $R'_c = S'_c$ for all c) may result in an unnecessarily large number of retunings.

Lemma 5.2.1 *Assume that the traffic matrix has changed so that at least one retuning is required under the optimal permutation. Then, the difference between the number of retunings required by the identity permutation and that required by the optimal permutation can be equal to N (the number of receivers) minus one, in the worst case.*

Proof. See Appendix A. □

Unfortunately, this approach to obtaining the new wavelength assignment does not scale well with the size of the network. Even though the *LPT* or *MULTIFIT* algorithms can successfully balance the traffic load across the C channels, this approach performs poorly in terms of the number of retunings required to change the network to the new wavelength assignment. The next lemma states that, even under an optimal solution to the *CA* problem, the number of retunings required may be very large.

Lemma 5.2.2 *Let $\mathcal{R}$ and $\mathcal{S}'$ be the initial wavelength assignment and new partition, respectively, of an arbitrary instance of the* CA *problem for a network with N nodes and C channels. If the optimal solution to this instance yields wavelength assignment $\mathcal{R}'$, $N - C$ is an upper bound on the number of retunings required, i.e.,*

$$\mathcal{D}(\mathcal{R}, \mathcal{R}') \quad \leq \quad N - C \tag{5.3}$$

Proof. See Appendix B. □

The main disadvantage of this solution is that it always satisfies the load balancing objective at the expense of the number of retunings. Furthermore, all the algorithms for the assignment problem are computationally expensive [1], making it difficult to apply them in dynamic high-speed environments. What is needed is a fast algorithm that looks at both objectives at the same time, and which allows the designer to adjust the tradeoff among them in favor of one or the other.

5.2.2 The Constrained Load Balancing Problem

We now consider a different approach to obtaining a new wavelength assignment $\mathcal{R}'$, given an initial assignment $\mathcal{R}$ and a new traffic matrix $\mathbf{T}'$, one that attempts to simultaneously satisfy the two requirements for $\mathcal{R}'$ discussed earlier in this section. This approach gives rise to the *Constrained Load Balancing (CLB)* problem, which can be formally stated as a decision problem:

Problem 5.2.2 (CLB) *Given an initial wavelength assignment $\mathcal{R}$, a traffic matrix $\mathbf{T}'$, and two positive integers K and L, is there a wavelength assignment $\mathcal{R}'$ such that $\sum_{j \in R'_c} b'_j \leq K \ \forall \ c$ and $\mathcal{D}(\mathcal{R}, \mathcal{R}') \leq L$?*

The *CLB* problem is $\mathcal{NP}$-complete because for $L \geq N$ it reduces to the multiprocessor scheduling problem which is $\mathcal{NP}$-complete [9]. We now present a heuristic for the *CLB* problem, which is based on *LPT* [10], an approximation algorithm for the multiprocessor scheduling problem. In describing the heuristic we will use the terminology of [10], i.e., we will refer to processors, tasks, and execution times instead of channels, receivers, and bandwidth requirements, respectively. This will be helpful in referring to the results of [10] to prove certain properties regarding the performance of our heuristic.

Recall that *LPT* first sorts the N tasks in a list $L = (\nu_1, \cdots, \nu_N)$ in decreasing order of their execution times. Initially, each of the first C tasks in the list is assigned to a different processor to execute. Then, whenever a processor completes a task, it scans the list L for the first available task to execute, and this procedure repeats until all tasks have been executed. We modify *LPT* to take into account $\mathcal{R}$, the previous wavelength assignment (i.e., the previous assignment of tasks to processors), by introducing a parameter α, $1 \leq \alpha \leq N$.

Algorithm *Generalized LPT (GLPT)*

Input: Initial wavelength assignment $\mathcal{R} = \{R_c\}$, and new receiver bandwidth requirements $b'_j,\ j = 1, \cdots, N$, derived from the new traffic matrix $\mathbf{T}'$; $\lambda(j)$ denotes the receive wavelength of j under $\mathcal{R}$

Output: New wavelength assignment $\mathcal{R}' = \{R'_c\}$

Parameter: $\alpha, 1 \leq \alpha \leq N$

1. begin
2. Initialize $R'_c = \phi,\ c = 1, \cdots, C$
3. Order the receivers as $(\nu_1, \cdots, \nu_N)$ such that $b'_{\nu_1} \geq \cdots \geq b'_{\nu_N}$
4. $R'_c \leftarrow \{\nu_1\}$ where $\lambda(\nu_1) = \lambda_c$ // assign the first receiver to its previous channel
5. For $j = 1$ to $N - 1$ do
6. Order the channels as $(\lambda_{\pi_1}, \cdots, \lambda_{\pi_C})$ such that $\sum_{l \in R'_{\pi_1}} b'_l \leq \cdots \leq \sum_{l \in R'_{\pi_C}} b'_l$
7. Order the non-assigned receivers as $(\nu_1, \cdots, \nu_{N-j})$ such that $b'_{\nu_1} \geq \cdots \geq b'_{\nu_{N-j}}$
   // If one of the first α receivers was assigned to λ_{π_1} under $\mathcal{R}$, assign it to the same channel
8. For $i = 1$ to $\min\{\alpha, N - j\}$ do
9. If $\lambda(\nu_i) = \lambda_{\pi_1}$ then
10. $R'_{\pi_1} \leftarrow R'_{\pi_1} \cup \{\nu_i\}$
11. goto 5
    // Otherwise, assign the first receiver to λ_{π_1}
12. $R'_{\pi_1} \leftarrow R'_{\pi_1} \cup \{\nu_1\}$
13. end of algorithm

Figure 5.1: The *Generalized LPT* algorithm for the *CLB* problem

The new algorithm also orders the tasks in a list L in decreasing order of their execution times. However, when a processor i searches for a new task to execute (initially, or after the completion of a task) it does not immediately select the first available task in the list. Instead, it considers the first α available tasks in the list (if there are less than α remaining tasks, then all of them are considered). If at least one of these tasks was assigned to the same processor i under the previous assignment $\mathcal{R}$, then the processor starts executing the larger such task, even if it is not the first one in the list of available tasks. Otherwise, if no such task exists, the processor executes the first available task, as in *LPT*. There is one exception to this rule, namely, the first task in the list L (i.e., task ν_1) is always assigned

to its processor under $\mathcal{R}$.

We will call the algorithm just presented the *Generalized LPT (GLPT)* algorithm; its detailed description can be found in Figure 5.1. It can be easily verified that, by implementing appropriate data structures, the complexity of *GLPT* is $O(N \max\{\log N, C, \alpha\})$. We note that *GLPT* reduces to pure *LPT* for $\alpha = 1$. For higher values of α, it is more likely that receivers will be assigned to the same channels as before, and the new wavelength assignment $\mathcal{R}'$ will be closer to $\mathcal{R}$; this may be achieved at the expense of load balancing. By selecting a value for α between 1 and N when implementing *GLPT*, the network designer can achieve the desired tradeoff between the two objectives: load balancing and number of retunings.

The following lemma provides an absolute performance ratio regarding the behavior of *GLPT* in terms of load balancing, *regardless* of the value of parameter α.

Lemma 5.2.3 *Let ω denote the finish time of a multiprocessor schedule constructed by* GLPT *for any value of α, and let $\omega^\star$ denote the optimal finish time for the same set of tasks. Then,*

$$\frac{\omega}{\omega^\star} \quad \leq \quad \frac{3}{2} \; - \; \frac{1}{2C} \tag{5.4}$$

Proof. Let us choose the m, $0 \leq m \leq N$ longest tasks of the set of tasks to be executed and arrange them in a list L which gives the *optimal* solution for these m tasks under this strategy: upon completion of a task, a processor scans the list and starts executing the next available task. Now let us extend L to include all the tasks by adjoining the remaining $N - m$ tasks arbitrarily to L, forming list $L(m)$. Let $\omega(m)$ denote the finish time for the N tasks when using the above strategy on $L(m)$, and let $\omega^\star$ denote the optimal finish time for all N tasks. From [10, Theorem 3] we have that:

$$\frac{\omega(m)}{\omega^\star} \quad \leq \quad 1 \; + \; \frac{1 \; - \; \frac{1}{C}}{1 \; + \; \lceil \frac{m}{C} \rceil} \tag{5.5}$$

Let L' denote the corresponding list of tasks for *GLPT*. This list is not known *a priori*, instead, it is formed dynamically during the execution of the algorithm. However, by construction, the same strategy is followed on L', namely, a processor that becomes idle is always assigned the next available task on L'. Then, the result in (5.4) follows immediately from (5.5) for $m = 1$, since, regardless of the value of the parameter α, list L' is formed by

concatenating some list of $N-1$ tasks (as formed by the algorithm) to the list that gives the optimal solution for the longest task ν_1. □

Finally, we note that the *CLB* problem is a generalized version of the classical multiprocessor scheduling problem [8], whereby it takes a non-negligible amount of time to transfer tasks between processors, and that, because of Lemma 5.2.3, *GLPT* is an approximation algorithm for this new problem.

5.3 Numerical Results

We now compare the two approaches for obtaining a new wavelength assignment $\mathcal{R}'$, given an initial assignment $\mathcal{R}$ and a new traffic matrix $\mathbf{T}'$:

- The first approach is to run *LPT* [10] on the new receiver bandwidth requirements $\{b'_j\}$ derived from matrix $\mathbf{T}'$ to obtain a partition $\mathcal{S}'$ of the set of receivers into C subsets S'_c; we then run the *Shortest Augmenting Paths* algorithm [1] to obtain a solution to the *CA* problem, i.e., to map the subsets S'_c to the actual channels. The running time requirements of this approach are $O(N \log N + N^4)$.

- The second approach is to run algorithm *GLPT(α)*, shown in Figure 5.1, with $\mathcal{R}$ and $\mathbf{T}'$ as input, to directly obtain the new assignment $\mathcal{R}'$; in our experiments, we have used various values for parameter α.

The two performance measures of interest are load balancing and the number of receiver retunings required. Since we do not have a polynomial time solution for the load balancing problem, we compare the two approaches against the lower bound, obtained from matrix $\mathbf{T}'$ as $\frac{\sum_{i,j} t'_{ij}}{C}$; we note that, in general, this lower bound may not be achievable.

Figures 5.2 and 5.3 show the performance of the two approaches in terms of load balancing and number of retunings, respectively, as we vary the number N of nodes in the network; the number of channels remains constant, $C = 10$. Figures 5.4 and 5.5 show results for the same performance measures as the number of channels varies while the number of nodes is kept constant at $N = 120$. To obtain the results shown in Figures $5.2 - 5.5$ we constructed random traffic matrices whose elements were integers uniformly distributed in the range 0 through 20. Each point plotted corresponds to the average of 100 random instances for the stated values of N and C; 95% confidence intervals have also been computed, but they are so narrow that they are not plotted in the figures.

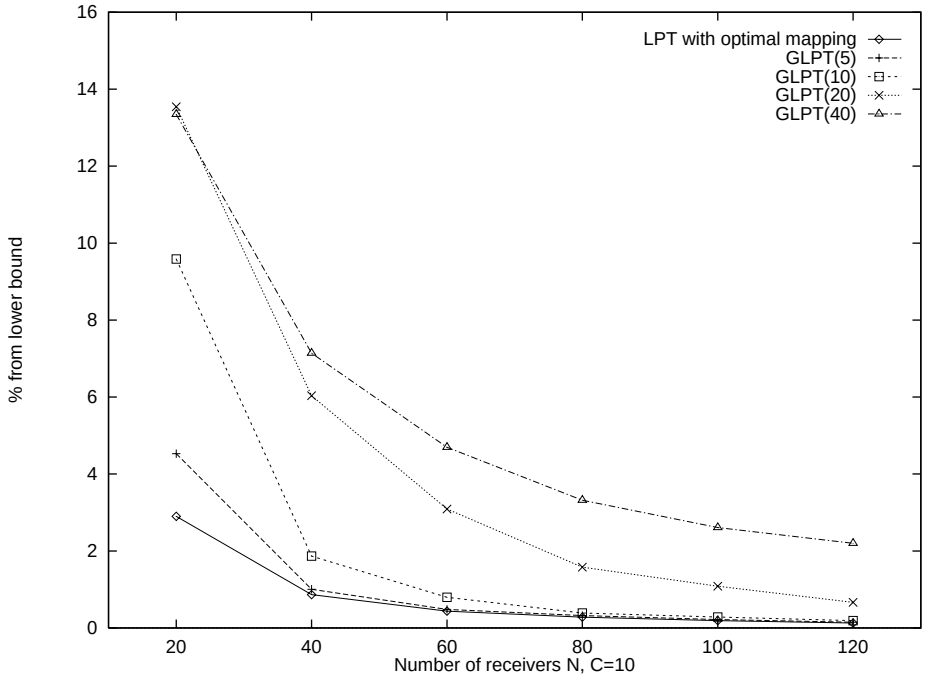


Figure 5.2: Algorithm comparison on load balancing ($C = 10$ channels, random traffic matrices)

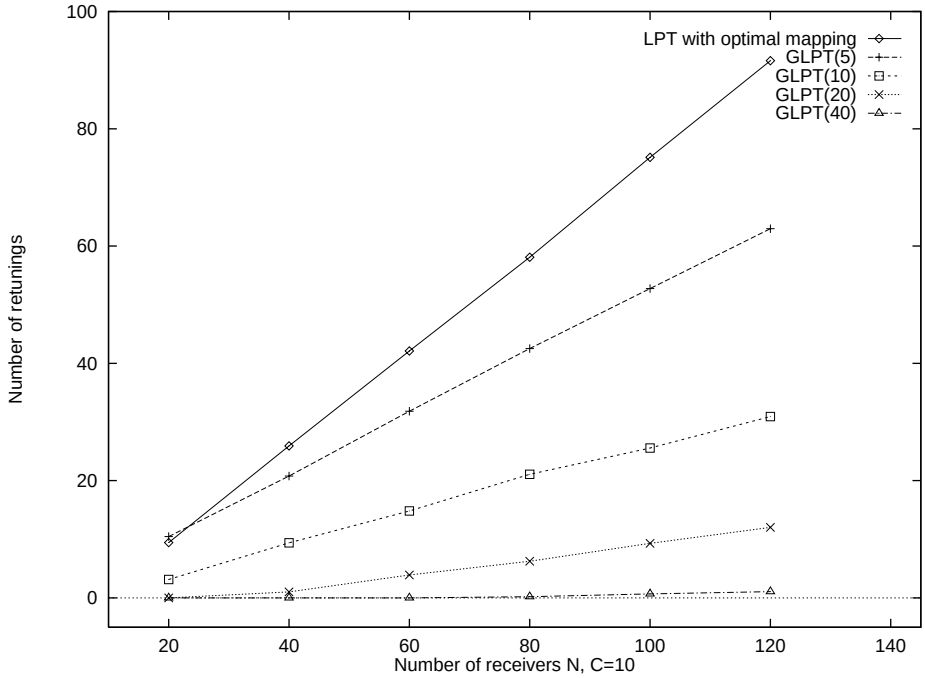


Figure 5.3: Algorithm comparison on number of retunings ($C = 10$ channels, random traffic matrices)

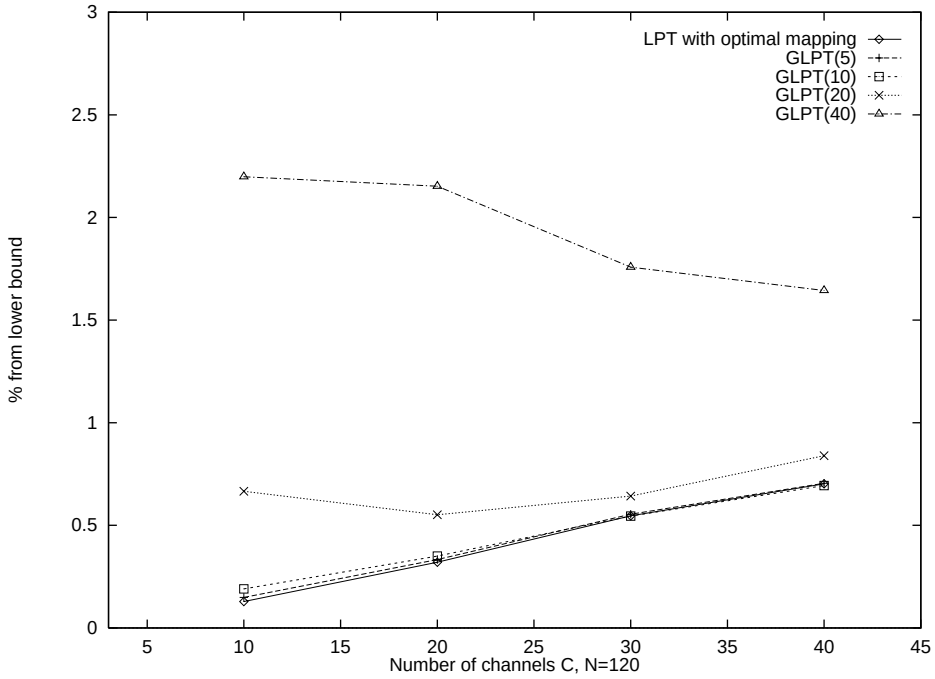


Figure 5.4: Algorithm comparison on load balancing ($N = 120$ nodes, random traffic matrices)

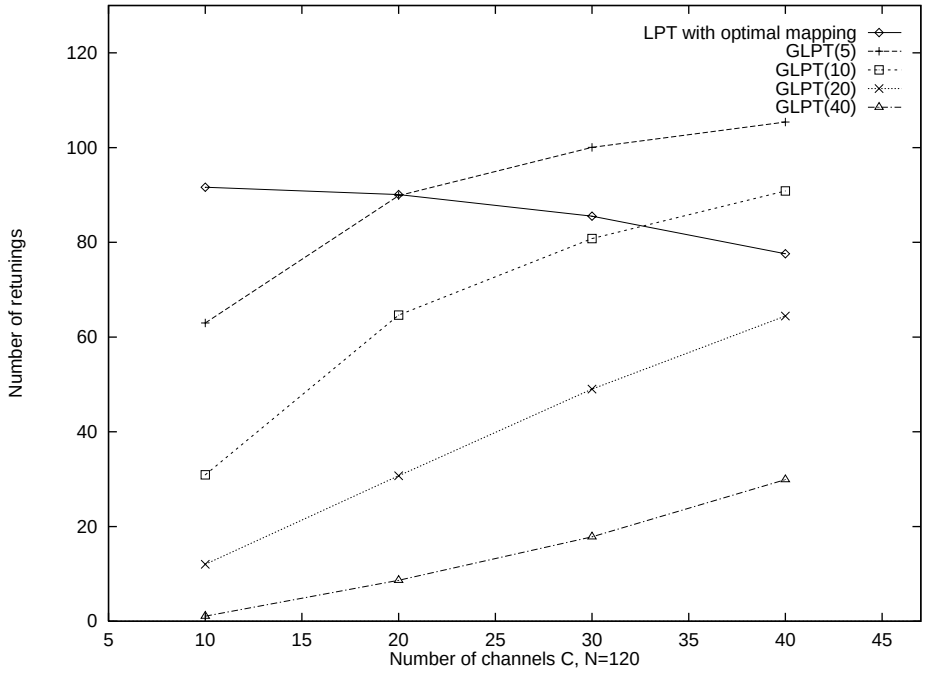


Figure 5.5: Algorithm comparison on number of retunings ($N = 120$ nodes, random traffic matrices)

Our first observation from Figures 5.2 and 5.4 is that the first approach (i.e., employing *LPT* for load balancing and then solving the channel assignment problem), provides the best performance in terms of load balancing, as expected. However, algorithm *GLPT* with $\alpha = 5$ (*GLPT(5)*) performs almost identical to *LPT*, while *GLPT(10)* is also very close to *LPT*. As α increases, *GLPT* starts behaving sub-optimally in terms of load balancing, as expected. However, even when α is as large as 40, *GLPT* is never more than 14% away from the lower bound, and in some cases it is as close as 3%. In fact, because of Lemma 5.2.3, *GLPT* is guaranteed to always be within 50% from the optimal, regardless of the value of parameter α.

Let us now turn our attention to Figure 5.3 which plots the average number of retunings as a function of the size N of the network. We observe that the first approach always requires the most number of retunings, and that its retuning requirements increase linearly with the size of the network. Furthermore, the expected fraction of receivers that need to be retuned increases with the number of nodes, from 50% when $N = 20$, to 75% when $N = 120$. This behavior suggests that the approach is not scalable, since, for large N, either the duration of the reconfiguration phase, or the fraction of the network that becomes unavailable, will be significant. The behavior of this approach in terms of number of retunings is in agreement with intuition: *LPT* is very successful in balancing the load of the network, but it does not take into account the previous wavelength assignment. As a result, the distance between the initial and target assignments tends to be large. We note also that, for all values of N, the expected number of retunings is very close to the upper bound in Lemma 5.2.2.

From the same figure we see that, for small values of α, algorithm *GLPT* requires a number of retunings which also increases linearly with the size of the network. However, the rate of increase is much slower (for instance, when $\alpha = 5$, about 50% of the receivers are retuned for all values of N, while when $\alpha = 10$, about 20% of the receivers are retuned on average). As α increases, the behavior of *GLPT* improves dramatically. For $\alpha = 20$, the number of retunings does increase with N, but it is always less than 10, while when $\alpha = 40$, only about one receiver needs to be retuned, *independently* of the number N of nodes. In fact, doubling the value of parameter α reduces the number of retunings to less than half its previous value. As a result, it does not make sense to use a value of α that is, in this case, larger than 40, since doing so may increase the running time requirements of algorithm *GLPT* without any significant effect on the number of retunings. This behavior of *GLPT*

can be explained by noting that, for sufficiently large values of α, *GLPT* will assign most of the receivers to their previous channels. Only a few of the receivers with the smallest requirements will be assigned to new channels if it is necessary to do so in order to keep the channels balanced. This feature of *GLPT*, namely, that the receivers with the smallest requirements under the new traffic pattern are more likely to be retuned, is highly desirable. This is because it implies that the reconfiguration will affect the part of the network that is least utilized, minimizing the impact of the transition phase (in terms of packet loss, delay, etc.) on the overall traffic carried by the network.

In Figure 5.5 we plot the number of retunings required against the number of channels, for $N = 120$. We note that the first approach always requires a number of receivers to be retuned which is very close to the upper bound $N - C$ of Lemma 5.2.2. On the other hand, the number of retunings required by *GLPT* increases almost linearly with C for all values of α; also, larger values of α result in a smaller number of retunings, as expected. This result, combined with our previous observations, indicates that, for certain values of parameter α (in this case, for $20 \leq \alpha \leq 40$), *GLPT* provides a scalable approach to reconfiguring the network since (a) it achieves a guaranteed level of performance in terms of load balancing, (b) its retuning requirements are low, and more importantly, (c) the number of retunings scales with the number of channels, *not* the number of nodes in the network.

The results plotted in Figures 5.2 – 5.5 were obtained by randomly selecting the initial traffic matrix $\mathbf{T}$, and then randomly selecting the target matrix $\mathbf{T}'$, independently of $\mathbf{T}$. In practice, the new traffic matrix $\mathbf{T}'$ will be related to the old matrix $\mathbf{T}$, differing only by the changes in the traffic demands that have taken place in the time interval between successive reconfiguration instants. To study the relative performance of the two approaches under a model that more closely captures the characteristics of a realistic traffic scenario, we have run a set of experiments in which we have used Brownian motion to model the change in the source-destination traffic demands.

In the new model, the initial random matrix $\mathbf{T}$ was constructed as before. This matrix was then evolved through a series of steps to obtain the target matrix $\mathbf{T}'$. The Brownian motion was modeled by using two asymmetric probabilities: the probability of the particle moving towards the "wall" and the probability of moving away from the "wall", the "wall" being either the lower or the upper limit on the source-destination traffic demand (to obtain results comparable to those in Figures 5.2 - 5.5, we used 0 and 20, respectively, for the lower and upper limit on the traffic demands). At every step of the evolution process,

each element of the demand matrix $\mathbf{T}$ is treated as a one-dimensional Brownian particle. In our model, the probability of moving away from the wall (set to 0.5) was higher than the probability of moving towards the wall (set to 0.2) [1]. Thus, a particle always has a "direction of likely movement". Based on these probabilities, a newly generated random number at each step determines whether the bandwidth demand for a certain source-destination pair will increase by an amount δ, decrease by δ, or remain the same, independently of the other elements; in our experiments we let $\delta = 1$. If an element hits the upper or lower limit, its "direction of likely movement" is reversed. After performing several steps ($\sim 10 - 20$) in this manner, the resulting matrix was taken as the new traffic matrix $\mathbf{T}'$.

The results from the Brownian model are shown in Figures 5.6 - 5.9. As we can see, the new model had little effect on the behavior of the various algorithms, confirming our conclusions regarding the relative performance of the two approaches.

5.4 Concluding Remarks

We considered the problem of updating the bandwidth allocation in single-hop WDM networks to accommodate varying traffic demands, by retuning a set of slowly tunable receivers. Our objective was to balance the traffic load across all channels, while keeping the number of retunings to a minimum. We studied a straightforward approach to obtaining a new wavelength assignment, one that employs well-known algorithms to satisfy the two requirements independently of each other, and we have shown that it is not scalable. We then presented a new algorithm that attempts to construct the new wavelength assignment in a way that simultaneously achieves the stated objectives. The algorithm provides for tradeoff selection between the two requirements, and scales well with the size of the network.

We presumed that the decision to perform the reconfiguration of the network will have already been made by the time GLPT is invoked. The following chapter will address the important question of a *Reconfiguration Policy* which will decide when it is advantageous to reconfigure the network in order to improve the overall throughput.

[1] Note that these probabilities need not sum up to unity; if they do not, the difference represents the probability that the particle will not change its position during a step.

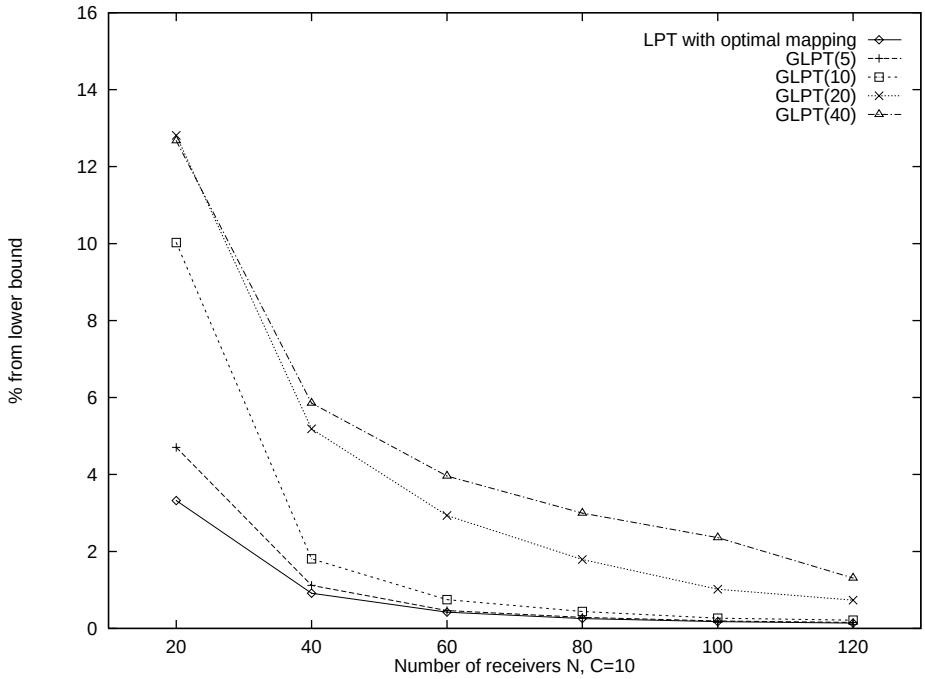


Figure 5.6: Algorithm comparison on load balancing ($C = 10$ channels, Brownian model)

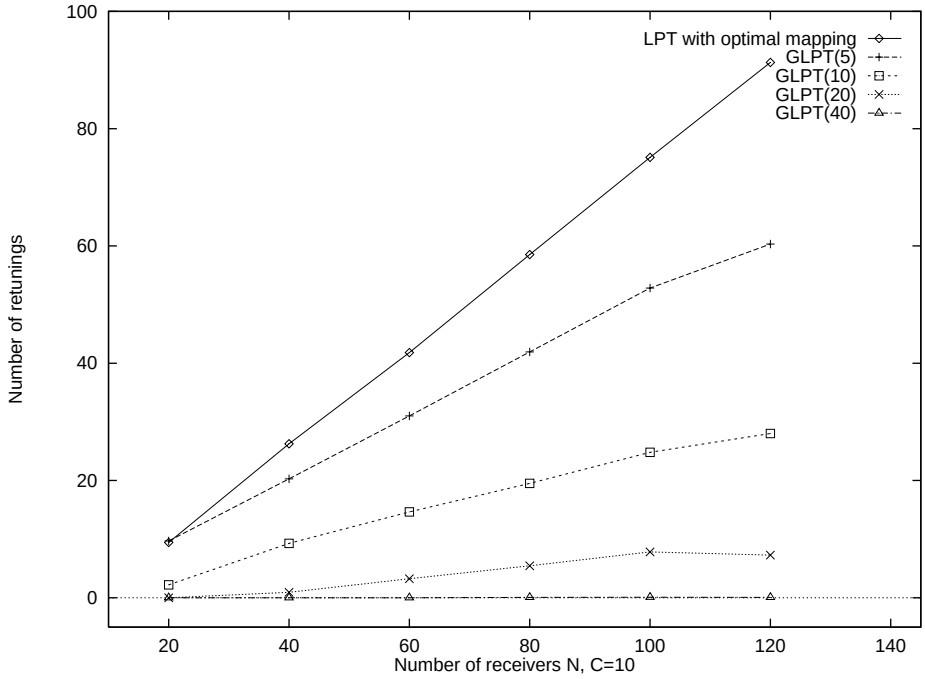


Figure 5.7: Algorithm comparison on number of retunings ($C = 10$ channels, Brownian model)

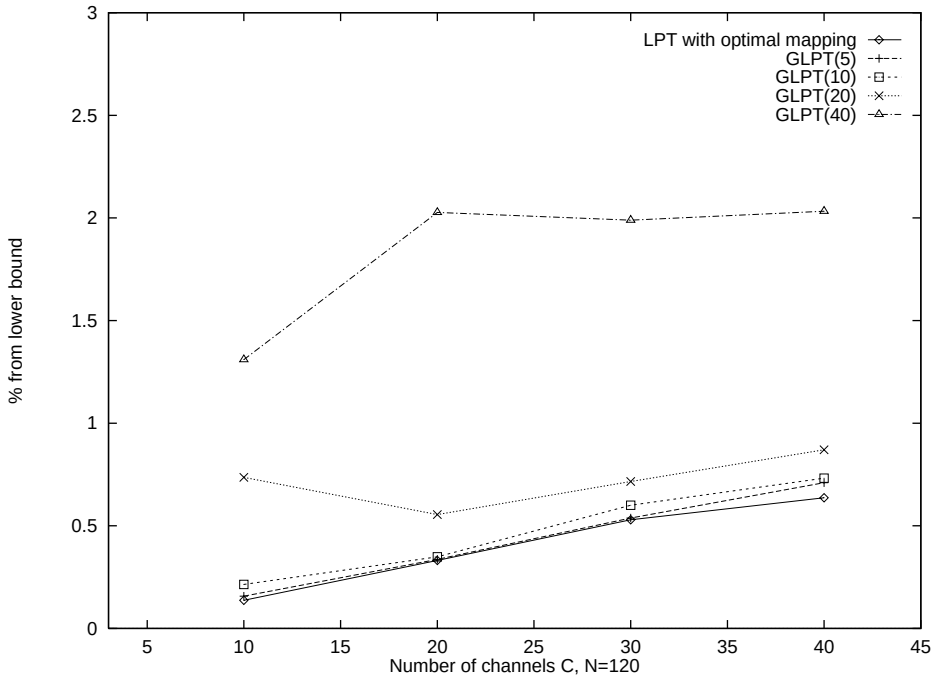


Figure 5.8: Algorithm comparison on load balancing ($N = 120$ nodes, Brownian model)

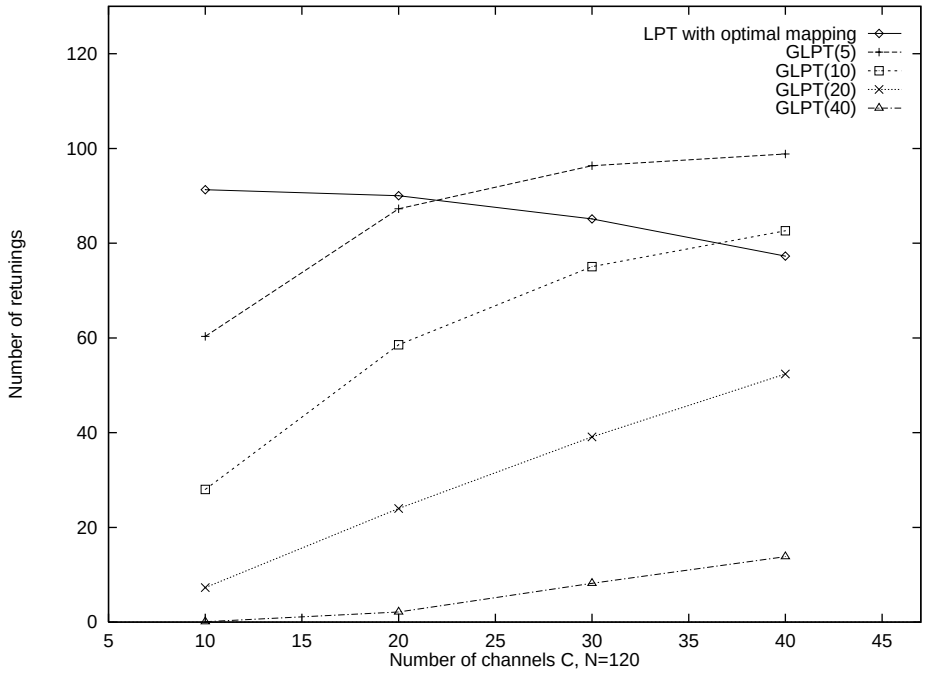


Figure 5.9: Algorithm comparison on number of retunings ($N = 120$ nodes, Brownian model)

Chapter 6

Optimal Reconfiguration Policies

During the reconfiguration phase, while the network makes a transition from one WLA to another, some cost is incurred in terms of packet delay, packet loss, packet desequencing, and the control resources involved in receiver retuning. Clearly, receiver retunings should not be very frequent, since unnecessary retunings affect the performance encountered by the users. Hence, it is desirable to minimize the number of network reconfigurations. However, postponing a necessary reconfiguration also has adverse effects on the overall performance. Since the network does not operate at an optimal point in terms of load balancing, it takes longer to clear a given set of traffic demands, causing longer delays and/or buffer overflows, as well as a decrease in the network's traffic carrying capacity (refer also to the stability condition in [26]). Similarly, if the decisions are made merely by considering the degree of load balancing, even tiny changes in the traffic demands can lead to constant reconfiguration, thereby significantly hurting network performance. Consequently, it is important to have a performance criterion which can capture the above tradeoffs in an appropriate manner and allow their simultaneous optimization.

In this chapter we develop a novel, systematic, and flexible framework in which to view and contrast reconfiguration policies. Specifically, we formulate the problem as a Markovian Decision Process and we show how an appropriate selection of reward and cost functions can achieve the desired balance between various performance criteria of interest. However, because of the huge state space of the underlying Markov process, it is impossible to directly apply appropriate numerical methods to obtain an optimal policy. We therefore develop an approximate model with a manageable state space, which captures the pertinent properties of the original model. We also demonstrate that results from Markov Decision

Process theory can be applied in an efficient way to obtain reconfiguration policies for networks of large size.

The rest of this chapter is organized as follows. In Section 6.1 we present a model of the broadcast WDM network under study. In Section 6.2 we formulate the reconfiguration problem as a Markovian Decision Process, and we discuss the issues of obtaining an optimal policy. We present numerical results in Section 6.3, where we also compare the optimal policy against a class of threshold policies.

6.1 The Broadcast WDM Network

As discussed earlier, we consider a packet-switched single-hop lightwave network with N nodes, and one transmitter-receiver pair per node. The nodes are physically connected to a passive broadcast optical medium that supports $C < N$ wavelengths, $\lambda_1, \cdots, \lambda_C$. Both the transmitter and the receiver at each node are tunable over the entire range of available wavelengths. However, the transmitters are *rapidly tunable*, while the receivers are *slowly tunable*. We refer to this tunability configuration as *rapidly tunable transmitter, slowly tunable receiver* (RTT-STR). Although we will only consider RTT-STR networks in this chapter, we note that all our results can be easily adapted to the dual configuration, STT-RTR.

During normal operation, each of the slowly tunable receivers is assumed to be fixed to a particular wavelength. The network operates by having each node employ a media access protocol, such as HiPeR-ℓ, that requires tunability only at the transmitting end. Nodes use HiPeR-ℓ to make reservations, and can schedule packets for transmission using algorithms that can effectively mask the (relatively short) latency of tunable transmitters [20].

For the purposes of this chapter we now define the *degree of load balancing (DLB)* $\phi(\mathcal{R}, \mathbf{T})$ for a network with traffic matrix $\mathbf{T}$ operating under WLA $\mathcal{R}$ as:

$$(1 \; + \; \phi(\mathcal{R}, \mathbf{T})) \; \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} t_{ij}}{C} \quad = \quad \max_{c=1,\cdots,C} \left\{ \sum_{i=1}^{N} \sum_{j \in R_c} t_{ij} \right\} \tag{6.1}$$

The right hand side of (6.1) represents the bandwidth requirement of the dominant (i.e., most loaded) channel, while the second term in the left hand side of (6.1) represents the lower bound, with respect to load balancing, for any WLA for traffic matrix $\mathbf{T}$. Thus, the

DLB is a measure of how far away WLA $\mathcal{R}$ is from the lower bound. If $\phi = 0$, then the load is perfectly balanced, and each channel carries an equal portion of the offered traffic, while when $\phi > 0$, the channels are not equally loaded. In other words, the DLB characterizes the efficiency of the network in meeting the traffic demands denoted by matrix $\mathbf{T}$ while operating under WLA $\mathcal{R}$: the higher the value of ϕ, the less efficient the WLA is.

6.1.1 The Transition Phase

In order to more efficiently utilize the bandwidth of the optical medium as traffic varies over time, a new WLA may be sought that distributes the new load more equally among the channels. We will refer to the transition of the network from one WLA to another as *reconfiguration*. In general, we assume that reconfiguration is triggered by changes in the traffic matrix $\mathbf{T}$. When such a change occurs, the following actions must be taken:

1. a new WLA for the new traffic matrix must be determined,
2. a decision must be made on whether or not to reconfigure the network by adopting the new WLA, and
3. if the decision is to reconfigure, the actual retuning of receivers must take place.

The first issue was addressed in the previous chapter, where we developed the *GLPT* algorithm which takes as input the current WLA $\mathcal{R}$ and the new traffic matrix $\mathbf{T}'$, and determines the new WLA. The rest of this chapter addresses the second problem of determining whether the changes in traffic conditions warrant the reconfiguration of the network to the new WLA. We now discuss the third issue of receiver retuning.

Let $\mathcal{R}$ and $\mathbf{T}$ be the current WLA and traffic matrix, respectively, and let $\mathbf{T}'$ be the new traffic matrix. Let $\mathcal{R}'$ be the new WLA computed by the *GLPT* algorithm with $\mathcal{R}$ and $\mathbf{T}'$ as input. Assuming that a decision has been made to reconfigure, there will be a transition phase during which a set of receivers is retuned to take the network from the current WLA $\mathcal{R}$ to the new WLA $\mathcal{R}'$. The distance $\mathcal{D}(\mathcal{R}, \mathcal{R}')$ (see 5.1) represents the number of receivers that need to be taken off-line for retuning during the transition phase.

There is a wide range of strategies for retuning the receivers, mainly differing in the tradeoff between the length of the transition period and the portion of the network that becomes unavailable during this period (see [13] for a discussion of similar issues in multihop networks). One extreme approach would be to simultaneously retune all the receivers which

are assigned new channels under $\mathcal{R}'$. The duration of the transition phase is minimized under this approach (it becomes equal to the receiver tuning latency), but a significant fraction of the network may be unusable during this time. At the other extreme, a strategy that retunes one receiver at a time minimizes the portion of the network unavailable at any given instant during the transition phase, but it maximizes the length of this phase (which now becomes equal to the receiver tuning latency times the distance $\mathcal{D}(\mathcal{R}, \mathcal{R}')$). Between these two ends of the spectrum lie a range of strategies in which two or more receivers are retuned simultaneously.

While the receiver of, say, node j, is being retuned to a new wavelength, it cannot receive data, and thus, any packets sent to node j are either buffered or lost. If, on the other hand, the network nodes are aware that node j is in the process of retuning its receiver, they can refrain from transmitting packets to it. In this case, packets destined to node j will experience longer delays while waiting for the node to become ready for receiving again. Moreover, packets for j arriving to the various transmitters during this time cannot be serviced, and may cause buffer overflows. This increase in delay and/or packet loss is the penalty incurred for reconfiguring the network.

We note that, in general, the reconfiguration penalty associated with retuning a given number D of receivers will depend on the actual retuning strategy employed (e.g., simultaneously retuning all D receivers versus retuning one receiver at a time). Furthermore, the relative penalty of the various retuning strategies is a function of system parameters such as the receiver tuning latency and the offered load. Determining the best retuning strategy for a given region of network operation will be considered in the following chapter. In this chapter, we instead develop an abstract model that includes a cost function to account for the reconfiguration penalty. Our model is flexible in that the cost function can be appropriately selected to fit any given strategy.

6.2 Markov Decision Process Formulation

6.2.1 Reconfiguration Policies

We define the state of the network as a tuple $(\mathcal{R}, \mathbf{T})$. $\mathcal{R}$ is the current WLA, and $\mathbf{T}$ is a matrix representing the prevailing traffic conditions. Changes in the network state occur at instants when the matrix $\mathbf{T}$ is updated. Since we have assumed that future traffic

changes only depend on the current values of the elements of $\mathbf{T}$, the process $(\mathcal{R}, \mathbf{T})$ is a semi-Markov process. Let $\mathcal{M}$ be the process embedded at instants when the traffic matrix changes. Then, $\mathcal{M}$ is a discrete-time Markov process. Our formulation is in terms of the Markov process $\mathcal{M}$.

A network in state $(\mathcal{R}, \mathbf{T})$ will enter state $(\mathcal{R}', \mathbf{T}')$ if the traffic matrix changes to $\mathbf{T}'$. Implicit in the state transition is that the system makes a decision to reconfigure to WLA $\mathcal{R}'$. In order to completely define the Markovian state transitions associated with our model, we need to establish *next WLA* decisions. The decision is a function of the current state and is denoted by $d[(\mathcal{R}, \mathbf{T})]$. Setting $d[(\mathcal{R}, \mathbf{T})] = \mathcal{R}_{next}$ implies that if the system is in state $(\mathcal{R}, \mathbf{T})$ and the traffic demands change, the network should be reconfigured into WLA $\mathcal{R}_{next}$. Note that WLA $\mathcal{R}_{next}$ can be the same as $\mathcal{R}$, in which case the decision is not to reconfigure. Therefore, for each state $(\mathcal{R}, \mathbf{T})$ there are two alternatives: either the network reconfigures to WLA $\mathcal{R}'$ obtained by the *GLPT* algorithm with $\mathcal{R}$ and $\mathbf{T}'$ as inputs (in which case the new state will be $(\mathcal{R}', \mathbf{T}')$), or it maintains the current WLA (in which case the new state will be $(\mathcal{R}, \mathbf{T}')$). The set of decisions for all network states defines a *reconfiguration policy.*

To formulate the problem as a Markov Decision Process, we need to specify reward and cost functions associated with each transition. Consider a network in state $(\mathcal{R}, \mathbf{T})$ that makes a transition to state $(\mathcal{R}', \mathbf{T}')$. The network acquires an *immediate expected reward* equal to $\alpha[\phi(\mathcal{R}', \mathbf{T}')]$, where $\alpha(\cdot)$ is a non-increasing function of $\phi(\mathcal{R}', \mathbf{T}')$, the DLB of WLA $\mathcal{R}'$ with respect to the new traffic matrix $\mathbf{T}'$. Also, if $\mathcal{R}' \neq \mathcal{R}$, a *reconfiguration cost* equal to $\beta[\mathcal{D}(\mathcal{R}, \mathcal{R}')]$ is incurred, where $\beta(\cdot)$ is an non-decreasing function of the number of receivers that have to be retuned to take the network to the new WLA $\mathcal{R}'$. In other words, a switching cost is incurred each time the network makes a decision to reconfigure. We assume that the rewards and costs are bounded, i.e.:

$$\alpha_{min} \leq \alpha[\phi(\mathcal{R}', \mathbf{T}')] \leq \alpha_{max} \quad \text{and} \quad 0 \leq \beta_{min} \leq \beta[\mathcal{D}(\mathcal{R}, \mathcal{R}')] \leq \beta_{max} \tag{6.2}$$

where α_{min}, α_{max}, β_{min} and β_{max} are real numbers.

The problem is how to reconfigure the network sequentially in time, so as to maximize the expected reward minus the reconfiguration cost over an infinite horizon. Let $(\mathcal{R}^{(k)}, \mathbf{T}^{(k)})$ denote the state of the network immediately after the k-th transition, $k = 1, 2, \cdots$. Let also Z be the set of admissible policies. The network reconfiguration problem can then be formally stated as follows (note that $\mathcal{D}(\mathcal{R}, \mathcal{R}) = 0$):

Problem 6.2.1 *Find an optimal policy $z^\star \in Z$ that maximizes the expected reward*

$$F \;=\; \lim_{k\to\infty}\; \frac{1}{k}\; E\left\{\sum_{l=1}^{k} \alpha[\phi(\mathcal{R}^{(l)}, \mathbf{T}^{(l)})] \;-\; \beta[\mathcal{D}(\mathcal{R}^{(l-1)}, \mathcal{R}^{(l)})]\right\} \tag{6.3}$$

The first term in the right hand side of (6.3) is the reward obtained by using a particular WLA, and the second term is the cost incurred at each instant of time that reconfiguration is performed. The presence of a reward which increases as the DLB ϕ decreases (i.e., as the load is better balanced across the channels) provides the network with an incentive to associate with a WLA that performs well for the current traffic load. On the other hand, the introduction of a cost incurred at each reconfiguration instant discourages frequent reconfigurations. Thus, the overall reward function captures the fundamental tradeoff between the DLB and frequent retunings involved in the reconfiguration problem.

6.2.2 Motivation

We now motivate the above formulation by showing how an appropriate selection of reward and cost functions yields various performance criteria of interest. Typically, such selection can be based on either measurements of an existing network or simulations.

One important performance objective is to minimize the probability that the network will not be able to handle the offered traffic load. This is equivalent to minimizing the probability that the DLB increases beyond a maximum value ϕ_{max}. Let γ_{max} be the maximum traffic load (in packets per packet transmission time) that will ever be allowed into the network. By definition of the DLB in (6.1), the load offered to the dominant channel when the DLB is ϕ will be $(1+\phi)\gamma_{max}/C$. Since each channel can clear at most one packet per packet time, we have that $1+\phi_{max} \leq C/\gamma_{max}$. Therefore, this objective can be achieved by selecting $\alpha(\cdot)$ a function as shown in Figure 6.1 and β_{max} small.

Another performance measure of interest is the probability of unnecessary reconfigurations. By making β_{min} and β_{max} large, and letting $\alpha(\cdot)$ a slowly decreasing function as ϕ increases, minimizing the probability of unnecessary reconfigurations becomes equivalent to maximizing (6.3). Similarly, the objective to minimize the probability that the portion of the network that becomes unavailable due to reconfiguration is greater than a certain threshold D_{max} can be achieved by letting β_{min} small, β_{max} large, and selecting $\beta(\cdot)$ a function as shown in Figure 6.2.

It is also possible to select rewards and costs that reflect performance measures

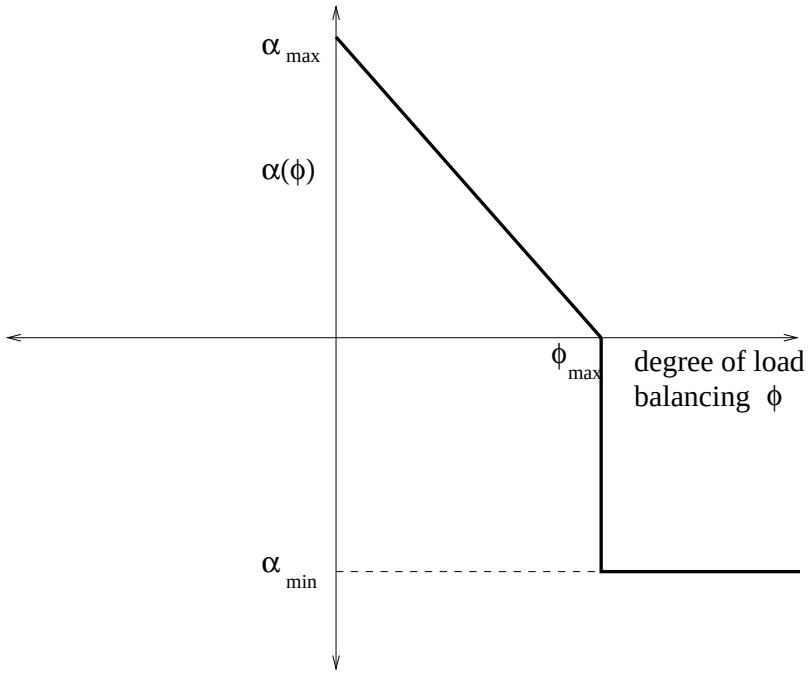


Figure 6.1: Reward function to minimize the probability that the DLB is above a certain threshold ϕ_{max}

such as throughput, delay, packet loss, or the control resources involved in receiver retuning. For example, a reward function of the form $A/(1+\phi)$ may, depending on the value of parameter A, capture either the throughput or average packet delay experienced while the network operates with a DLB equal to ϕ. On the other hand, using a cost which is proportional to the number $D = \mathcal{D}(\mathcal{R}, \mathcal{R}')$ of retunings (i.e., $\beta(\mathcal{D}(\mathcal{R}, \mathcal{R}')) = BD$) can account for the control requirements for retuning the receivers, or for the data loss incurred during reconfiguration. Furthermore, parameter B can be chosen based on which of the retuning strategies discussed in Section 6.1.1 is employed. Thus, network designers can select in a unified fashion appropriate rewards and costs to achieve the desired balance among the various performance criteria of interest.

For the case $\beta_{max} = 0$, the problem of finding an optimal policy is trivial, since it is optimal for the network to associate with the WLA which best balances the offered load at each instant in time. This is because the evolution of the traffic matrix $\mathbf{T}$ is not affected by the network's actions and reconfigurations are free. However, when $\beta_{max} > 0$, there is a conflict between *future reconfiguration costs incurred* and *current reward obtained*, and it is not obvious as to what constitutes an optimal policy. We also note that as $\beta_{min} \to \infty$,

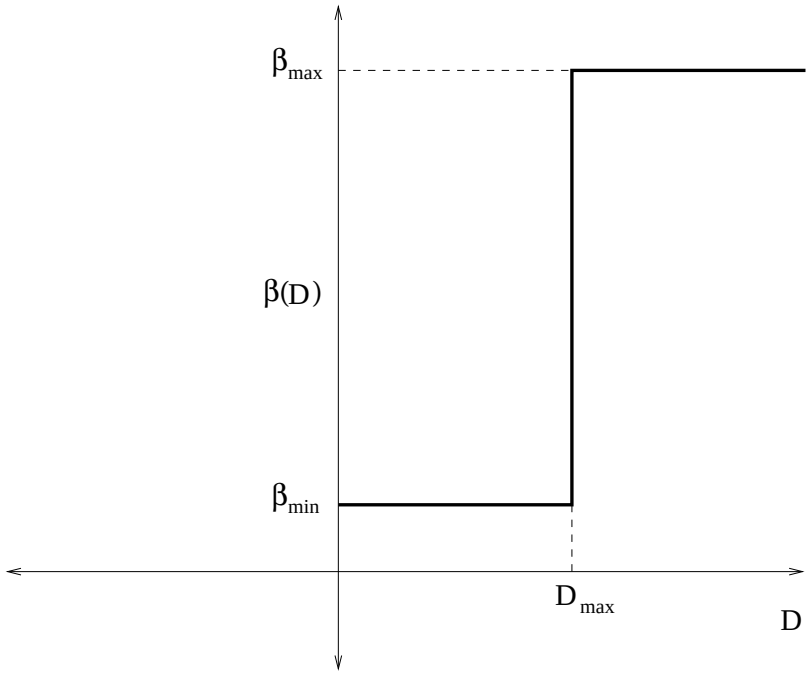


Figure 6.2: Cost function to minimize the probability that network unavailability exceeds a certain threshold D_{max}

the optimal policy would be to never reconfigure, since this is the only policy for which the expected reward in (6.3) would be non-negative. Again, however, the point (i.e., the smallest value of β_{min}) at which this policy becomes optimal is not easy to determine, as it depends on the transition probabilities of the underlying Markov chain.

One way to determine an optimal policy is to use Howard's Policy-Improvement Algorithm (see Appendix C and [?, ST:howard-dynamic]. A difficulty in applying the policy-iteration algorithm to the Markov process $\mathcal{M}$ is that its running time per iteration is dominated by the complexity of solving a number of linear equations in the order of the number of states in the Markov chain. Even if we restrict the elements of traffic matrix $\mathbf{T}$ to be integers [1] and impose an upper bound on the values they can take, the potential number of states $(\mathcal{R}, \mathbf{T})$ is so large that the policy-iteration algorithm cannot be directly applied to anything but networks of trivial size. In the next subsection we show how to overcome this problem by making some simplifying assumptions that will allow us to set up a new Markov process whose state space is manageable.

[1] If the elements of $\mathbf{T}$ are real numbers, then $\mathcal{M}$ becomes a continuous-state process and the policy-iteration algorithm cannot be applied.

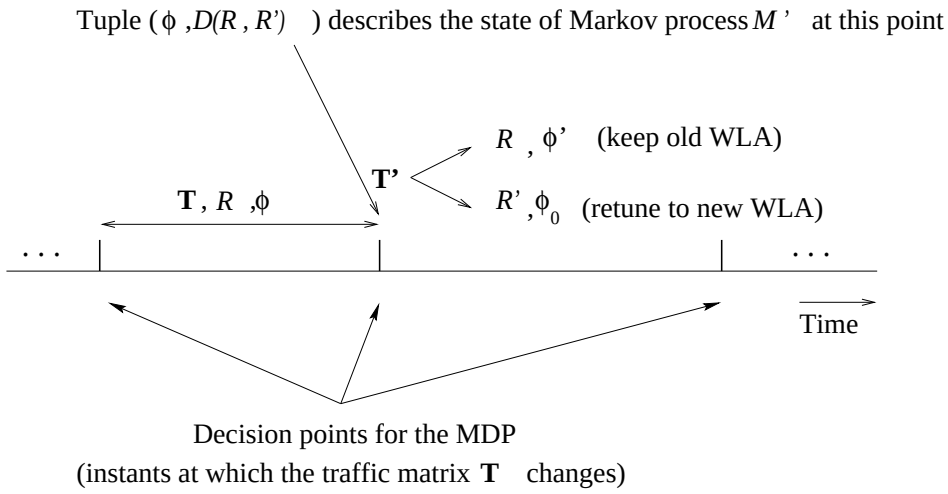


Figure 6.3: State of the new Markov process $\mathcal{M}'$

6.2.3 Alternative Formulation

Consider a network in state $(\mathcal{R}, \mathbf{T})$, and a new traffic matrix $\mathbf{T}'$ for which the WLA obtained with the *GLPT* algorithm is $\mathcal{R}'$. A closer examination of the reward function in (6.3) reveals that the immediate reward acquired when the network makes a transition does not depend on the actual values of the traffic elements or the actual WLAs involved, but only on the values of the DLBs $\phi(\mathcal{R}, \mathbf{T}')$ and $\phi(\mathcal{R}', \mathbf{T}')$, and the distance $\mathcal{D}(\mathcal{R}, \mathcal{R}')$. Thus, we make the simplifying assumption that the decision to reconfigure will also depend on the DLBs and the distance only. This is a reasonable assumption, since it is the DLB, not the actual traffic matrix or WLA that determine the efficiency of the network in satisfying the offered load. Similarly, it is the number of retunings that determines the reconfiguration cost, not the actual WLAs involved.

Based on these observations, we now introduce a new process embedded, as Markov process $\mathcal{M}$, at instants when the traffic matrix changes, as illustrated in Figure 6.3. The state of this process is defined to be the tuple (ϕ, D), where ϕ is the DLB achieved by the current WLA with respect to the current traffic matrix, and D is the number of retunings required if the network were to reconfigure. Transitions in the new process have the Markovian property, since they are due to changes in the traffic matrix which, in turn, are Markovian. However, as defined, the process is a continuous-state process since, in general, the DLB ϕ is a real number. In order to apply Howard's algorithm (see Appendix C.) we need a discrete-state process. We obtain such a process by using discrete values for random

variable ϕ as follows.

By definition (refer to expression (6.1)), the DLB ϕ can take any real value between 0 and $C-1$, where C is the number of channels in the network [2]. We now divide the interval $[0, C-1]$ into a number $K+1$ of non-overlapping intervals $[\phi_0^{(l)}, \phi_0^{(u)}), [\phi_1^{(l)}, \phi_1^{(u)}), \cdots, [\phi_K^{(l)}, \phi_K^{(u)}]$, where $\phi_k^{(l)}$ and $\phi_k^{(u)}$ are the lower and upper values of interval $k, k = 0, \cdots, K$, and: $\phi_k^{(l)} < \phi_k^{(u)}$, $\phi_0^{(l)} = 0$, $\phi_k^{(u)} = \phi_{k+1}^{(l)}$, and $\phi_K^{(u)} = C - 1$. Let ϕ_k denote the midpoint of interval k. We now define a new discrete-state process $\mathcal{M}'$ with state (ϕ_k, D). We will use state (ϕ_k, D) to represent any state (ϕ, D) of the continuous-state process such that $\phi_k^{(l)} \leq \phi < \phi_k^{(u)}$. Clearly, the larger the number K of intervals, the better the approximation.

Before we proceed, we make one further refinement to the new discrete-state process $\mathcal{M}'$. We note that the *GLPT* algorithm in [3] is an approximation algorithm for the load balancing problem, and it guarantees that the DLB of the WLA obtained using the algorithm will never be more than 50% away from the degree of load balancing of the optimal WLA. The importance of this result is as follows. Consider a network in which the traffic matrix changes in such a way that the current WLA provides a DLB ϕ for the new traffic matrix such that $\phi < 0.5$. Based on the guarantee provided by algorithm *GLPT*, we can safely assume that the load is well balanced and avoid a reconfiguration. This is because the network will incur a cost for reconfiguring, without any assurance that the new DLB will be less than ϕ. Therefore, we choose to let $\phi_0^{(u)} = 0.5$, and therefore the midpoint for the first interval is $\phi_0 = 0.25$. We will call any state (ϕ_0, D) a *balanced* state since the offered load is balanced within the guarantees of the *GLPT* algorithm.

We now specify decision alternatives, as well as reward and cost functions associated with each transition in the new process $\mathcal{M}'$. Consider a network in state (ϕ_k, D). At the instant the traffic matrix changes, the network has two options. It may maintain the current WLA, in which case it will make a transition into state (ϕ_l, D'), where ϕ_l is the DLB of the current WLA with respect to the new traffic matrix, and D' is the new distance. Or, it will reconfigure into a new WLA. In the latter case, the network will move into state (ϕ_0, D''), since its new DLB is guaranteed to be less than 0.5. When the network makes

[2] The value $\phi = 0$ is achieved when the load is perfectly balanced across the C channels, in which case the expression in the right hand side of (6.1) becomes equal to $\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} t_{ij}}{C}$. The value $\phi = C - 1$ corresponds to the worst case scenario where one channel carries all the traffic; in this case, the right hand side of (6.1) becomes equal to $\sum_{i=1}^{N}\sum_{j=1}^{N} t_{ij}$.

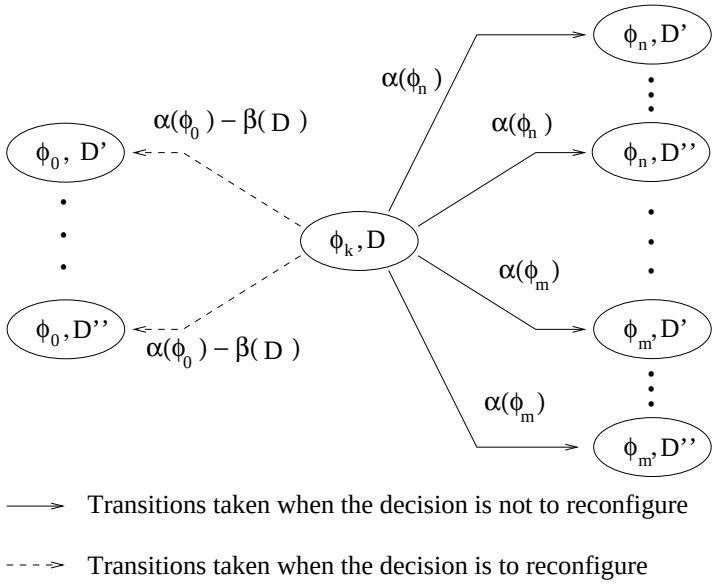


Figure 6.4: Transitions and rewards out of state (ϕ_k, D) of process $\mathcal{M}'$ under the two decision alternatives (*Note:* the labels along the transitions represent rewards, *not* transition probabilities)

a transition into state $(\phi_l, D'), l \geq 0$, it acquires an immediate expected reward which is equal to $\alpha(\phi_l)$. In addition, if (ϕ_l, D') is a balanced state (i.e., if $l = 0$), a reconfiguration cost equal to $\beta(D)$ is incurred.

The transitions out of state (ϕ_k, D) and the corresponding rewards are illustrated in Figure 6.4. If the decision of the policy is not to reconfigure, then the process will take one of the transitions indicated by the solid arrows in Figure 6.4. Since the network does not incur any reconfiguration cost, the immediate reward acquired is a function of the new DLB in the new state. If, on the other hand, the decision is to reconfigure, the transition out of state (ϕ_k, D) will always take the network to a balanced state with a DLB equal to ϕ_0. These transitions are shown in dotted lines in Figure 6.4. A reconfiguration cost is incurred in this case, making the immediate reward equal to $\alpha(\phi_0) - \beta(D)$.

The new process $\mathcal{M}'$ is a discrete-space, discrete-time Markov process with rewards and two alternatives per state, and we can use the policy-iteration algorithm [11] to obtain an optimal policy off-line and cache its decisions. The optimal policy decisions can then be applied to a real network environment in the following way (refer also to Figure 6.3). Consider a network with traffic matrix $\mathbf{T}$ operating under WLA $\mathcal{R}$. Let $\mathbf{T}'$ be the new traffic matrix and $\mathcal{R}'$ be the WLA constructed by algorithm *GLPT* [3] with $\mathcal{R}$ and $\mathbf{T}'$ as inputs.

Let also $D = \mathcal{D}(\mathcal{R}, \mathcal{R}')$ be the number of receivers that need to be retuned to obtain WLA $\mathcal{R}'$ from WLA $\mathcal{R}$. To determine whether the network should reconfigure to the new WLA $\mathcal{R}'$, let $\phi(\mathcal{R}, \mathbf{T})$ be the current DLB for the network, and suppose that $\phi(\mathcal{R}, \mathbf{T})$ falls within the k-th interval, $0 \leq k \leq K$. By definition of the Markov process $\mathcal{M}'$, the current network state is modeled by state (ϕ_k, D) of this process. If, under the optimal policy, the decision associated with this state is to reconfigure, then the network must make a transition to the new WLA $\mathcal{R}'$; otherwise, the network will continue operating under the current WLA $\mathcal{R}$.

We note that the discrete-space Markov process (ϕ_k, D) is an approximation of the continuous-space process (ϕ, D), since, as discussed above, in general the DLB ϕ is a real number between 0 and $C - 1$. We also note that as the number of intervals $K \to \infty$, the discrete-state process approaches the continuous-state one. Therefore, we expect that as the number of intervals K increases, the accuracy of the approximation will also increase and the decisions of the optimal policy obtained through the process (ϕ_k, D) will "converge". This issue will be discussed in more detail in the next section, where numerical results to be presented will show that the decisions of the optimal policy "converge" for relatively small values of K. This is an important observation since the size of the state space of Markov process $\mathcal{M}'$ increases exponentially with K. By using a relatively small value for K we can keep the state space of the process to a reasonable size, making it possible to apply the policy-iteration algorithm [11].

6.3 Numerical Results

In this section we demonstrate the properties of the optimal policies obtained by applying the policy-iteration algorithm [11] to the Markov decision process developed in the previous section. We also show how the optimal policy is affected by the choice of reward and cost functions, and we compare the long-term reward acquired by the network when the optimal policy is employed to the reward acquired by other policies. All the results presented in this section are for the approximate Markov process $\mathcal{M}'$ with state space (ϕ_k, D).

In this study, we consider a *near-neighbor* traffic model . More specifically, we make the assumption that, if the network currently operates with a DLB equal to ϕ_k and no reconfiguration occurs, the next transition is more likely to take the network to the same DLB or its two nearest neighbors ϕ_{k-1} and ϕ_{k+1}, than to a DLB further away from ϕ_k.

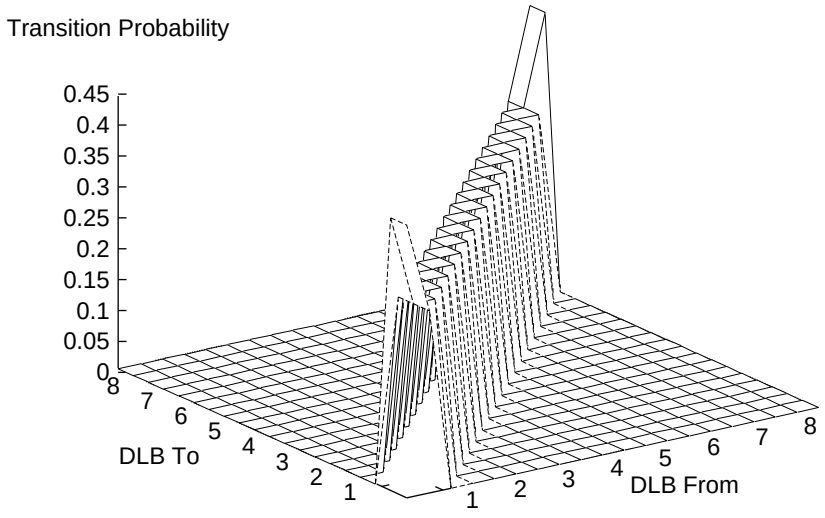


Figure 6.5: Near-neighbor model

Specifically, we assume that

$$P[\phi_l \mid \phi_k] \quad = \quad \begin{cases} 0.3, & k = 1, \cdots, K-1,\ l = k-1, k, k+1 \\ 0.1/(K-2), & k = 1, \ldots, K-1,\ l \neq k-1, k, k+1 \\ 0.45, & k = 0, l = 1 \text{ or } k = K, l = K-1 \\ 0.1/(K-2), & k = 0, l = 2, \cdots, K \text{ or } k = K, l = 0, \cdots, K-2 \end{cases} \tag{6.4}$$

This traffic model is illustrated in Figure 6.5 which plots the conditional probability $P[\phi_k \mid \phi_l]$ that the next DLB will be ϕ_l given that the current DLB is ϕ_k, for $K = 20$ intervals. The near-neighbor model captures the behavior of networks in which the traffic matrix $\mathbf{T}$ changes slowly over time and abrupt changes in the traffic pattern have a low probability of occurring.

Given the probabilities in (6.4), we let the transition probability, *when no reconfiguration occurs*, from state (ϕ_k, D) to state (ϕ_l, D') be equal to:

$$P[(\phi_l, D') \mid (\phi_k, D)] \quad = \quad P[\phi_l \mid \phi_k]\, p_{D'} \tag{6.5}$$

where $p_{D'}$ is the probability that D' retunings will be required in the next reconfiguration. The probabilities p_D were measured experimentally, and we also observed that the proba-

bility that random variable D takes on a particular value is independent of the DLB ϕ_k, thus the expression (6.5).

We note that we need to obtain two different transition probabilities out of each state [11], one for each of the two possible options: the do-not-reconfigure option and the reconfigure option. The above discussion explains how to obtain the transition probability matrix for the do-not-reconfigure option. The transition probability matrix for the reconfigure option is easy to determine since we know that regardless of the value ϕ_k of the current state, the next state will always be a balanced state, i.e., its DLB will be ϕ_0. The individual transition probabilities from a state (ϕ_k, D) to a state (ϕ_l, D') are then obtained by making the same assumption that all values of D have an equal probability of occurring. Therefore, the transition probabilities under the reconfigure option are:

$$P[(\phi_l, D') \mid (\phi_k, D)] \quad = \quad \begin{cases} p_{D'}, & l = 0 \\ 0, & \text{otherwise} \end{cases} \tag{6.6}$$

6.3.1 Convergence of the Optimal Policy

Let us first consider the following reward and cost functions

$$\alpha[(\phi_k, D)] \quad = \quad \frac{A}{1 + \phi_k} \ , \qquad \beta(D) \quad = \quad BD \tag{6.7}$$

discussed in Section 6.2.2, where A and B are weights assigned to the rewards and costs. We apply Howard's algorithm [11] to a network with $N = 20$ nodes and $C = 5$ wavelengths with a near-neighbor traffic model similar to the one shown in Figure 6.5. Our objective is to study the effect that the number of intervals K in the range $[0, C-1]$ of possible values of DLB ϕ has on the decisions of the optimal policy. As we mentioned in Section 6.2.3, we expect the decisions of the optimal policy to "converge" as $K \to \infty$. More formally, let φ be a real number such that $0 \leq \varphi \leq C-1$, and let k_K be the interval in which φ falls when the total number of intervals is K. Also let $d^{(K)}[(\phi_{k_K}, D)]$ be the decision of the optimal policy for state (ϕ_{k_K}, D) of Markov process $\mathcal{M}'$ when the number of intervals is K. We will say that the decisions of the optimal policy converge if

$$\lim_{K \to \infty} \quad d^{(K)}[(\phi_{k_K}, D)] \quad = \quad d[(\varphi, D)] \quad \forall \ \varphi, \ D \tag{6.8}$$

In Figures 6.6 to 6.8 we plot the decisions of the optimal policy for the 20-node, 5-wavelength network with a near-neighbor traffic model, and for three different values of

K; the weights used in the functions (6.7) were set to $A = 30$ and $B = 1$. Figure 6.6 corresponds to the optimal policy for $K = 20$ intervals, while in Figures 6.7 and 6.8 we increase K to 30 and 40, respectively. The histograms shown in Figures 6.6 to 6.8, as well as in other figures in this section, should be interpreted as follows. In each figure, the x axis represents the DLB ϕ_k (with a number of intervals equal to the corresponding value of K), while the y axis represents the possible values of D. The vertical bar at a particular DLB value ϕ_k has a height equal to D_k^{thr} such that:

$$d^{(K)}[(\phi_k, D)] \quad = \quad \begin{cases} \text{reconfigure}, & D \leq D_k^{thr} \\ \text{do not reconfigure}, & D > D_k^{thr} \end{cases} \tag{6.9}$$

In other words, *for each value of* ϕ_k, there exists a *retuning threshold* value D_k^{thr} such that the decision is to reconfigure when the number of receivers to be retuned is less than D_k^{thr}, and not to reconfigure if it is greater than D_k^{thr}. Since the optimal policy had similar behavior for all the different reward and cost functions we considered, its decisions will be plotted as a histogram similar to those in Figures 6.6 to 6.8 [3].

As we can see in Figures 6.6 to 6.8, the decisions of the optimal policy do converge (in the sense of expression (6.8)) as K increases. For instance, let us consider a DLB of 1, which falls in the fourth interval when $K = 20$ (in Figure 6.6), the sixth interval when $K = 30$ (in Figure 6.7), and the seventh interval when $K = 40$ (in Figure 6.8). In all three cases, the retuning threshold is equal to 9 for these intervals, therefore, the decisions of the optimal policy for the three values of K are the same. On the other hand, for a DLB of 2, the retuning threshold is 14 in Figure 6.6, but it drops to 13 in Figure 6.7, same as in Figure 6.8. In other words, for a DLB of 2, the decisions of the optimal policy are different when $K = 20$ than when $K = 30$ or 40 (in the former case, the decision is to reconfigure as long as the number of retunings is at most 14, while in the latter the decision is to reconfigure only when the number of retunings is at most 13). But the important observation is that the policy decisions do not change when the number K of intervals increases from 30 to 40, indicating convergence. In fact, there are no changes in the optimal policy for values of K greater than 40 (not shown here). We have observed similar behavior for a wide range of

[3] That the optimal policy was found to be a threshold policy (with a possibly different retuning threshold) for each value of ϕ_k, can be explained by the fact that we only consider cost functions that are non-decreasing functions of random variable D. As a result, if the decision of the optimal policy for a state (ϕ_k, D_1) is not to reconfigure, intuitively one expects the decision for state (ϕ_k, D_2), where $D_2 > D_1$ to also be not to reconfigure since the reconfiguration cost $\beta(D_2)$ for the latter state would be at least as large as the reconfiguration cost $\beta(D_1)$ for the former.

values for the weights A and B, for different network sizes, as well as for other reward and cost functions. These results indicate that a relatively small number of intervals is sufficient for obtaining an optimal policy.

Another important observation from Figures 6.6 to 6.8 is that the retuning threshold increases with the DLB values. This behavior can be explained by noting that, because of the near-neighbor distribution (refer to Figure 6.5), when the network operates at states with high DLB values, it will tend to remain at states with high DLB values. Since the reward is inversely proportional to the DLB value, the network incurs small rewards by making transitions between such states. Therefore, the optimal policy is such that the network decides to reconfigure even when there is a large number of receivers to be retuned. By doing so, the network pays a high cost, which, however, is offset by the fact that the network makes a transition to the balanced state with a low DLB, reaping a high reward. On the other hand, when the network is at states with low DLB, it also tends to remain at such states where it obtains high rewards. Therefore, the network is less inclined to incur a high reconfiguration cost, and the retuning threshold for these states is lower.

6.3.2 The Effect of Reward and Cost Functions

In Figures 6.9 to 6.11 we apply Howard's algorithm to a network with $N = 100$ nodes and $C = 10$ wavelengths, operating under a near-neighbor model similar to the one shown in Figure 6.5. For this network we used $K = 20$ intervals, and we varied the weights A and B in the reward and cost functions in (6.7) to study their effect on the optimal policy. Specifically, we let $B = 1$ and we varied A from 20 (in Figure 6.9) to 35 (in Figure 6.10) to 50 (in Figure 6.11). We first observe that the optimal policy is again a threshold policy for each value ϕ_k of the DLB. However, as A increases, we see that the retuning threshold associated with each DLB value also increases. This behavior of the optimal policy is in agreement with intuition since, by increasing A we increase the reward obtained by taking the network to a balanced state relative to the cost of reconfiguration, making reconfigurations more attractive. Similarly, if we keep A constant and we increase B (a case not shown here), reconfiguring the network becomes less desirable, and thus the retuning threshold associated with each DLB value decreases. Overall, in our study we have found that one can obtain a wide variety of policies by varying the values of weights A and B.

We now proceed to study the effect of different reward and cost functions. Let us

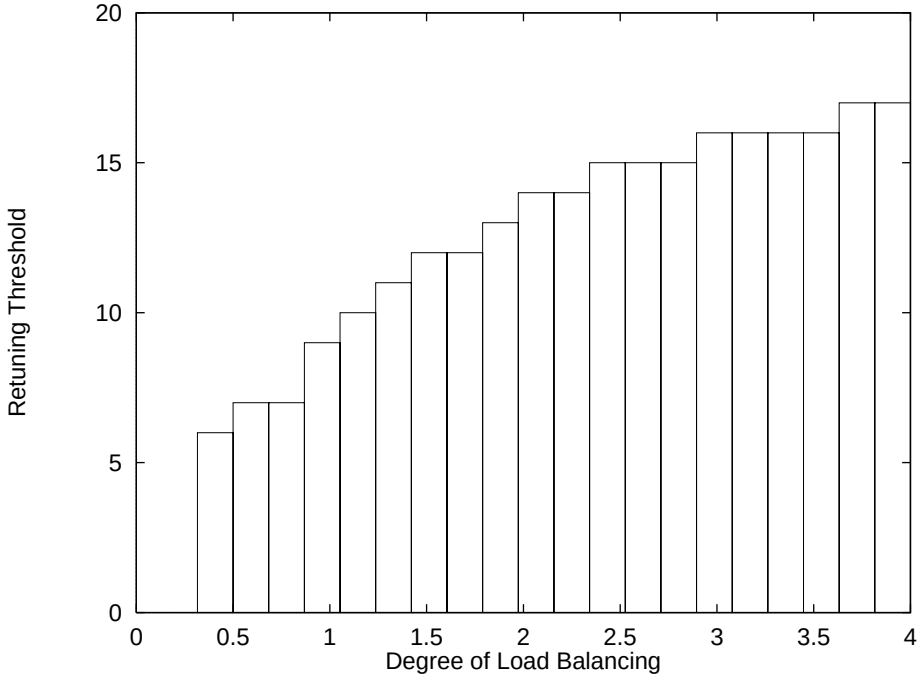


Figure 6.6: Optimal policy decisions for $N = 20$, $C = 5$, $K = 20$, $A = 30$, $B = 1$

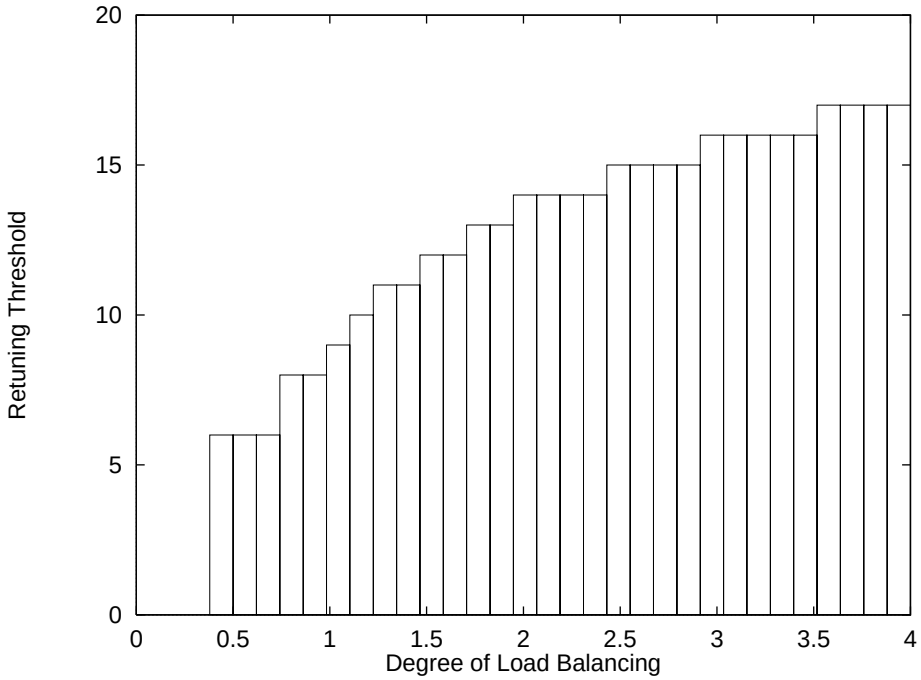


Figure 6.7: Optimal policy decisions for $N = 20$, $C = 5$, $K = 30$, $A = 30$, $B = 1$

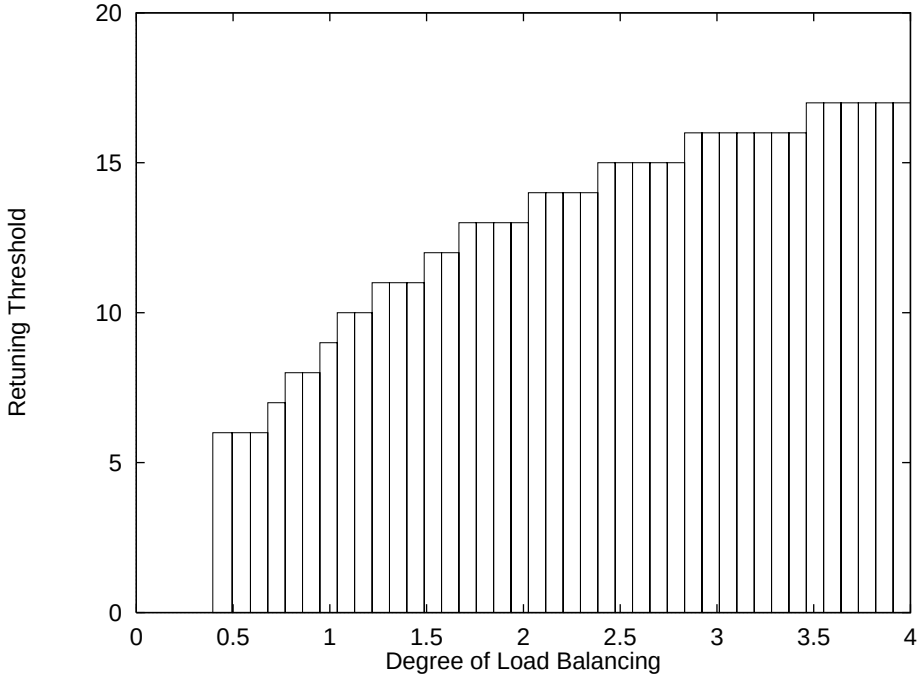


Figure 6.8: Optimal policy decisions for $N = 20$, $C = 5$, $K = 40$, $A = 30$, $B = 1$

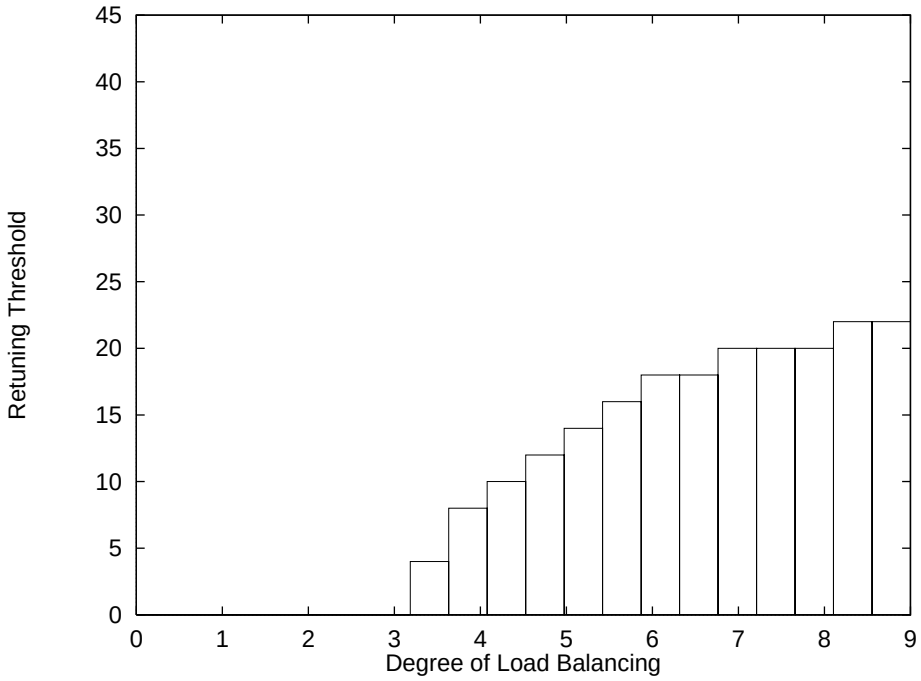


Figure 6.9: Optimal policy decisions for $N = 100$, $C = 10$, $K = 20$, $A = 20$, $B = 1$

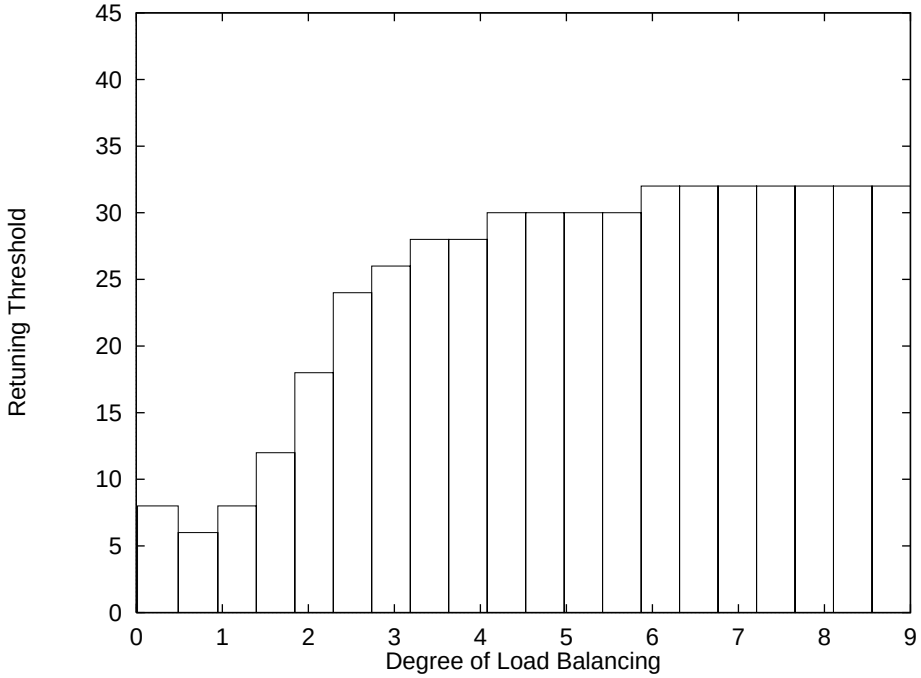


Figure 6.10: Optimal policy decisions for $N = 100$, $C = 10$, $K = 20$, $A = 35$, $B = 1$

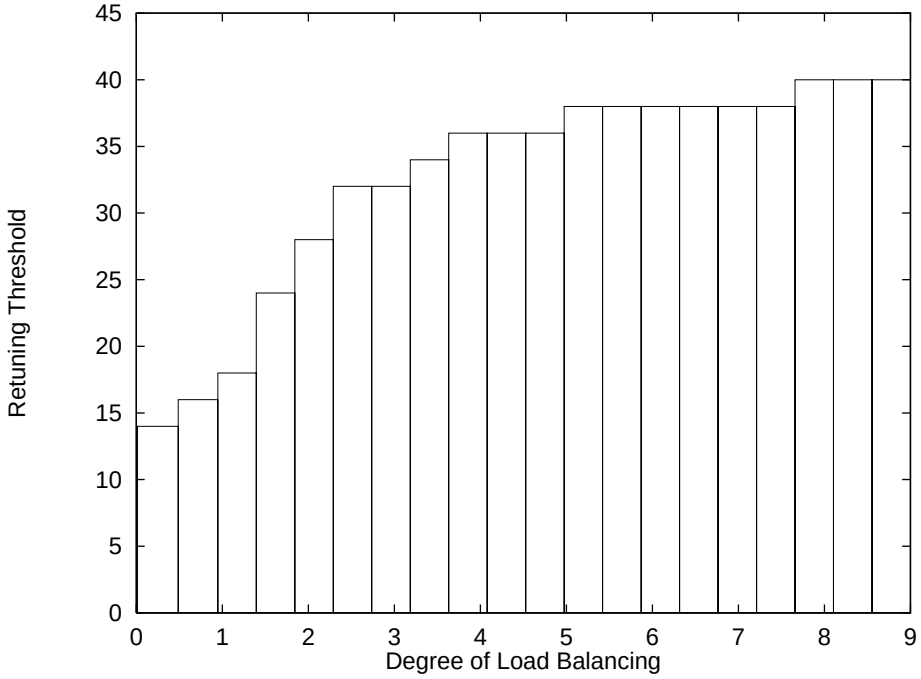


Figure 6.11: Optimal policy decisions for $N = 100$, $C = 10$, $K = 20$, $A = 50$, $B = 1$

first consider a new cost function $\beta(D)$ as plotted in Figure 6.13, while the reward function is as in (6.7) with $A = 50$. This cost function is similar to the the one shown in Figure 6.2 with $\beta_{min} = 1$, $\beta_{max} = 2$, and $D_{max} = 30$. As we discussed in Section 6.2.2, maximizing the expected reward in this case will minimize the probability that more than D_{max} receivers will have to be retuned during reconfiguration. In Figure 6.13 we show the decisions of the optimal policy for a network with $N = 100$, $C = 10$, and a near-neighbor traffic model, when $K = 20$. As we can see, the retuning threshold never exceeds the value $D_{max} = 30$, therefore, the network will never reconfigure when the number of retunings is greater than 30, as expected.

We also obtained the optimal policy for the same network as above, but with the reward function shown in Figure 6.14, and a cost function $\beta(D) = BD$, with $B = 1$. This reward function is similar to that in Figure 6.1, with $\alpha_{min} = -30$, $\alpha_{max} = 80$, and $\phi_{max} = 4.5$. As the reader may recall, this reward function can be used in order to minimize the probability that the network will operate with a DLB greater than ϕ_{max}. The resulting policy is shown in Figure 6.15, where we can see that the retuning threshold is 100 for DLB values greater than 4.5. Since the maximum number of receivers that will ever need to be retuned is $N - C = 90$ [3], a retuning threshold equal to 100 means that the network will always reconfigure when the DLB becomes greater than 4.5. Thus, although the network is not prohibited from entering a state with a DLB value greater than 4.5, once doing so, in the very next transition the network will reconfigure and will enter the balanced state. Subsequently, because of the nature of the near-neighbor traffic model, the network will tend to stay at states with low DLB values. In effect, therefore, the probability that the network will be operating at states with DLB values greater than 4.5 is very small when the reward function in Figure 6.14 is used.

6.3.3 Comparison to Threshold Policies

In this section we compare the optimal policy against three classes of threshold-based policies:

- **DLB-threshold policies.** There exists a threshold DLB value ϕ_{max} such that, if the system is about to make a transition into a state $(\phi_k, D), \phi_k > \phi_{max}$, then the network will reconfigure and make a transition to a state with DLB ϕ_0, regardless of the reconfiguration cost involved. Otherwise, no reconfiguration occurs. This class

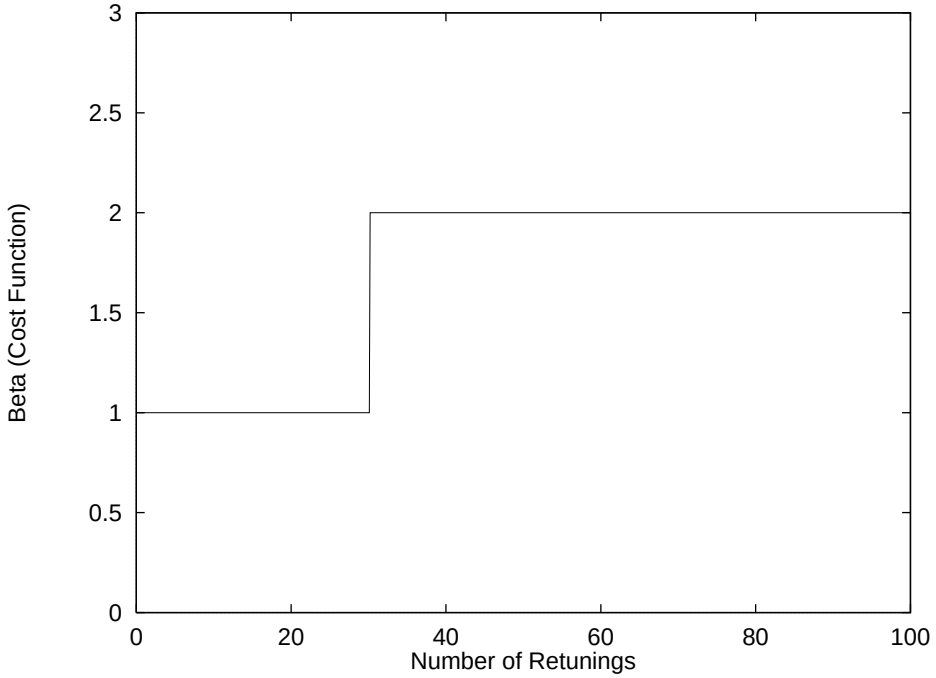


Figure 6.12: Cost function $\beta(D)$ used for the policy shown in Figure 6.13

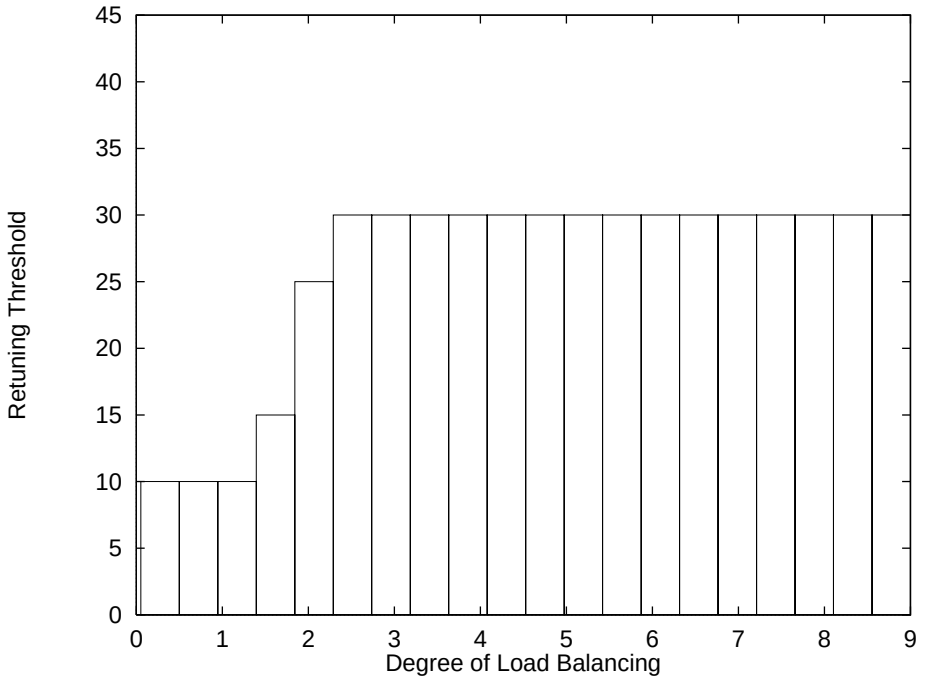


Figure 6.13: Optimal policy decisions for $N = 100$, $C = 10$, $K = 20$, $A = 50$

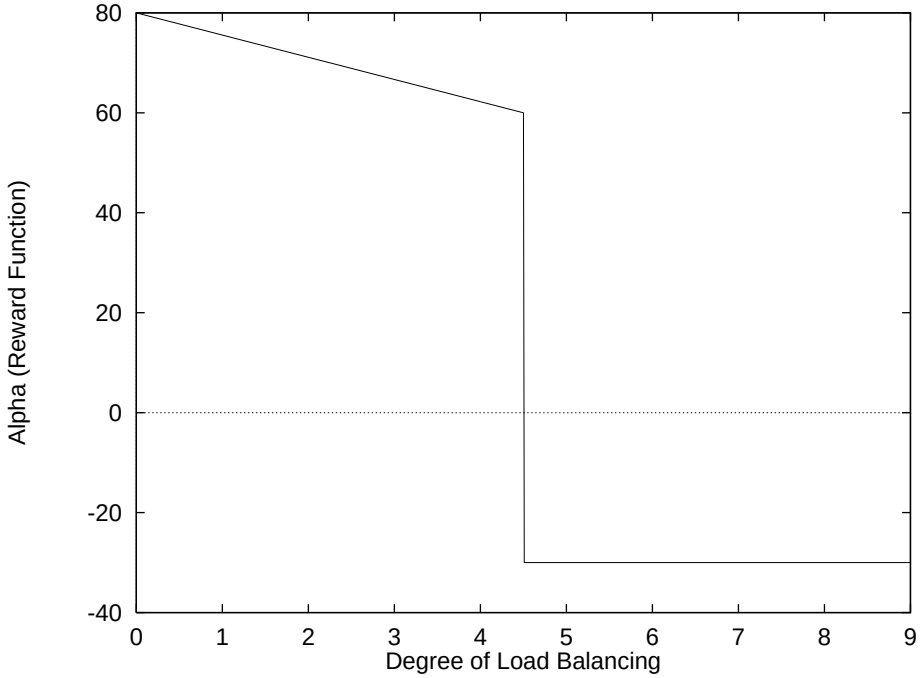


Figure 6.14: Reward function $\alpha(\phi)$ used for the policy shown in Figure 6.15

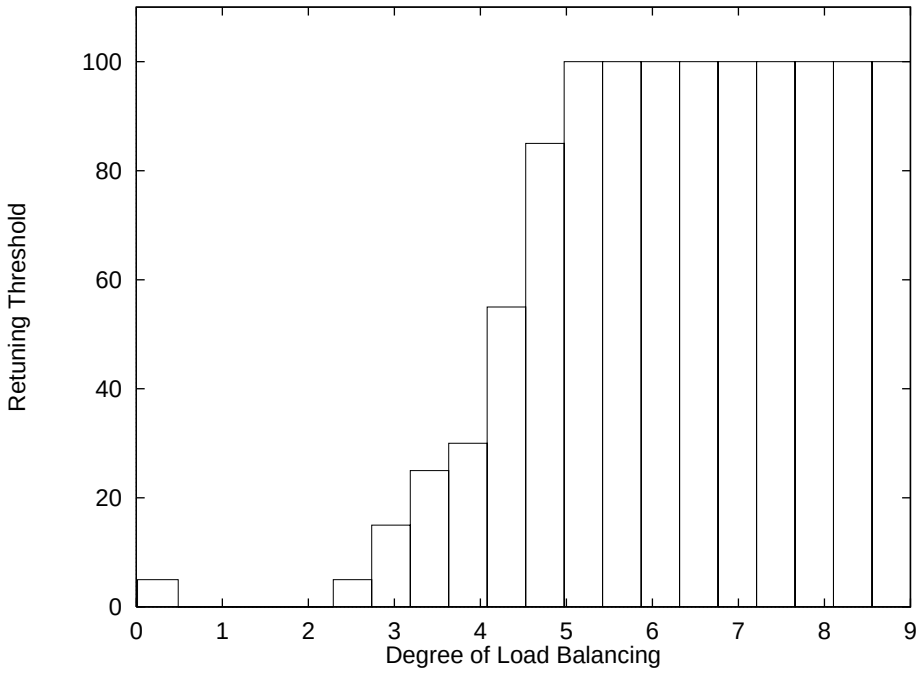


Figure 6.15: Optimal policy decisions for $N = 100$, $C = 10$, $K = 20$, $B = 1$

of policies is not concerned with the reconfiguration cost incurred. Instead it ensures that the traffic carrying capacity of the network will never fall below the value $\gamma_{min} = C/(1+\phi_{max})$.

- **Retuning-threshold policies.** This class of policies is in a sense a "dual" of the previous one, in that decisions are based solely on the number of retunings involved in the reconfiguration, not on the DLB. Specifically, if the network is about to make a transition, then the network will reconfigure only if the number D of receivers that must be retuned is less than or equal to a threshold D_{max}. If $D > D_{max}$, no reconfiguration takes place. This class of policies ensures that the portion of the network that becomes unavailable due to reconfiguration never exceeds D_{max}.

- **Two-threshold policies**. This class of policies attempts to combine the objectives of the two classes of policies above. Specifically, there are two thresholds, ϕ_{max} and D_{max}. If the system is about to make a transition into a state (ϕ_k, D), then the network will reconfigure if $\phi_k > \phi_{max}$. Otherwise, if $\phi_k \leq \phi_{max}$, the network will reconfigure if the number D of receivers that must be retuned is less than or equal to D_{max}, and it will not reconfigure if $D > D_{max}$. We note that if we let $D_{max} = N - C$ (i.e., the maximum number of receiver that will ever need to be retuned [3]), these policies reduce to the class of DLB-threshold policies. Similarly, if we let $\phi_{max} = C - 1$ (i.e., the DLB threshold is equal to the maximum DLB value), these policies become simple retuning threshold policies. Therefore, the two-threshold policies are the most general class of policies, and include the DLB-threshold and retuning-threshold policies as special cases.

The DLB-threshold and the general two-threshold policies above define Markov processes which are *outside* the class of Markovian Decision Processes considered in Section 6.2. In a Markovian Decision Process, there are several alternatives per state, but once an alternative has been selected for a state, then transitions from this state are always governed by the chosen alternative (refer also to Figure 6.4). In a DLB-threshold policy, on the other hand, the alternative selected does not depend on the *current* state, but rather on the *next* state. Therefore, the system may select different alternatives when at a particular state, depending on what the next state is [4]. Similarly for the two-threshold policies. Since

[4] If the next state is one with a DLB less than the threshold, the alternative selected is not to reconfigure, otherwise the alternative selected is to reconfigure.

Howard's algorithm [11] is optimal only within the class of Markovian Decision Processes, it is possible that these threshold policies obtain rewards higher than the optimal policy determined by the algorithm. Retuning-threshold policies, however, are such that there is a unique alternative per state, so we expect them to perform no better than the optimal policy [5].

All the results presented in this section are for a network with $N = 100$ nodes, $C = 10$ wavelengths, a near-neighbor traffic model, and $K = 20$ intervals. The reward and cost functions considered are those in expression (6.7). In Figure 6.16 we compare the optimal policy obtained by Howard's algorithm [11] to a number of retuning-threshold policies. The figure plots the average long-term reward acquired by each of the policies against the retuning threshold D_{max}. The horizontal line corresponds to the reward of the optimal policy, which, clearly, is independent of the retuning threshold. Each point of the second line in the figure corresponds to the reward of a retuning-threshold policy with the stated threshold value. As we can see, retuning-threshold policies obtain a reward which is significantly less than that of the optimal policy, as expected. Furthermore, the reward of retuning-threshold policies varies depending on the actual threshold used. Since the best threshold depends on system parameters such as the traffic patterns and the reward and cost functions and the associated weights, it is impossible to know the best threshold to use unless one experiments with a large number of threshold values.

In Figure 6.17 we compare the optimal policy to a DLB-threshold policy and a number of two-threshold policies. For these results, we used $A = 50$ and $B = 1$ as the values for the weights in the reward and cost functions, respectively, in (6.7). This time we plot the reward of each policy against the DLB threshold value; similar to Figure 6.16, the optimal policy is independent of the DLB threshold, resulting in a horizontal line in Figure 6.17. We also plot the reward of DLB-threshold policies with varying DLB thresholds, and of a family of two-threshold policies. Each of the three plots of two-threshold policies corresponds to a different retuning threshold (namely, $D_{max} = 40, 32,$ and 24) and varying DLB thresholds. Also, recall that the DLB-threshold policy is equivalent to a two-threshold policy with a retuning threshold equal to $N - C = 90$.

[5] As we have seen, the optimal policies are in fact threshold policies with a different retuning threshold for each DLB value. Therefore the optimal policy will in general perform better than a reconfiguration policy with the same threshold for all DLB values.

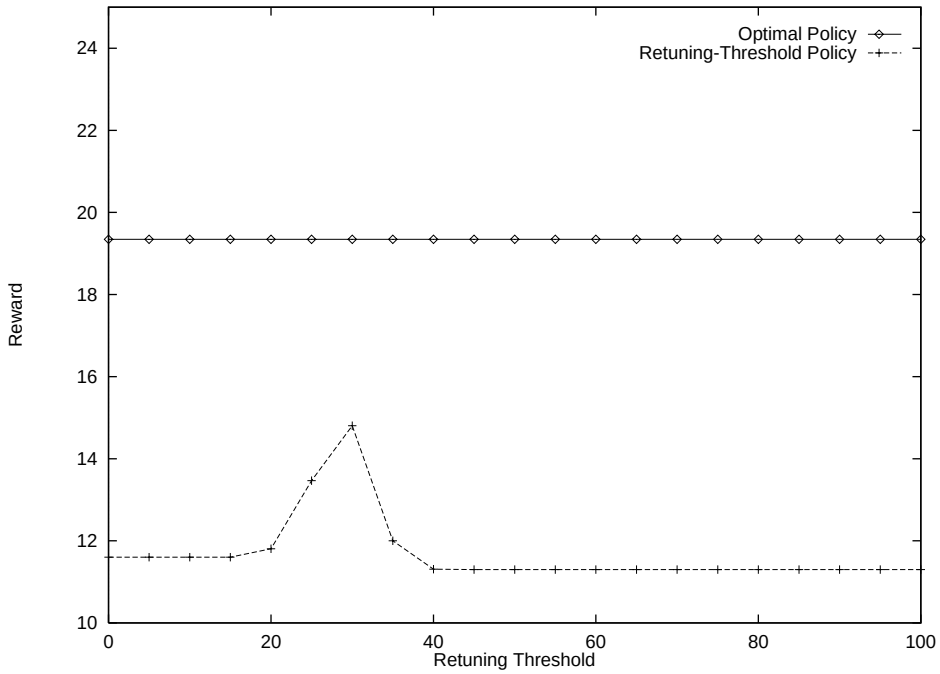


Figure 6.16: Policy comparison, $N = 100$, $C = 10$, $K = 20$, $A = 50$, $B = 1$

The most interesting observation from Figure 6.17 is that, for certain values of the DLB-threshold, the DLB-threshold policy and the two-threshold policy with retuning threshold $D_{max} = 40$ achieve a higher reward than the optimal policy obtained through Howard's algorithm. This result is possible because, as we discussed earlier, the class of two-threshold policies is more general than the class of policies for which Howard's algorithm is optimal. On the other hand, we note that the reward of the DLB-threshold policy depends strongly on the DLB threshold used, and that the reward of the two-threshold policies depends on the values of both thresholds. Although within a certain range of these values the threshold policies perform better than the optimal policy, the latter outperforms the former for most threshold values. Therefore, threshold selection is of crucial importance for the threshold policies, but searching through the threshold space can be expensive. The optimal policy, however, guarantees a high overall reward and is also simpler to implement since the network does not need to *look ahead* to the next state to decide whether or not to reconfigure.

Figure 6.18 is similar to Figure 6.17 in that we again compare the optimal policy against a DLB-threshold and two-threshold policies. For these experiments, however, we

have used $A = 20$ and $B = 1$ in the reward and cost functions, respectively, of (6.7). As we can see, the reward of the optimal policy is strictly higher than that of threshold policies across all possible threshold values. These results demonstrate that DLB- or two-threshold policies do not always perform better than the optimal policy, and their performance depends on the system parameters and/or the reward and cost functions. Furthermore, it is not possible to know ahead of time under what circumstances the threshold policies will achieve a high reward. Equally important, if the network's operating parameters change, threshold selection must be performed anew, since, for instance, the DLB threshold that maximizes the reward of the DLB-threshold policy in Figure 6.17 results in very poor performance in Figure 6.18, and vice versa.

Overall, the results presented in this section demonstrate that the optimal policy obtained through Howard's algorithm can successfully balance the two conflicting objectives, namely the DLB and the number of retunings, and always achieves a high reward across the whole range of the network's operating parameters. We have also shown that, by appropriately selecting the reward and cost functions, the optimal policy can be tailored to specific requirements set by the network designer. On the other hand, pure threshold policies, although they can sometimes achieve high reward, are less flexible, and they introduce an additional degree of complexity, namely, the problem of threshold selection.

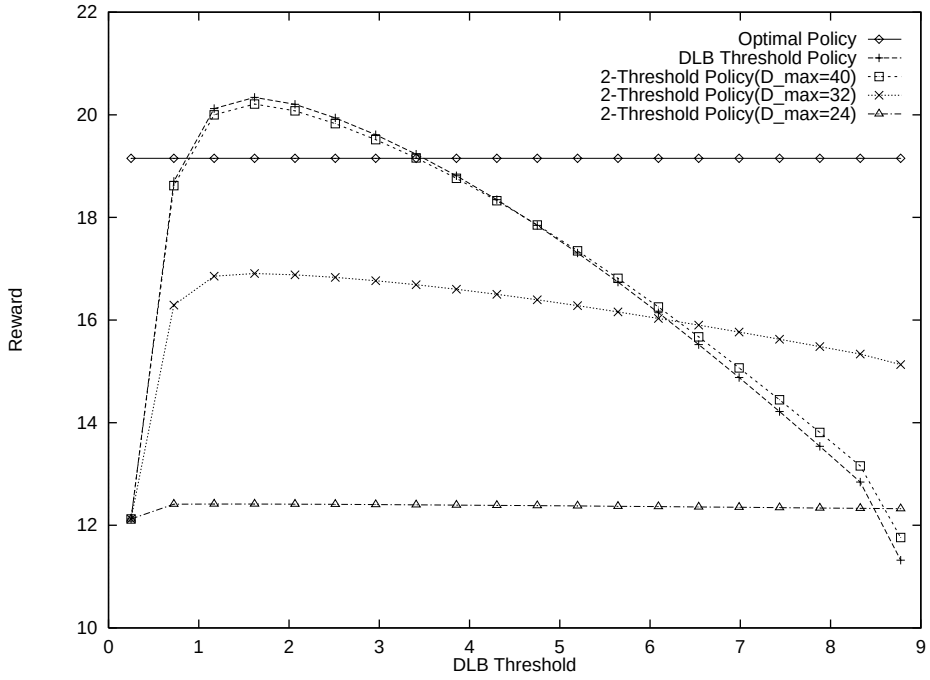


Figure 6.17: Policy comparison, $N = 100$, $C = 10$, $K = 20$, $A = 50$, $B = 1$

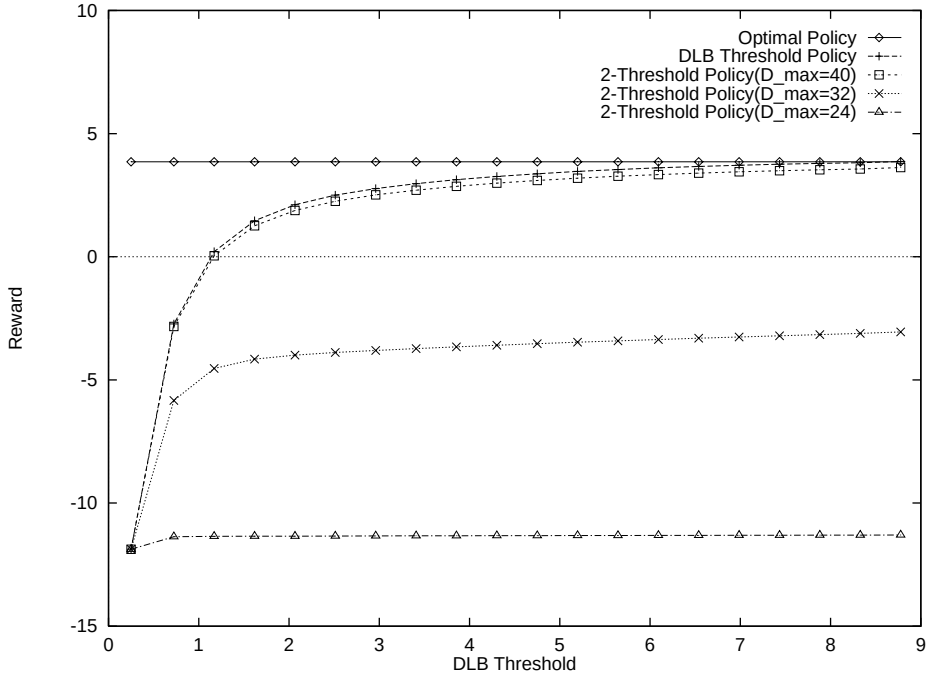


Figure 6.18: Policy comparison, $N = 100$, $C = 10$, $K = 20$, $A = 20$, $B = 1$

Chapter 7

Retuning Strategies

In the previous chapters we have been able to answer two of the most important questions in reconfiguring a broadcast WDM network with fast-tunable transmitters and slowly-tunable receivers. Those were how to determine a balanced wavelength assignment and how to make a decision when best to reconfigure the network from the standpoint of packet losses vs. performance (e.g. average cell delay or throughput). In this chapter we study the final problem associated with reconfigurable WDM networks: how to transfer the network from the original WLA $\mathcal{R}$ to the new balanced WLA $\mathcal{R}'$. In order to do that we have created a realistic simulation of a broadcast WDM network that applies the solutions of the previous two chapters and uses them to compare the various strategies of retuning the network from one WLA to another. In the following sections we will first introduce our retuning strategies, then we will take a closer look at the simulation and finally examine some numerical results acquired through the simulation.

7.1 Retuning Strategies

Once the decision to reconfigure the network is made according to the optimal policy described in the previous chapter and the new WLA $\mathcal{R}'$ is calculated, what remains is the procedure that transfers the network from the original WLA $\mathcal{R}$ to the new WLA $\mathcal{R}'$. We call this procedure a *retuning strategy*. It is clear that in order to complete the reconfiguration a number of receivers equal to $D(\mathcal{R}, \mathcal{R}')$ must be retuned. The retuning strategy determines when and in what sequence these receivers are retuned. Once all of these receivers complete retuning, we may say that the network has completed a reconfiguration

Parameterized non-overlapping greedy retuning strategy

Input: Parameter $1 \leq P \leq N$, set $S(\mathcal{R}, \mathcal{R}')$ of receivers that need to be retuned.

1. Whenever a batch of receivers has finished retuning, until set S is exhausted
2. Pick at most P receivers from set S with the smallest number of cells queued for them.
3. Remove these receivers from set S.
4. Remove cells addressed to these receivers from all node buffers.
5. Schedule this batch of receivers for retuning at the beginning of the next schedule.
6. end.

Figure 7.1: Description of our retuning strategy

and may proceed to function normally until a new reconfiguration is required.

The set of retuning strategies is rather large, however all of them must follow one important rule in order to be compatible with network operations. This rule states that each receiver can be scheduled to begin retuning *only at the HiPeR-l schedule boundary*. It is necessary to ensure the correct functioning of HiPeR-l which presumes that all of the receivers that are available in the beginning of the scheduling cycle, will remain available until the current schedule is exhausted. In other words, HiPeR-l is not capable of generating transmission schedules in which a particular receiver is available only part of the time. Scheduling retunings to begin only at the schedule boundaries ensures that a receiver is either available for scheduling by HiPeR-l with no restrictions or not available for scheduling at all (while retuning) and therefore is ignored by HiPeR-l in the current schedule.

Besides this important rule, a retuning strategy can schedule the receivers to retune in a number of different ways. Two limiting cases are retuning all of the receivers at the same time or scheduling them to retune one by one at schedule boundaries. In between lies a large number of strategies that can schedule the retunings in batches of various sizes. The strategies may also vary in whether they allow the different batches to overlap or not, since a single batch may take longer than one schedule to retune. If a strategy requires that one batch must finish retuning before the new one can be scheduled, it is called *non-overlapping* otherwise it is an *overlapping* strategy.

In our work we have chosen to study a limited class of parameterized non-overlapping

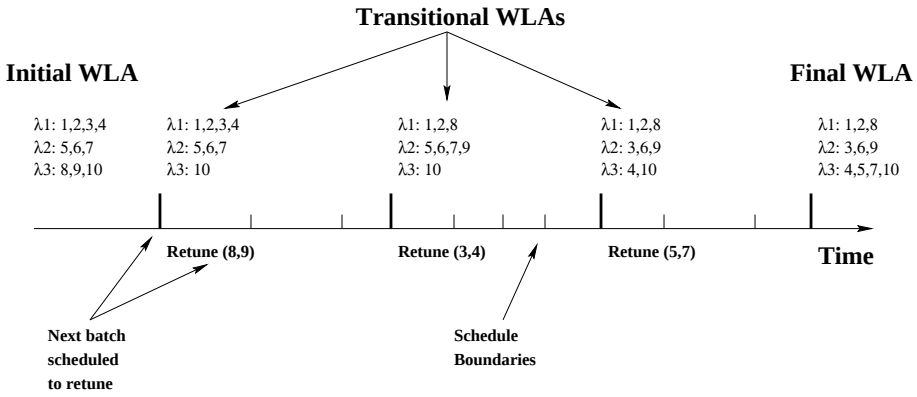


Figure 7.2: Actions of a non-overlapping reconfiguration strategy with $P = 2$ for a network with C=3, N=10

greedy retuning strategies. Each strategy has a parameter P and can be described by pseudocode in Figure 7.1.

The actions of such a strategy are depicted in Figure 7.2 by using an example of a network with 3 wavelengths (channels) and 10 nodes. Note that by using this strategy, or indeed, any strategy that allows to retune less receivers in a single batch than the total number of receivers that require retuning, the network's WLA goes through a series of transformations that begins with the initial WLA $\mathcal{R}$ and ends with the new balanced WLA $\mathcal{R}'$. The WLAs in between those two are called *transitional* WLAs, because they do not contain all of the receivers in the network - just those that aren't currently retuning. These transitional WLAs serve as intermediate points in the process of network reconfiguration. They are necessary in order to keep the packet losses during the reconfiguration to a minimum, since they allow packets to be transmitted even during the reconfiguration phase, albeit not to all of the receivers and not under the best possible conditions, since the transitional WLAs are unbalanced. The alternative solution of allowing only the initial and the final WLAs to be used by the network would mean completely shutting down the network during the reconfiguration phase and would be inflexible in addition to causing unnecessary packet losses.

In the case of our parameterized strategies, by varying the value of parameter P we hope to determine whether there is a possible tradeoff between the number of nodes that can be scheduled to retune at the same time and the associated packet losses in the reconfiguration phase. In order to find this out we have created a simulation of a WDM

network, which will be introduced in the next section.

7.2 Simulating a Broadcast WDM Network

7.2.1 Components of the Simulation

For the implementation of the network simulation we have chosen an event-based type simulation (as opposed to slotted) in order to improve execution time. The simulation is guided by a single event queue, in which events are ordered according to their arrival time. An arrival of a particular event to the top of the queue causes the simulation to take actions necessary to process this event. Some examples of the possible event types are:

- An arrival of a cell into a node buffer.
- A departure of a cell from a node buffer.
- HiPeR-l schedule boundary.

Besides the event queue, the simulation consists of components that closely model their real-world counterparts such as cells, sources, buffers, nodes, WLAs etc. The structure of a node object is reflected in Figure 7.3. A node owns a set of sources, which generate cells according to preset parameters such as PCR (Peak Cell Rate), SCR (Sustained Cell Rate), MBS (Maximum Burst Size) and others. Each cell has a destination node, which determines which of the buffers it will be sent to. A node has C buffers (same number as the number of channels in the network). The arriving cells are buffered by their outgoing channel, which is derived from the current WLA, based on the destination node id. The buffers are simple FIFO queues of a predetermined size, such that if a cell arrives to a buffer which is already full to capacity, it is considered lost. The cells are taken out of the buffer based on cell departure events from the event queue.

The cell departure events are generated from the HiPeR-l transmission schedules, which, in turn, are calculated based on node buffer occupancy at the moment of the arrival of the schedule boundary event (for more on HiPeR-l scheduling see [26]) The schedule boundary events are offset from each other by the length of the current schedule. The frequency of schedule boundary events (schedule length) depends on the current traffic pattern and the current WLA. The next section will describe the traffic pattern used in our simulations.

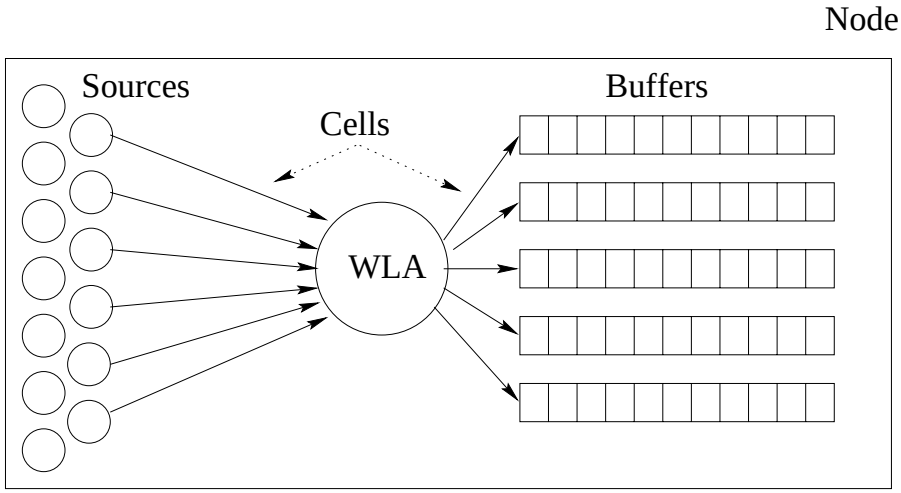


Figure 7.3: Node representation in the Network Simulation

7.2.2 Network Traffic Pattern

For the purposes of this study we decided on a WDM network with $C = 5$ channels and $N = 70$ nodes. Each channel was modeled as an OC-48 (2.48Gb/s) link. To create a traffic pattern with high variations in the Degree of Load Balancing we have adopted a client-server model of communications. The nodes were separated into two groups - a small group of servers and the rest of them clients.

Each node has two types of sources - one that creates the background traffic in the network by communicating with other client nodes at low speeds, and the other type that models the communication of this node with its assigned server and is distinguished by a significantly higher cell rate, than the background sources. Each client node has multiple background sources and a single client-server source. Server nodes only have background sources. The source, regardless of its type has a parameter that determines its call length (the time while the source is actively generating cells) and a call waiting time (the time the source spends quiet between calls).

By adjusting these parameters we have been able to create a traffic pattern with significant variations in the DLB. In order to sustain these variations even after the network was reconfigured once, the pattern was made to vary in time by dynamically changing the assignment of server nodes to clients after the end of every call.

Figure 7.4 shows how the length of the HiPeR-l schedules varied through time with our traffic pattern without network reconfigurations. We can see that due to the

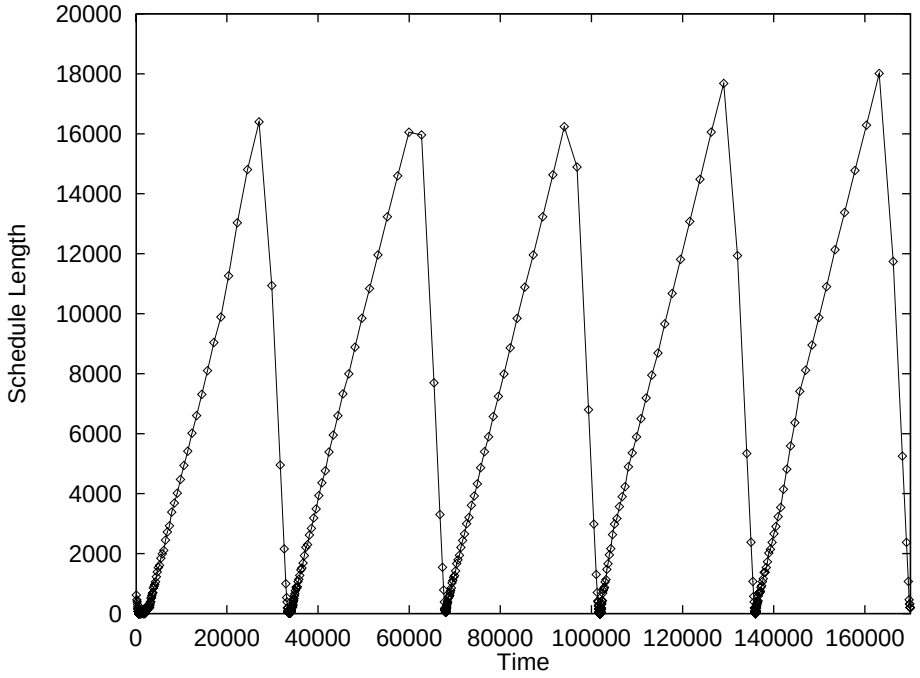


Figure 7.4: Variations in Schedule Length in a client-server traffic pattern.

highly unbalanced traffic load, the schedules tended to grow quite large (up to 18,000) cells, until all the client-server calls ended and the load was shifted to the new servers.

To see what kind of values of DLB we could create in the network we built a 20-bin distribution of DLB values similar to the near-neighbor distribution described in the previous chapter (see Figure 6.5). The difference was that the DLB varied between 0.2 and 1.2 in our model. This conditional distribution $P[\phi_k \mid \phi_l]$ can be seen in Figure 7.5.

As can be seen, abrupt traffic changes in our model caused the 'bunching' of the DLB values in two regions - the tuned region below 0.2 and the second region with values ≈ 1.

After applying the Howard's algorithm according to the method developed in the previous chapter we were able to calculate an optimal policy as shown in Figure 7.6. This policy was used in the comparison of retuning strategies performed later.

It should be mentioned that the Degree of Load Balancing was measured not on a per-schedule basis as one might think. We have discovered that under comparatively low cell loads, when the schedule length is small, it is often possible to observe high values of DLB, if measured on per-schedule basis, even though the traffic load may be well balanced. It is

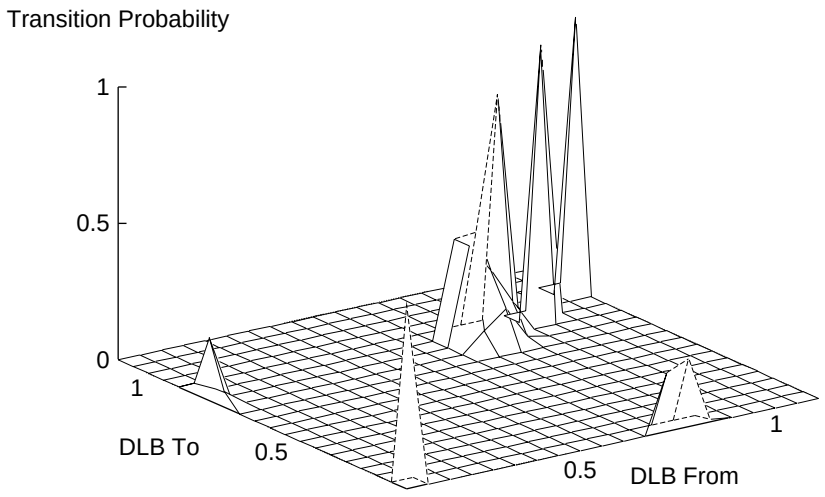


Figure 7.5: Conditional distribution of DLB values in the client-server model.

caused by the inherent burstiness of some of the sources, which are active for a short period of time, supplying a lot of cells into the buffer, even though their sustained rate may be low and the cell load, as mentioned before, may be well balanced in the long term. To avoid the unnecessary reconfigurations, caused by this effect, we accumulated the traffic data in a traffic matrix T at 2 millisecond intervals independent of the HiPeR-l schedules. This data was then used to determine the DLB during each of the intervals, and the conditional distribution was built on this data.

The introduction of this interval, of course, meant that no two reconfigurations could be spaced closer than 2 ms from each other, but in our model the traffic changed no more often than every 20 ms, which allowed us to accumulate ample data on the changes in the degree of load balancing in the network. We believe something similar will have to be done if our work were to be applied to a real-life WDM network.

The next section will show some numerical results acquired through the use of the simulation.

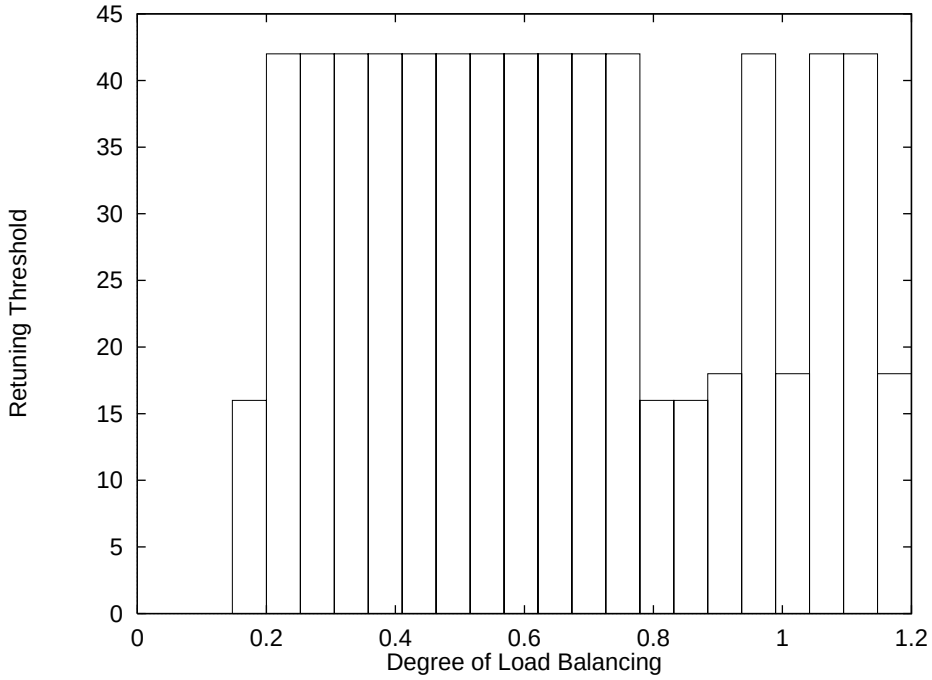


Figure 7.6: Optimal reconfiguration policy calculated for the client-server traffic pattern

7.3 Numerical Results

By using the network simulation described above we were able to perform two types of numerical comparisons. First we compared WDM networks with and without the dynamic reconfiguration capability on the same traffic pattern in order to verify that using this capability indeed resulted in improved network utilization. Second, we compared various retuning strategies in order to determine the effect of several of the important parameters on the network performance. The following two sections will present the results.

7.3.1 Benefits of Dynamic Network Reconfigurations

In this first experiment we compare the performance of the network with and without dynamic reconfigurations. When the reconfigurations are allowed, they are done according to the optimal policy depicted in Figure 7.6 and a DLB-threshold policy, as described in Section 6.3.3 with a threshold value of 0.8. The retuning strategy used for both of these policies used $P = N$ or, in other words, allowed to retune all of the receivers at once. The receiver tuning latency was equivalent to 1000 cell transmission slots while

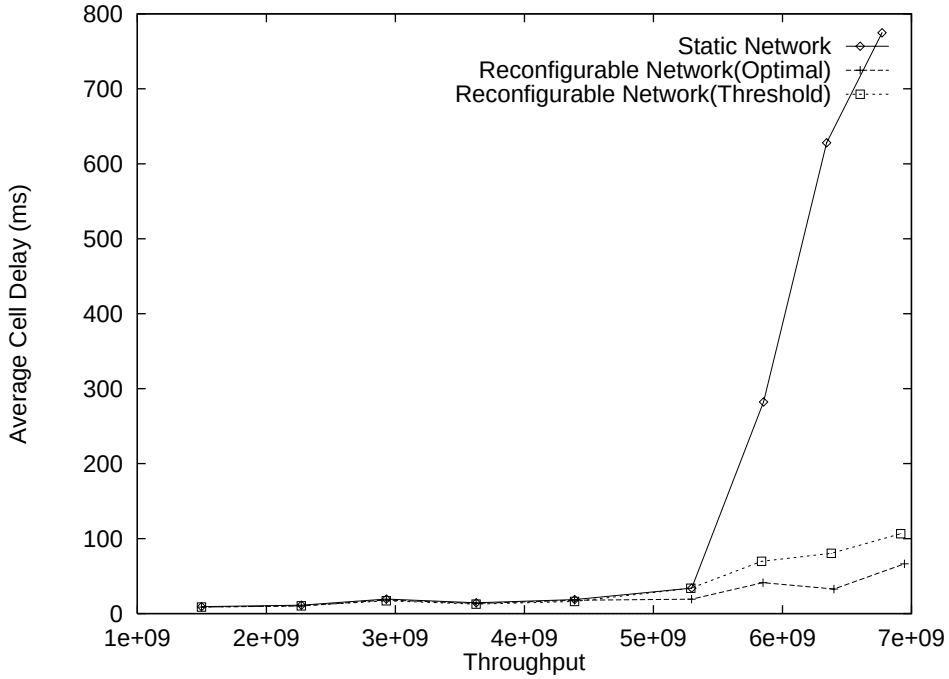


Figure 7.7: Average cell delay in networks with and without reconfiguration capabilities

the channel buffers (see Figure 7.3) in nodes were fixed at 200 cells/channel.

We compare the performance of the network based on three parameters: the average cell delay, the percentage of cells lost and the maximum buffer size used by the nodes. In all cases the network is allowed to process 10^7 cells before terminating. This only included cells delivered to their destinations, which means that the cells lost due to buffer overflow or reconfigurations were not counted. We compare these parameters of interest under increasing traffic loads, peaking at about 7Gb/s, of the $5 * OC - 48 = 12.4$Gb/s total available network bandwidth. As can be seen from Figure 7.7 the networks with a dynamic reconfiguration capability clearly outperform the network without such capability in terms of average cell delay. As the network load increases, the delay in the static network soars to nearly 800 microseconds, while in the dynamically reconfigurable networks it grows much slower and remains under 100 microseconds. Also, the reconfigurable network using an optimal reconfiguration policy achieves lower average cell delays than the same network using a DLB-threshold policy.

It also turns out that despite the inevitable cell losses associated with the reconfigurations in the dynamic networks, these losses are still smaller then those of the static

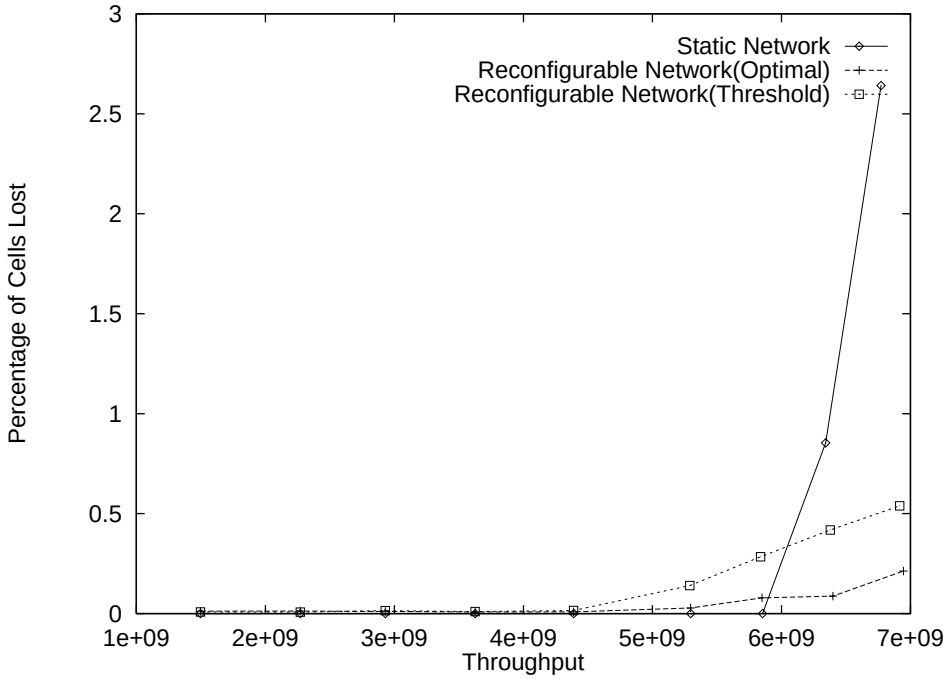


Figure 7.8: Percentage of cells lost in networks with and without reconfiguration capabilities

network, incurred due to the unbalanced wavelength assignment. The static network under maximum load loses as much as 2.5 percent of all cells, which is unacceptable. At the same time both the dynamic networks lose less than 1/2 of a percent of cells, and, in fact, the dynamic network using the optimal reconfiguration policy loses only 1/4 of a percent, which is one order of magnitude improvement over a static network. This of course, under relatively high network traffic loads. When the network load is low, the static network incurs no losses, because its buffers are large enough to accommodate the imperfections in the traffic load, while the dynamically reconfigurable networks have some losses due to the reconfigurations.

An additional benefit of using the reconfiguration capability seems to be that under high traffic loads, the network with a reconfiguration capability requires smaller buffers, than the static network, as shown in Figure 7.9. As we can see, the static network starts losing cells due to buffer overflow at high traffic loads, so its buffer requirements peak at 200 cells, which was the preset limit, and so does the dynamic network with a DLB-threshold policy. At the same time, the reconfigurable network using an optimal reconfiguration policy never loses cells due to buffer overflow, since it never reaches the maximum allowable buffer size.

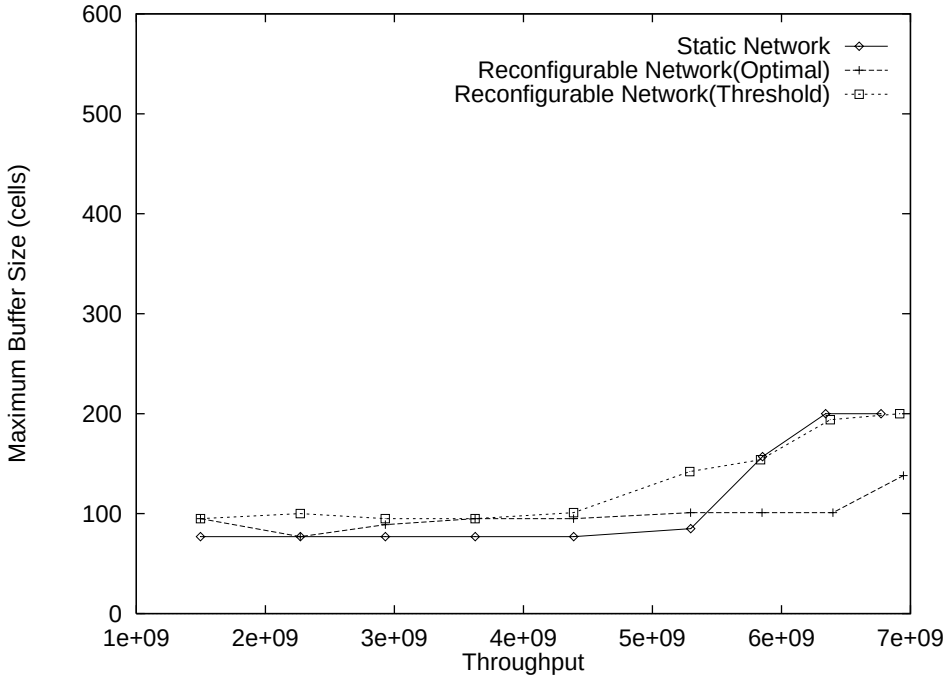


Figure 7.9: Maximum buffer sizes in networks with and without reconfiguration capabilities

All of the cell losses this network sustains are entirely due to the reconfigurations.

The conclusion we can draw from this data is that under moderate to high traffic load, the performance of a WDM network would significantly benefit from having a dynamic reconfiguration capability. This validates our original intent to improve the network performance, by using a novel RTT-STR architecture that can adapt itself to the traffic conditions. It also shows the importance of using an optimal reconfiguration policy in such a network, since the reconfigurable network with a DLB-threshold type policy did not perform as well as the same network with an optimal policy.

7.3.2 Comparison of Retuning Strategies

In this section we presume that the optimal reconfiguration policy is in use and we attempt to compare the various strategies of the family we described in section 7.1. We compare them according to the same three parameters of interest, used in the previous comparison: average cell delay, percentage of cells lost and maximum buffer size. The average cell delay and the maximum buffer size tell us about the network performance, since the lower delay results in higher throughput and the need for smaller buffers, while

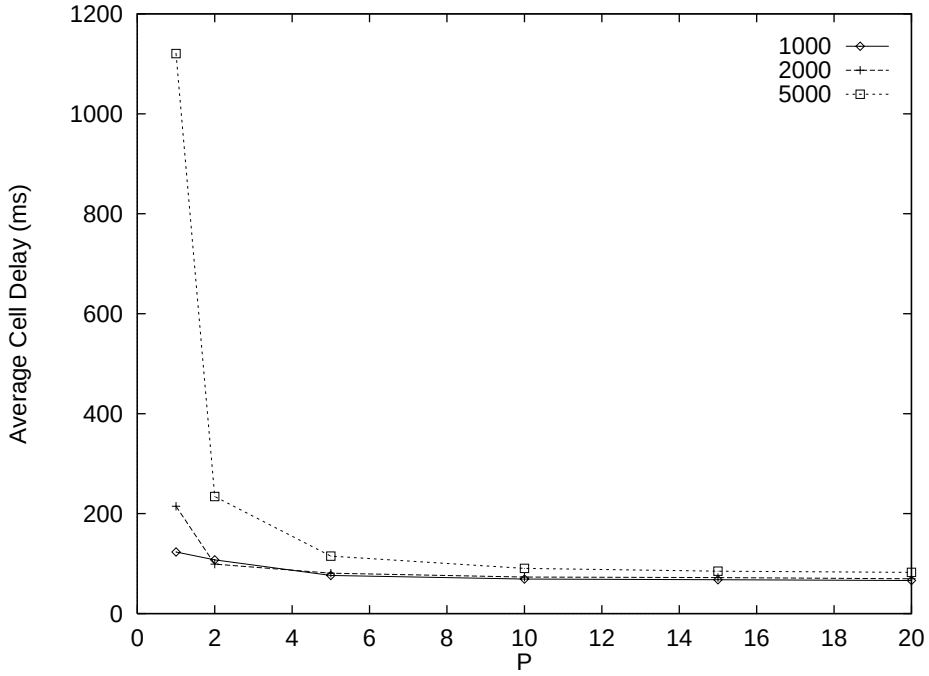


Figure 7.10: Average cell delay comparison of retuning strategies for various values of receiver latency

the cell losses tell us what price the performance improvement comes at. We used the offered load of 7Gb/s, equivalent to the maximum load, used in the previous section.

One important parameter that directly impacts the performance of a retuning strategy is the receiver tuning latency described in section 4.1. In Figures 7.10 and 7.11 we show the dependence of strategies on this parameter, by plotting the behavior of the retuning strategies with various values of the receiver tuning latency (equivalent to 1000, 2000 and 5000 cell transmission slots). Each line connecting the data points refers to a family of strategies using the same value of the receiver tuning latency, which is shown in the legend in units of cells. The average cell delay and cell losses are shown along the Y axes, while we plot the value of the parameter P (see section 7.1) against the X axes.

We can see from these plots that increasing the receiver tuning latency increases the average cell delay, cell losses and the maximum buffer size. It can be explained by the fact that the longer a single receiver takes to retune, the longer a batch of receivers will take to retune, which will, therefore, result in more cells lost, while each batch is retuning. It also takes the network longer to reconfigure to an optimal WLA, which negatively affects the average cell delays. It should be noted that in this experiment the buffer size was

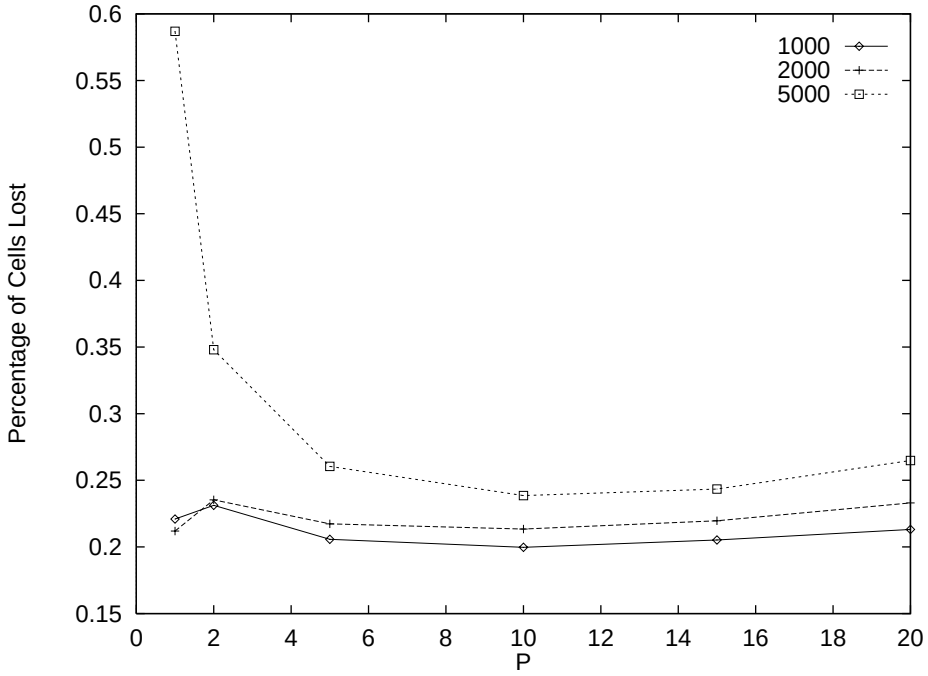


Figure 7.11: Cell loss comparison of retuning strategies for various values of receiver latency

not limited, as we wished to see the actual maximum values it would reach under varying conditions. This means that any cell losses sustained by the network were entirely due to reconfigurations, and none due to buffer overflow.

From these plots we can also see that the two extreme strategies of retuning the receivers one at a time or all at once, tend to do worse than the 'in between' strategies when some significant fraction of the receivers is allowed to retune in a single batch. We can attempt to explain this by studying each case separately. When all of the receivers are allowed to retune at the same time, the network takes the shortest possible time to reconfigure to an optimal WLA, however during that time none of the receivers are available so, even though the average cell delay is the best in this case, due to the short reconfiguration period, the cell losses are somewhat high. When the network is only allowed to retune the receivers one at a time, the reconfiguration period takes the longest among all such strategies, so both the cell delays and the losses suffer due to the fact that the WLA being used by the network is not optimal for longer periods of time, than with other strategies. The strategies that allowed to retune several receivers at once have done well in terms of cell delay and best in terms of cell loss. In a real network these two parameters will have

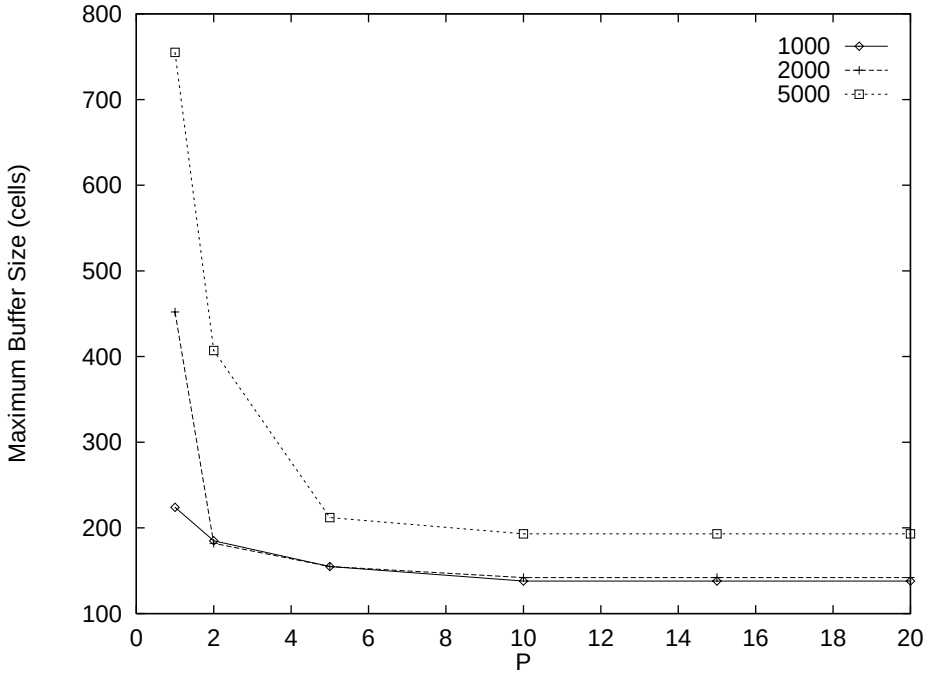


Figure 7.12: Maximum buffer sizes of retuning strategies for various values of receiver latency

to weighed against each other in order to determine which strategy is best suited for the environment.

7.4 Conclusion

In this chapter we have been able to compare by using a simulation, the performance of static and dynamically reconfigurable WDM networks. We also compared various retuning strategies and their effect on the overall network performance.

It is clear from these results that the dynamically reconfigurable networks will outperform their static counterparts in the environments with shifting and uneven traffic loads. The comparison of the parameterized retuning strategies showed that the 'in-between' strategies do better than the extreme strategies but the choice of the particular value for parameter P should be determined by the network quality of service requirements and the parameters of the hardware elements involved.

Chapter 8

Conclusions and Future Work

We believe that the main contribution of this research has been as much in identifying the problems associated with the dynamic reconfiguration capability in broadcast WDM networks and developing a systematic way of approaching them, as in finding the solutions to them. First we proposed a novel network architecture we call RTT-STR (Rapidly Tunable Transmitters-Slowly Tunable Receivers) which is capable of dynamically adapting to the changing traffic conditions, in order to improve the overall network performance. Then we identified and solved three of the main problems that arise when one attempts to implement the dynamic reconfiguration capability in a WDM network. The problems and our solutions to them are presented in Table 8.1.

These solutions allowed us to create a realistic simulation of the type of the network under study, which, in turn, has verified the validity of our approach as a whole, by proving that a dynamically reconfigurable network does perform better than a static one. In addition, the simulation has also provided us with valuable numerical data concerning the network performance under various conditions.

8.1 Future Work

This work was centered around RTT-STR (rapidly-tunable transmitters, slowly tunable receivers) architecture. It should be possible to extend this research to the dual STT-RTR (Slowly Tunable Transmitters-Rapidly Tunable Receivers) architecture, and quite possibly many of the conclusions of this work could be directly applied to such an architecture.

Problem	Solution
Decide when it is best to reconfigure from the point of view of network resources.	An optimal reconfiguration policy based on representing a network state as a Markovian Decision Process.
Determine a balanced assignment of the receivers to channels.	An algorithm called GLPT, which calculates the new balanced WLA based on the previous one, thus minimizing the number of receivers that require retuning.
Perform the reconfiguration with minimal losses in short amount of time.	A class of reconfiguration strategies, governed by a single parameter P, which allow to adjust the tradeoff between the cell losses incurred during network reconfiguration against the long-term improvement in network performance.

Table 8.1: Problems and solutions of dynamic reconfiguration in WDM networks.

Our simulation of a reconfigurable WDM network used a fairly unsophisticated traffic model, so an interesting area of study would be in finding other models that better capture the salient features of the traffic likely to be observed in a multi-channel WDM network. A better understanding of the traffic behavior in such a network would lead to the refinement of our solutions and, perhaps, identify other potential problems that might be faced in implementing such a network.

In our research we have studied one class of retuning strategies and we have been able to show that a trade-off exists in this class between the several parameters involved (e.g. depending on the value of parameter P, we might observe various average cell delays and cell losses). Although the cell losses due to reconfigurations with most retuning strategies amounted to a fraction of a percent of overall traffic, other classes of retuning strategies could be studied to determine whether our greedy strategy can be improved upon.

Bibliography

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, N.J., 1993.

[2] M. Azizoglu, R. A. Barry, and A. Mokhtar. Impact of tuning delay on the performance of bandwidth-limited optical broadcast networks with uniform traffic. *IEEE Journal on Selected Areas in Communications*, 14(5):935–944, June 1996.

[3] I. Baldine and G. N. Rouskas. Dynamic load balancing in broadcast WDM networks with tuning latencies. In *Proceedings of INFOCOM '98*, pages 78–85. IEEE, March 1998.

[4] M. S. Borella and B. Mukherjee. Efficient scheduling of nonuniform packet traffic in a WDM/TDM local lightwave network with arbitrary transceiver tuning latencies. *IEEE Journal on Selected Areas in Communications*, 14(5):923–934, June 1996.

[5] E. Coffman, M. R. Garey, and D. S. Johnson. An application of bin-packing to multiprocessor scheduling. *SIAM Journal of Computing*, 7:1–17, Feb 1978.

[6] F.Bernabei, L.Gratta, and M.Listanti. Throughout analysis of multihop shufflenets in a hot spot traffic scenario: Impact of routing strategies. *Computer Networks and ISDN Sytems*, April 1996.

[7] F.Glover. Tabu search: A tutorial. *Interfaces*, 20, July-August 1990.

[8] M. R. Garey, R. L. Graham, and D. S. Johnson. Performance guarantees for scheduling algorithms. *Operations Research*, 26:3–21, Jan 1978.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman and Co., New York, 1979.

[10] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17(2), Mar 1969.

[11] R. A. Howard. *Dynamic Programming and Markov Processes*. M.I.T. Press, Cambridge, 1960.

[12] J.A.Bannister, L.Fratta, and M.Gerla. Topological design of the wavelength-division optical network. *Proceedings of INFOCOM '90, San Francisco California*, June 1990.

[13] J-F. P. Labourdette, F. W. Hart, and A. S. Acampora. Branch-exchange sequences for reconfiguration of lightwave networks. *IEEE Transactions on Communications*, 42(10):2822–2832, October 1994.

[14] Jean Francois P. Labourdette. Traffic optimization and reconfiguration management of multiwavelength multihop broadcast lightwave networks. *To appear in Computer Networks and ISDN Systems*, 1998.

[15] A. Muir and J. J. Garcia-Luna-Aceves. Distributed queue packet scheduling algorithms for WDM-based networks. In *Proceedings of INFOCOM '96*, pages 938–945. IEEE, March 1996.

[16] Z. Ortiz, G. N. Rouskas, and H. G. Perros. Scheduling of multicast traffic in tunable-receiver WDM networks with non-negligible tuning latencies. In *Proceedings of SIGCOMM '97*, pages 301–310. ACM, September 1997.

[17] H. G. Perros and K. M. Elsayed. Call admission control schemes: A review. *IEEE Communications Magazine*, 34(11):82–91, 1996.

[18] G. R. Pieris and G. H. Sasaki. Scheduling transmissions in WDM broadcast-and-select networks. *IEEE/ACM Transactions on Networking*, 2(2):105–110, April 1994.

[19] Rajiv Ramaswami and Kumar N.Sivarajan. *Optical Networks: A Practical Perspective*. Morgan/Kaufmann, San Francisco, CA, 1998.

[20] G. N. Rouskas and V. Sivaraman. Packet scheduling in broadcast WDM networks with arbitrary transceiver tuning latencies. *IEEE/ACM Transactions on Networking*, 5(3):359–370, June 1997.

[21] S.Banerjee and B.Mukherjee. The photonic ring: Algorithms for optimized node arrangements. *Journal of Fiber and Integrated Optics*, December 1993.

[22] S.Banerjee and B.Mukherjee. Algorithms for optimized node arrangements in shufflenet based multihop lightwave networks. *Journal of High Speed Networks*, April 1995.

[23] S.Banerjee, B.Mukherjee, and D.Sarkar. Heuristic algorithms for constructing optimized structures of linear multihop lightwave networks. *IEEE Transactions on Communications*, April 1994.

[24] Michael S.Borella, Jason P.Jue, Dhritiman Banerjee, Byrav Ramamurthy, and Biswanath Mukherjee. Optical components for wdm lightwave networks. *Proceedings of the IEEE*, 85(8):1274–1306, August 1997.

[25] K. Sivalingam and P. Dowd. A multi-level WDM access protocol for an opticall interconnected multi-processor system. *IEEE/OSA Journal of Lightwave Technology*, 13(11):2152–2167, November 1995.

[26] V. Sivaraman and G. N. Rouskas. HiPeR-ℓ: A High Performance Reservation protocol with ℓook-ahead for broadcast WDM networks. In *Proceedings of INFOCOM '97*, pages 1272–1279. IEEE, April 1997.

Appendix A

Proof of Lemma 5.1.1

Proof. We will construct an instance of the CA problem for which the difference in the number of retunings under the identity and optimal permutations is equal to $N-1$. Consider a network with $C \geq 3$ and $N > C$. Let the initial wavelength assignment $\mathcal{R} = \{R_c\}$ be any arbitrary assignment such that $\mid R_c \mid \geq 2$ and let $j \in R_C$. Let the new partition $\mathcal{S}' = \{S'_c\}$ be such that

$$S'_c = \begin{cases} R_2 \cup \{j\}, & c = 1 \\ R_{c+1}, & c = 2, \cdots, C-2 \\ R_C - \{j\}, & c = C-1 \\ R_1, & c = C \end{cases} \tag{A.1}$$

It is straightforward to verify that the identity permutation requires that all receivers retune to new wavelengths (N retunings), while the optimal permutation $(C, 1, 2, 3, \cdots, C-1)$ requires only one retuning, that of receiver j from wavelength λ_C to wavelength λ_2. □

Appendix B

Proof of Lemma 5.1.2

Proof. We will first prove that no more than $N - C$ retunings are needed under an optimal solution to the CA problem. We will then show that this is a tight bound by constructing instances of the CA problem that require a number of retunings equal to the upper bound.

Consider a network with N nodes and $C \leq N$ channels. Let m be an integer such that, for any arbitrary instance $(\mathcal{R}(N), \mathcal{S}'(N))$ of the CA problem, there will be at least m (out of N) receivers that do not need to be retuned under the optimal solution (the reason why we express $\mathcal{R}$ and $\mathcal{S}'$ as functions of the number of nodes will become apparent shortly). In other words, if $\mathcal{R}'(N)$ is the optimal new wavelength assignment for instance $(\mathcal{R}(N), \mathcal{S}'(N))$, we have that:

$$\sum_{c=1}^{C} \mid R_c(N) \cap R'_c(N) \mid \;\geq\; m \tag{B.1}$$

Now consider a network with $N' > N$ nodes and C wavelengths. We show by contradiction that, if $(\mathcal{R}(N'), \mathcal{S}'(N'))$ is an arbitrary instance of the CA problem for this network, and $\mathcal{R}'(N')$ is the optimal new wavelength assignment, then we also have:

$$\sum_{c=1}^{C} \mid R_c(N') \cap R'_c(N') \mid \;\geq\; m' \;=\; m, \qquad N' \;>\; N \tag{B.2}$$

Indeed, suppose that $m' < m$, and consider an instance of the CA problem for this network for which the left part of (B.2) holds with equality. Then, by removing from this instance $N' - N$ receivers that need to be retuned [1], we obtain an instance of the CA problem for a

[1] There will be $N' - N$ receivers that need retuning because $N' - N \leq N' - m < N' - m'$.

network with N nodes such that

$$\sum_{c=1}^{C} \mid R_c(N) \cap R'_c(N) \mid \;=\; m' \;<\; m \tag{B.3}$$

But, because of our hypothesis that (B.1) holds, (B.3) is impossible. Therefore, (B.2) must necessarily hold. The result in (5.3) now follows from (B.2) and the fact that, when $C = N$, each channel is assigned exactly one receiver, and, under optimal channel assignment, no receiver needs to be retuned (i.e., when $N = C$, $m = C$ in (B.1)).

A trivial instance for which the upper bound is achieved is for a network with $N = C + 1$ nodes where (a) in the initial assignment all receivers are assigned a unique channel, except i and j who share the same channel, and (b) in the new partition, i is in a subset by itself and j moves to a subset with, say, receiver k. Then, under optimal channel assignment, exactly $N - C = 1$ receiver must be retuned, receiver j, from its original channel to the channel of k. However, even for large N, the number of retunings may be very close to the upper bound $N - C$. Specifically, we now construct an instance of CA that requires exactly $N - C - 1$ retunings. Consider a network with $N = C^2$, and an initial wavelength assignment given by:

$$R_c \;=\; \{(c-1)C + 1, \cdots, cC\} \qquad c = 1, \cdots, C \tag{B.4}$$

The new partition $\mathcal{S}'$ is:

$$S'_c \;=\; \begin{cases} \{c\}, & c = 2, \cdots, C \\ \{1, C+1, \cdots, C^2\}, & c = 1 \end{cases} \tag{B.5}$$

It is straightforward to verify that (a) a permutation is optimal if it assigns S'_1 to any of channels λ_2 through λ_C, and that (b) exactly $C^2 - C - 1 = N - C - 1$ retunings are required under an optimal permutation. □

Appendix C

Howard's Algorithm

C.1 Introduction

Consider a **completely** ergodic N-state Markov process with rewards described by a transition probablity matrix $P = \{p_{ij}\}$ and a reward matrix $R = \{r_{ij}\}$. A quantity, which describes the performance of the process over an infinite horizon in terms of its expected reward is called the *gain* of the process and is expressed as

$$g = \sum_{i=1}^{N} \pi_i q_i \tag{C.1}$$

where π_i are the limiting state probabilities and the quantities q_i are the immediate expected rewards ($q_i = \sum_{j=1}^{N} p_{ij} r_{ij}$) for each state.

In Markovian **decision**processes, instead of a single transition probability vector p_{ij}, all or some of the states may have alternatives described by their own transition probabilities p_{ij}^k (k being the index of the alternative). For such a process a **policy** is an N-vector describing the alternatives to be picked in each of the states. These policies may have different conditions of optimality, and one of the more important ones involves determining the policy with the highest gain. This involves picking the alternatives in each state in such a way that the overall gain of the resulting process is maximized. A straightforward solution would involve studying each possible combination of alternatives, which in most cases would be prohibiteively complex. Howard's Policy-Iteration Algorithm is intended to calculate such an optimal policy in a small number of iterations.

C.2 Steps of the Algorithm

Each iteration of the Howard's algorithm consists of two steps:

1. Value-Determination Operation

2. Policy-Improvement Routine

The Value-Determination Operation involves solving the following system of $N+1$ linear equations:

$$\begin{cases} g + v_i = q_i + \sum_{j=1}^{N} p_{ij} v_j \ , & i = 1, 2, ..., N \\ v_N = 0 \end{cases} \tag{C.2}$$

where g is the gain of the process. These equations are solved for g and the v_i quantities which are called the *relative values* of the process. The state transition probabilities p_{ij} in this system are determined by the policy-vector $d \in R^N$ which resulted from the previous iteration.

The Policy-Improvement Routine finds the alternative k' for each state i such that the quantity

$$q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j \tag{C.3}$$

is maximized. Here the q_i^k quantities are the immediate expected rewards of state i if the alternative k is used. The value of k' which maximizes the above quantity for state i becomes part of the new policy so that q_i^k becomes simply q_i and p_{ij}^k becomes simply p_{ij} and the new iteration may begin.

The iterations stop when the policy calculated at the previous step remains unchanged after a new iteration. The algorithm may begin with either step. If the Value-Determination Operation is used as the first step, some initial policy d must be provided (it can be a policy which, for instance, always chooses the first alternative in every state). If the Policy-Improvement Routine is used as the initial step of the iteration, then the relative values v_i must be chosen (as a vector of all 0's, for instance).

The algorithm has good convergence characteristics, frequently terminating after 5-10 iterations for a 500-state process. For a complete description of the algorithm as well as the proof of correctness see [11].