Dynamic Reconfiguration Policies for WDM Networks *

Ilia Baldine† *George N. Rouskas*‡
†Advanced Networking Research Group, MCNC, RTP, NC 27709
‡Department of Computer Science, North Carolina State University, Raleigh, NC 27695-7534

Abstract

We study the issues arising when considering the problem of reconfiguring broadcast optical networks in response to changes in the traffic patterns. Although the ability to dynamically optimize the network under changing traffic conditions has been recognized as one of the key features of multiwavelength optical networks, this is the first in-depth study of the tradeoffs involved in carrying out the reconfiguration process. We first identify the degree of load balancing and the number of retunings as two important, albeit conflicting, objectives in the design of reconfiguration policies. We then formulate the problem as a Markovian Decision Process and we develop a systematic and flexible framework in which to study reconfiguration policies. We apply results from Markov Decision Process theory to obtain optimal reconfiguration policies for networks of large size. The advantages of optimal policies over a class of threshold-based policies are illustrated through numerical results.

1 Introduction

One of the key features of multiwavelength optical networks is *rearrangeability* [5], i.e., the ability to dynamically optimize the network for changing traffic patterns. This ability arises as a consequence of the independence between the logical connectivity and the underlying physical infrastructure of fiber glass. By employing tunable optical devices, the assignment of transmitting or receiving wavelengths to the various network nodes may be updated on the fly, allowing the network to closely track changing traffic conditions.

While rearrangeability makes it possible to design traffic-adaptive networks, the reconfiguration phase will interfere with existing traffic and disrupt network performance, causing a degradation of the quality of service perceived by the users. The issues that arise in reconfiguring a lightwave network by retuning a set of slowly tunable receivers have been studied in the context of multihop networks in [6, 7, 9]. In [6] the problem was to obtain a virtual topology that minimizes the maximum link flow, while in [7] algorithms to minimize the number of branch-exchange operations required to take the network from an initial to a target virtual topology were developed. The objective of [9] was to obtain near-optimal policies to dynamically determine when and how to reconfigure the network.

In this paper we study the reconfiguration issues in single-hop lightwave networks, an architecture suitable for LANs and MANs [8]. The single-hop architecture employs wavelength division multiplexing (WDM) whereby the various channels are dynamically shared by the attached nodes, and the logical connections change on a packet-by-packet basis creating all-optical paths between sources and destinations. Thus single-hop networks require rapidly tunable optical devices that can rapidly switch between channels.

When tunability only at one end, say, at the transmitters, is employed, each fixed receiver is permanently assigned to one of the wavelengths. Since the number of channels is expected to be smaller than the number of attached nodes, each channel will have to be shared by multiple receivers, and the problem of assigning receive wavelengths arises. Intuitively, a wavelength assignment (hereafter referred to as WLA) must balance the offered load across all channels, such that each channel carries an approximately equal portion of the overall traffic [11]. But with fixed receivers, the WLA cannot be updated in response to changes in the traffic pattern.

Alternatively, one can use *slowly tunable*, rather than fixed, receivers. While rapidly tunable lasers or filters have a tuning latency in the order of a packet transmission time at the high-speed rates at which optical networks are expected to operate, slowly tunable devices have tuning times that can be significantly longer. As a result, these devices cannot be assumed "tunable" for media access purposes as this requires fast tunability. Motivation for the use of slowly tunable devices for reconfiguration is provided by two factors. First, they can be significantly less expensive than rapidly tunable devices, making it possible to design network architectures that can be realized cost effectively. Second, the variation in traffic demands is expected to take place over larger time scales. Hence, even very slow tunable devices will be adequate for updating the WLA over time to accommodate varying traffic demands.

Assuming an existing WLA and information about the new traffic demands, a new WLA, optimized for the new traffic pattern, must be determined. In [1] we proposed an approach to reconfiguring the network that is minimally disruptive to existing traffic. Specifically, we devised the *GLPT* algorithm for obtaining a new WLA such that (a) the new traffic load is balanced across the channels, and (b) the number of receivers that need to be retuned to take the network from the old to the new WLA is minimized. For more details on the operation and performance of the *GLPT* algorithm, the reader is referred to [1].

While the network makes a transition from one WLA to another, some cost is incurred in terms of packet delay, packet loss, packet resequencing, and the control resources involved in re-

*This work was supported by the NSF under grant NCR-9701113.

0-7803-5417-6/99/$10.00 ©1999 IEEE.

tuning. Clearly, receiver retunings should not be very frequent, since unnecessary retunings affect network performance. 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, buffer overflows, and a decrease in the network's traffic carrying capacity. 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 paper we develop a systematic and flexible framework in which to study reconfiguration policies by formulating the problem as a Markovian Decision Process. We also develop an approximate model with a manageable state space, which captures the pertinent properties of the original model. We finally apply results from Markov Decision Process theory to obtain reconfiguration policies for networks of large size.

In Section 2 we present a model of the broadcast WDM network under study. In Section 3 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 4, and we conclude the paper in Section 5.

2 The Broadcast WDM Network

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 will refer to this tunability configuration as *rapidly tunable transmitter, slowly tunable receiver* (RTT-STR). All our results can be easily adapted to the dual configuration, STT-RTR.

We represent the current traffic conditions in the network by a $N \times N$ traffic demand matrix $\mathbf{T} = [t_{ij}]$, where t_{ij} is a measure of the long-term traffic from i to j. As traffic varies over time, the elements of matrix $\mathbf{T}$ will change. This variation in traffic takes place at larger time scales, e.g., we assume that changes in matrix $\mathbf{T}$ occur at connection arrival or termination instants. We also assume that the current matrix $\mathbf{T}$ completely summarizes the entire history of traffic changes, so that future changes only depend on the current values of the elements of $\mathbf{T}$.

During normal operation, each of the slowly tunable receivers is fixed to a particular wavelength. Let $\lambda(j) \in \{\lambda_1, \cdots, \lambda_C\}$ be the wavelength currently assigned to receiver j. A WLA is a partition $\mathcal{R} = \{R_c,\ c = 1, \cdots, C\}$ of the set $\mathcal{N} = \{1, \cdots, N\}$ of nodes, such that $R_c = \{j \mid \lambda(j) = \lambda_c\}$, $c = 1, \cdots, C$, is the subset of nodes currently receiving on wavelength λ_c. This WLA is used to determine the target channel for a packet given the packet's destination. Network nodes employ a media access protocol, such as HiPeR-ℓ [11], that requires tunability only at the transmitting end. Nodes use HiPeR-ℓ to make reservations, and schedule packets for transmission using algorithms that can mask the (short) latency of tunable transmitters [10].

We define the *degree of load balancing* (DLB) $\phi(\mathcal{R}, \mathbf{T})$ of a network with traffic matrix $\mathbf{T}$ and WLA $\mathcal{R}$ as:

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

The right hand side represents the bandwidth requirement of the dominant (most loaded) channel, while the second term in the left hand side represents the lower bound, with respect to load balancing, for 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. Thus, the DLB characterizes the efficiency of WLA $\mathcal{R}$ in meeting the traffic demands denoted by $\mathbf{T}$: the higher the value of ϕ, the less efficient the WLA is.

2.1 The Transition Phase

In order to more efficiently utilize the bandwidth 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*. We assume that reconfiguration is triggered by changes in the matrix $\mathbf{T}$. When such a change occurs, the following actions must be taken:

1. a new WLA 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 [1], 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 the paper 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 $\mathcal{R}'$. Let us define the distance $\mathcal{D}$ between the two WLAs $\mathcal{R}$ and $\mathcal{R}'$ as:

$$\mathcal{D}(\mathcal{R}, \mathcal{R}') = N - \sum_{c=1}^{C} | R_c \cap R'_c | \quad (2)$$

$\mathcal{D}(\mathcal{R}, \mathcal{R}')$ 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 [7] 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, 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. 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 it cannot receive data, and any packets sent to it are lost. If, on the other hand, the network nodes are aware that j is 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.

In general, the reconfiguration penalty associated with retuning a given number D of receivers depends 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 is beyond the scope of this paper. 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.

3 Markov Decision Process Formulation

3.1 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.:

$$\alpha_{min} \leq \alpha[\phi(\mathcal{R}', \mathbf{T}')] \leq \alpha_{max}$$

$$0 \leq \beta_{min} \leq \beta[\mathcal{D}(\mathcal{R}, \mathcal{R}')] \leq \beta_{max}$$

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

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

The first term in the right hand side 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.

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*

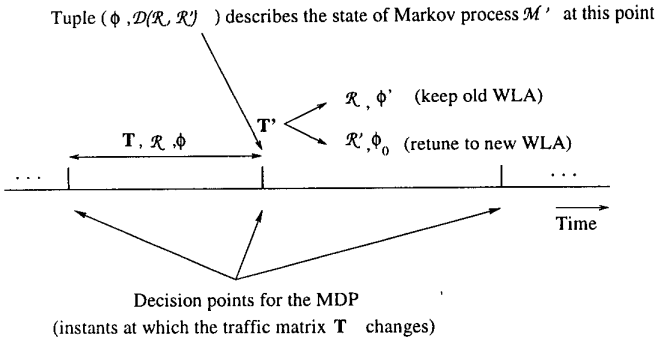

Figure 1: State of the new Markov process $\mathcal{M}'$

costs incurred and *current reward obtained*, and it is not obvious as to what constitutes an optimal policy. We also note that as $\beta_{min} \rightarrow \infty$, the optimal policy would be to never reconfigure, since this is the only policy for which the expected reward 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.

Consider an ergodic, discrete-space discrete-time Markov process with rewards and a set of alternatives per state that affect the probabilities and rewards governing the process. The *policy-iteration* algorithm in [4] can be used to obtain a policy that maximizes the long-term reward for such a process. A difficulty in applying the policy-iteration algorithm to the Markov process $\mathcal{M}$ is the potentially very large number of states $(\mathcal{R}, \mathbf{T})$. Next, 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.

3.2 Alternative Formulation

A closer examination of the reward function 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.

We now introduce a new process embedded, as process $\mathcal{M}$, at instants when the traffic matrix changes, as illustrated in Figure 1. The state of this process is 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. Since the policy-iteration algorithm can only be applied to a discrete-state process, we obtain such a process by using discrete values for random variable ϕ as follows.

The DLB ϕ can take any real value between 0 and $C-1$, where C is the number of channels. We 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 define a new discrete-state process $\mathcal{M}'$ with state (ϕ_k, D), and we use (ϕ_k, D) to represent any state (ϕ, D) of the continuous-state process with $\phi_k^{(l)} \leq \phi < \phi_k^{(u)}$. Clearly, the larger the number K of intervals, the better the approximation.

We make one further refinement to the new process $\mathcal{M}'$. The *GLPT* algorithm in [1] guarantees that the DLB of the obtained WLA 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 changes so that the current WLA provides a DLB $\phi < 0.5$ for the new traffic matrix. Then, we can safely assume that the load is well balanced and avoid a reconfiguration since there is no assurance that the DLB of the new WLA will be less than ϕ. Therefore, we choose to let $\phi_0^{(u)} = 0.5$. 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 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.

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 $K \rightarrow \infty$, 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 [4].

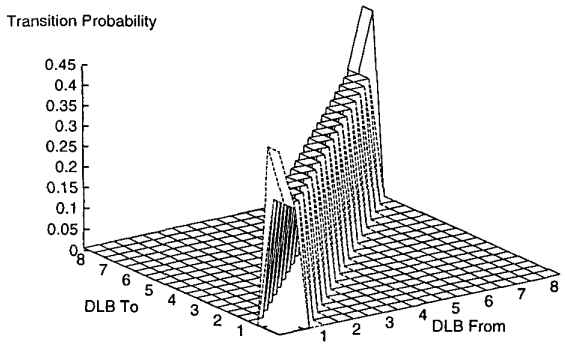

Figure 2: Near-neighbor model

4 Numerical Results

We demonstrate the properties of the optimal policies obtained by applying the policy-iteration algorithm [4] to the Markov decision process developed in Section 3.2. We also 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 (other traffic models have been considered in [3]). 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. Specifically, we assume that

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

This traffic model is illustrated in Figure 2 which plots the conditional probability $P[\phi_l \mid \phi_k]$ 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 above probabilities, we let the transition probability, *when no reconfiguration occurs*, from state (ϕ_k, D) to state (ϕ_l, D') be equal to:

$$P[(\phi_l, D') \mid (\phi_k, D)] = P[\phi_l \mid \phi_k]\, p_{D'} \tag{3}$$

where $p_{D'}$ is the probability that D' retunings will be required in the next reconfiguration. The probabilities p_D where measured experimentally, and we also observed that the probability that random variable D takes on a particular value is independent of the DLB ϕ_k, thus the expression (3).

We note that we need to obtain two different transition probabilities out of each state [4], 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 the distribution of D is independent of the DLB ϕ_k. Therefore, the transition probabilities under the reconfigure option are:

$$P[(\phi_l, D') \mid (\phi_k, D)] = \begin{cases} p_{D'}, & l = 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

We only consider the following reward and cost functions

$$\alpha[(\phi_k, D)] = \frac{A}{1+\phi_k}, \qquad \beta(D) = BD \tag{5}$$

where A and B are weights assigned to the rewards and costs; other reward and cost functions are studied in [2]. Our first 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 3.2, 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

$$\lim_{K\to\infty} d^{(K)}[(\phi_{k_K}, D)] = d[(\varphi, D)] \quad \forall\, \varphi,\, D \tag{6}$$

In Figures 3 to 5 we plot the decisions of the optimal policy to a 20-node, 5-wavelength network with a near-neighbor traffic model, and for three different values of K; the weights used in the functions (5) were set to $A = 30$ and $B = 1$. Figure 3 corresponds to the optimal policy for $K = 20$ intervals, while in Figures 4 and 5 we increase K to 30 and 40, respectively. The histograms shown in Figures 3 to 5, 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:

$$d^{(K)}[(\phi_k, D)] = \begin{cases} \text{reconfigure}, & D \leq D_k^{thr} \\ \text{do not reconfigure}, & D > D_k^{thr} \end{cases} \tag{7}$$

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 3 to 5.

As we can see in Figures 3 to 5, the decisions of the optimal policy do converge (in the sense of (6)) as K increases. Specifically, the policy decisions do not change when the number K of intervals increases from 30 to 40. 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 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 3 to 5 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 2), 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.

In Figures 6 to 8 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 2. For this network we used $K = 20$ intervals, and we varied the weights A and B in the reward and cost functions in (5) to study their effect on the optimal policy. Specifically, we let $B = 1$ and we varied A from 20 (in Figure 6) to 35 (in Figure 7) to 50 (in Figure 8). 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. Overall, we have found that one can obtain a wide variety of policies by varying the weights A and B.

We now compare the optimal policy against a class of threshold-based policies. 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 $D > D_{max}$. This class of policies define Markov processes which are *outside* the class of Markovian Decision Processes considered in Section 3. 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. In the threshold policies, on the other hand, the alternative selected does

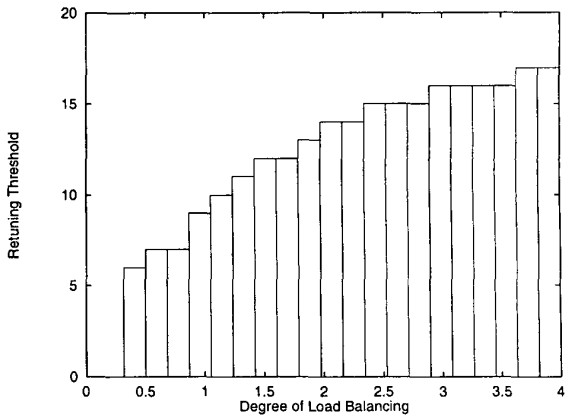

Figure 3: Optimal policy decisions for $N = 20$, $C = 5$, $K = 20$, $A = 30$, $B = 1$

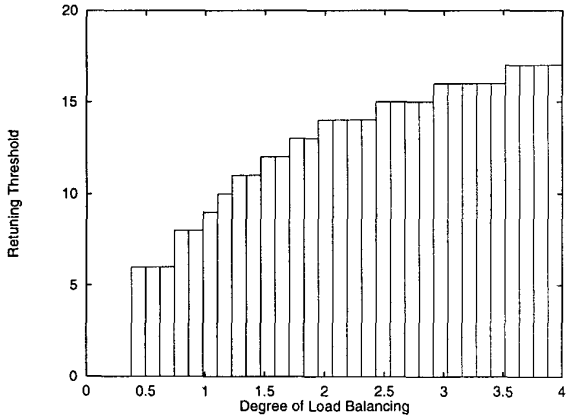

Figure 4: Optimal policy decisions for $N = 20$, $C = 5$, $K = 30$, $A = 30$, $B = 1$

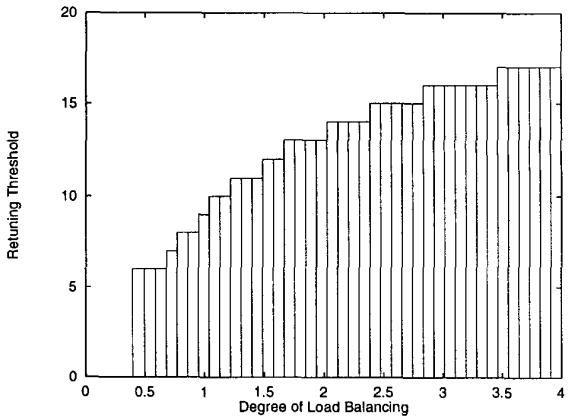

Figure 5: Optimal policy decisions for $N = 20$, $C = 5$, $K = 40$, $A = 30$, $B = 1$

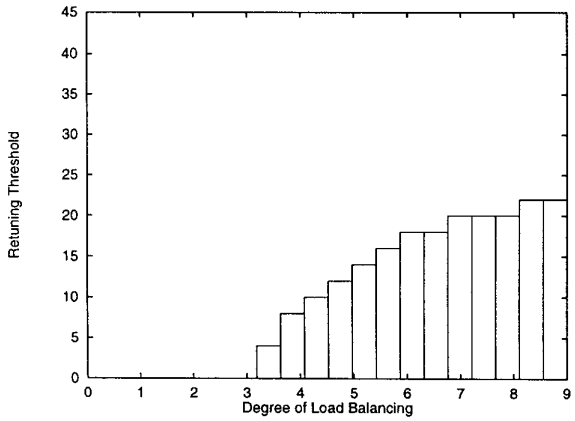

Figure 6: Optimal policy decisions for $N = 100$, $C = 10$, $K = 20$, $A = 20$, $B = 1$

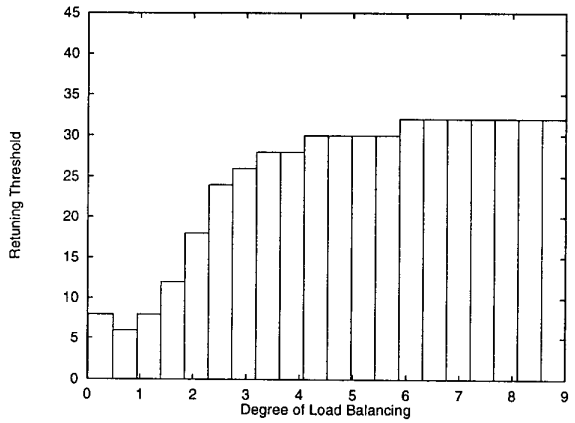

Figure 7: Optimal policy decisions for $N = 100$, $C = 10$, $K = 20$, $A = 35$, $B = 1$

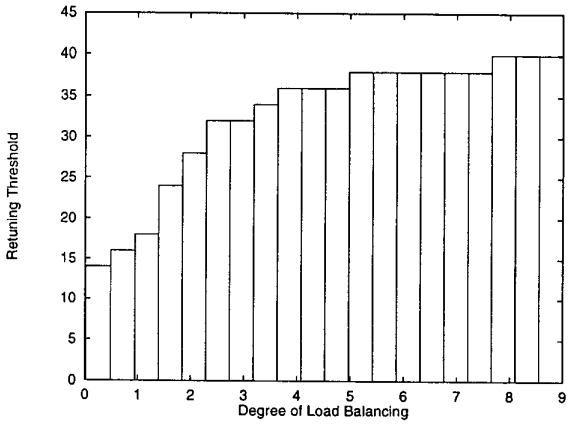

Figure 8: Optimal policy decisions for $N = 100$, $C = 10$, $K = 20$, $A = 50$, $B = 1$

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. Since Howard's algorithm [4] 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.

In Figure 9 we compare the optimal MDP policy to a number of two-threshold policies. The results presented are for a network with $N = 100$ nodes, $C = 10$ wavelengths, a near-neighbor traffic model, $K = 20$ intervals, and the reward and cost functions of (5) with $A = 50$ and $B = 1$. We plot the reward of each policy against the DLB threshold value; the optimal policy is independent of the DLB threshold, resulting in a horizontal line. We also plot the reward of three two-threshold policies, each of which corresponds to a different retuning threshold (namely, $D_{max} = 40, 32$, and 24) and varying DLB thresholds.

The most interesting observation from Figure 9 is that, for certain values of the DLB-threshold, the two-threshold policy with retuning threshold $D_{max} = 40$ achieves a higher reward than the optimal MDP 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 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 10 is similar to Figure 9, but we have used $A = 20$ and $B = 1$ as weights in the reward and cost functions, respectively. 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 the 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, and if the network's operating parameters change, threshold selection must be performed anew.

Overall, we have found that the optimal policy can successfully balance the two conflicting objectives, namely, the DLB and the number of retunings, and that, by appropriately selecting reward and cost functions, the optimal policy can be tailored to specific requirements set by the network designer.

5 Concluding Remarks

We have studied the problem of reconfiguring broadcast multiwavelength optical networks and we have used results from Markov Decision Process theory to obtain optimal reconfigura-

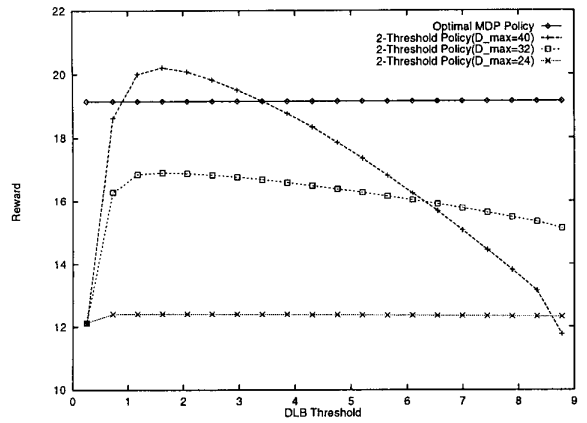

Figure 9: Policy comparison, $N = 100$, $C = 10$, $K = 20$, $A = 50$, $B = 1$

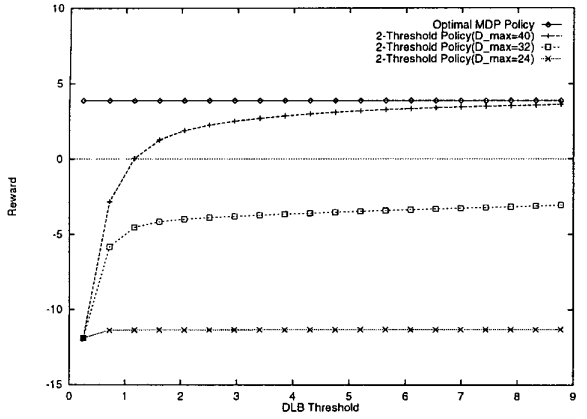

Figure 10: Policy comparison, $N = 100$, $C = 10$, $K = 20$, $A = 30$, $B = 1$

tion policies. The formulation presented provides a unified framework for reconfiguration problems in optical networks, and provides further insight into the fundamental tradeoffs involved in the design of reconfiguration policies.

References

[1] I. Baldine and G. N. Rouskas. Dynamic load balancing in broadcast WDM networks with tuning latencies. *INFOCOM '98*, pages 78–85, Mar. 1998.

[2] I. Baldine and G. N. Rouskas. On the design of dynamic reconfiguration policies in broadcast WDM networks. TR-98-06, NCSU, Raleigh, NC, June 1998.

[3] Ilia Baldine. *Dynamic Reconfiguration in Broadcast WDM Networks*. PhD thesis, NCSU, Aug. 1998.

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

[5] J-F. P. Labourdette. Traffic optimization and reconfiguration management of multiwavelength multihop broadcast lightwave networks. *CN & ISDN Systems*, 1998.

[6] J-F. P. Labourdette and A. S. Acampora. Logically rearrangeable multihop lightwave networks. *IEEE Tran. Commun.*, pages 1223–1230, Aug. 1991.

[7] J-F. P. Labourdette, *et al.* Branch-exchange sequences for reconfiguration of lightwave networks. *IEEE Trans. Commun.*, pages 2822–2832, Oct. 1994.

[8] B. Mukherjee. WDM-Based local lightwave networks Part I: Single-hop systems. *IEEE Network Magazine*, pages 12–27, May 1992.

[9] G. N. Rouskas and M. H. Ammar. Dynamic reconfiguration in multihop WDM networks. *J. High Speed Networks*, pages 221–238, 1995.

[10] G. N. Rouskas and V. Sivaraman. Packet scheduling in broadcast WDM networks with arbitrary transceiver tuning latencies. *IEEE/ACM Trans. Netw.*, pages 359–370, June 1997.

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