Dynamic Reconfiguration in Multihop WDM Networks

*George N. Rouskas**
Computer Science Department
North Carolina State University
Raleigh, NC 27695-8206

Mostafa H. Ammar
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

Abstract

The advent of fast tunable optical transceivers makes it feasible to dynamically update the connectivity of multihop WDM networks to accommodate traffic demands that vary over time. Of major concern in such design is how the connectivity should react to changes in traffic patterns. We formulate the problem as a Markovian Decision Process, and we develop heuristics to obtain good reconfiguration policies and manage the state and decision space explosion.

1 Introduction

Wave Division Multiplexing (WDM) divides the low-loss wavelength spectrum of the optical fiber into independent channels, each operating at a data rate accessible by the attached network stations. WDM introduces transmission concurrency and provides a means to overcome the speed mismatch between electronics and optics. As a result, WDM networks may deliver an aggregate throughput that can grow with the number of wavelengths deployed, and can be in the order of Terabits per second.

In multihop networks each station is equipped with a small number of transceivers [1]. An assignment of transmit and receive wavelengths defines an interconnection pattern independent of the underlying physical topology. Packets are relayed to their destination through, possibly, intermediate stations, undergoing conversion from the optical to the electrical domain at each hop. By properly assigning the wavelengths the connectivity can be optimized with respect to some performance parameters, such as the mean packet delay [2], and the maximum link flow [3].

In environments where traffic demands change over time, it is desirable to have the network connectivity dynamically respond to these changes. With the advent of fast tunable optical transceivers [4], it is feasible to contemplate the design of such networks. Of major concern in such design is *when* and *how* the connectivity should react to changing traffic patterns. The approach taken by Labourdette and Acampora [5] is to reconfigure the network infrequently, and only when the traffic pattern changes dramatically or when the current connectivity cannot accommodate the traffic load. Reconfiguration is achieved through a series of branch exchange operations, whereby only one pair of transceivers is retuned at a time. At the other extreme, Auerbach and Pankaj [6] have devised a distributed algorithm to rearrange the connectivity at, potentially, the beginning of every packet burst. Their algorithm recursively tries to establish 1-hop, 2-hop, etc., paths, and can handle concurrent requests.

These approaches suffer from two problems. First, no attempt is made to model the effect of the reconfiguration phase on the overall network performance. The transition from one connectivity to another incurs some cost due to packet loss and the control resources involved in transceiver retuning; this cost is not taken into account in the design process. Secondly, the issue of when to reconfigure the network has been decided upon *a priori*, without investigating alternative solutions or considering the trade-offs involved.

In this paper we start by modeling the effect of the reconfiguration phase on network performance in terms of packet loss. We then take this penalty into account in the design of reconfiguration policies. Therefore, the reconfiguration policy and, consequently, the frequency of reconfiguration is determined by the extent of packet loss.

In Section 2 we present a model of the network and of the reconfiguration phase. In Section 3 we formulate the problem as a Markovian Decision Process, and in Section 4 we develop a heuristic to obtain good configuration policies. We conclude the paper in Section 5.

2 Network Model

We consider a network of N stations, each equipped with p transceivers attached to a broadcast optical medium that can support $C = pN$ wavelengths (see Figure 1). In tunable-transmitter, fixed-receiver (TT-FR) networks, each receiver is assigned a unique receive wavelength, while the transmitters can tune over the entire range of wavelengths; similarly for a fixed-transmitter, tunable-receiver (FT-TR) network. The tuning delay is defined as the time it takes a transceiver to tune from one wavelength to another, and can be different for different transceivers and/or wavelength pairs. For our purposes, knowledge of Δ_{min}, the minimum tuning delay in the network, is sufficient.

We define a *template* as a logical connectivity that provides at least one path between any pair of stations. For a given network (i.e., for a given N and p), a large number of different templates is possible. In general, the set of templates, $\mathcal{T}$, that we will consider will be a subset of the set of all templates, and will be derived using information about the traffic characteristics. At any time instant the connectivity will be described by a template $\tau \in \mathcal{T}$. The connectivity can be changed to a new template $\tau' \in \mathcal{T}$ by retuning all or some of the transceivers. Communication in the network is connection oriented; a connection must be

*This work was performed while the first author was with the College of Computing, Georgia Institute of Technology.

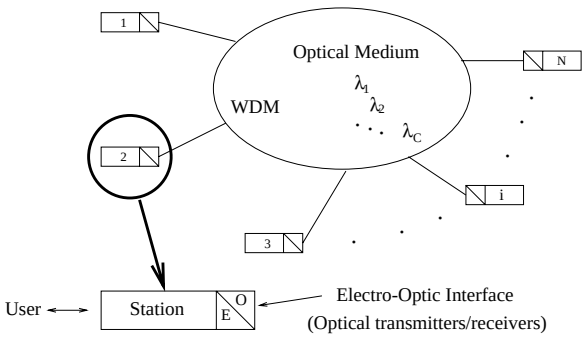

Figure 1: A Lightwave WDM Network

established prior to any data been transferred between two stations. Connections are established by issuing *connect* requests. A *disconnect* request is issued at the conclusion of a session. A connection, c, is identified by two end-point stations and its duration follows an exponential distribution with mean $\frac{1}{\mu_c}$. The time between the termination of connection c until it is requested again is exponentially distributed with mean $\frac{1}{\lambda_c}$.

2.1 The Reconfiguration Phase

In general, reconfiguration of the network connectivity from one template to another will be triggered by the occurrence of an *event*. When such an event occurs, several actions must be taken: (1) a new connectivity (template) must be determined, based on the current connectivity and the information carried by the triggering event, (2) the decision to reconfigure, as well as the new connectivity must be communicated to all the stations, not just those that will have to retune their transceivers, since the routing tables may need to be updated, and (3) the actual transceiver retuning must take place. The rest of the paper addresses the problem of determining the new connectivity. We now focus on the other two issues.

One option for reporting a reconfiguration triggering event would be to have a dedicated station detect the occurrence of events, compute the new connectivity, and inform all other stations. A distributed version would require each station to detect local events and report them, possibly on a common control channel employing TDMA. In the latter case, the station reporting an event may also compute and transmit the new connectivity. A problem may arise due to concurrent events arriving at different parts of the network. A solution would be to have the stations report only the occurrence of events; at the end of each TDMA cycle on the control channel all stations would use the same algorithm to determine the new connectivity based on the events that took place during the last cycle.

Let t_r be the time a reconfiguration triggering event, e, takes place. The event will be detected by one or more stations and will be reported to the network, possibly by one of the mechanisms discussed above. Regardless of the implementation, a station i will find out about the occurrence of e at time $t_r + T_i(e)$; $T_i(e)$ is the delay introduced by the event reporting mechanism. This delay is a function of the event e (if i detects e then $T_i(e) = 0$, otherwise $T_i(e) > 0$), and, in general, $T_i(e) \neq T_j(e)$ for $i \neq j$.

In order to eliminate inconsistencies in routing tables and minimize the need for synchronization among the network stations, the reconfiguration phase must be as short as possible. We, therefore, require that all stations retune their transceivers "simultaneously". For the distributed environment under consideration "simultaneously" should be interpreted as "as soon as they find out about the reconfiguration triggering event". In particular, station i's actions at time $t_r + T_i(e)$ for each of its transceivers that needs to be retuned are as follows: (1) complete the transmission (reception) of the current packet, if any, (2) retune the transceiver to the new wavelength and, at the same time, update the routing tables to reflect the new connectivity, and (3) start transmitting (receiving) packets as soon as retuning is complete. If a transceiver does not need to be retuned, its operation is not affected.

2.2 Packet Loss

We are interested in the effect of the reconfiguration phase on packet loss. Lost packets have to be retransmitted, increasing the average delay experienced by an application. Also, some loss-sensitive applications may not tolerate excessive packet loss. We now show how to compute the packet loss during the reconfiguration phase for a TT-FR network. The case of FT-TR networks is very similar and is omitted.

Let RP be a reference point in the network, such that the optical signal passing through it is the combination of the signals of all the transmitters in the network. The propagation delay from station i to RP is given by d_i. In Figure 2 we show the occurrence of a reconfiguration event that causes the transmitter of j (denoted by X_j) to retune to wavelength λ, used previously by the transmitter of i (which now will retune to a new wavelength). The vertical axis shows the distance of the two stations from RP, while the horizontal axis represents time. The figure shows a worst case scenario, in the sense that (a) at time $T_r + T_i$ when i is informed about the upcoming reconfiguration, it has just started a packet transmission packet on wavelength λ and has to delay the retuning of its transmitter until the transmission is completed, and (b) j starts retuning its transmitter at the earliest possible time, $t_r + T_j$, its tuning delay is equal to Δ_{min}, and it has a packet to send immediately after tuning to wavelength λ. The first bit of j's packet will arrive at RP at time $t_r + T_j + \Delta_{min} + d_j$, while the last bit of i's packet will arrive at RP at time $t_r + T_i + T_P + d_i$; T_P is the packet transmission time. As a result, there is a time interval of length

$$\max\{0, d_i - d_j + T_i - T_j + T_P - \Delta_{min}\} \qquad (1)$$

during which, packets by either i or j arriving at RP may collide; for a worst case scenario, we may assume that *all* packets arriving at RP within this time interval will collide.

Observe that in some cases no packets will collide (for example, if in Figure 2 we interchange the positions of i and j relative to RP). Also, in the case of an ATM switch when all stations would be within the same room or building, we may have $d_i \approx d_j, T_i \approx T_j \forall\ i, j$, and the collision interval

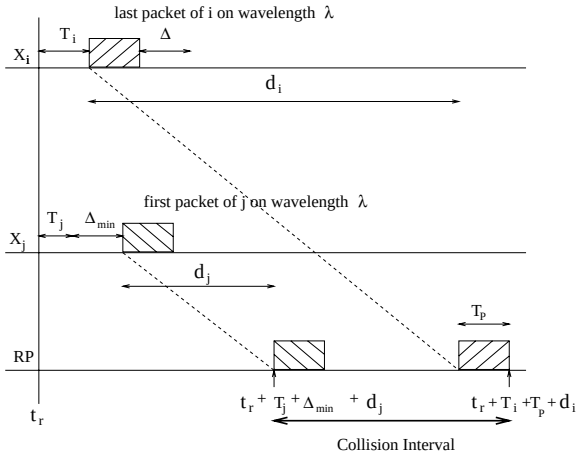

Figure 2: Packet loss during reconfiguration (TT-FR)

can be as short as $\max\{0, T_P - \Delta_{min}\}$. Another way to reduce packet loss is to delay transmissions from station j in Figure 2 by a time δ_j such that $d_j + T_j + \delta_j \geq \max_i\{d_i + T_i\}$, provided that j has enough buffer capacity to store packets arriving during a time interval equal to δ_j.

In general, packet loss cannot be altogether eliminated. Our model can then be used to identify limitations in the network size and frequency of reconfiguration, or the buffer capacity so that packet loss be within acceptable levels. Recovery from lost packets is assumed to take place via some higher level protocol.

3 Configuration Policies

The state of the network is defined as a tuple $(\mathbf{v}, \tau)$. $\mathbf{v}$ is a connection state that describes the established connections; $\tau \in \mathcal{T}$ is a template representing the current network connectivity. Changes in the network state occur at connect and disconnect request instants. Since we define the connect and idle times to have exponential distributions, our system is Markovian. We will refer to $\mathbf{\Omega}$, the set of all possible connection states, and $\mathcal{T}$, the set of all templates, as the *state* and *decision* spaces, respectively.

A network in state $(\mathbf{v_1}, \tau_1)$ will enter state $(\mathbf{v_2}, \tau_2)$ if a connection request or termination causes the connection state to change form $\mathbf{v_1}$ to $\mathbf{v_2}$. Implicit in the state transition is that the system makes a decision to reconfigure into template τ_2. In order to completely define the Markovian state transitions associated with our model we need to establish *next template* decisions. The decision is a function of the current state and the next event and is denoted by $d[(\mathbf{v}, \tau), event]$. Setting $d[(\mathbf{v}, \tau), event] = \tau_{next}$ implies that if *event* occurs while the system is in this state, the network should be reconfigured into template τ_{next}. Note that τ_{next} can be the same as τ, in which case the decision is not to reconfigure. A decision needs to be defined for each possible system state and for each valid event. Connect and disconnect requests are the only valid events. The set of decisions for all network states defines a *configuration policy*. A given configuration policy in conjunction with rates $\{\lambda_c\}$ and $\{\mu_c\}$ completely defines a continuous time, discrete state Markov process.

3.1 Problem Formulation

Our objective is to obtain a configuration policy such that the "cost" of running the network is minimized. We now formulate the problem as a Markovian Decision Process (MDP). There are two ways in which an MDP incurs cost: (1) *Transition Cost*, which is incurred in a lump sum when a state transition occurs, and (2) *State Occupancy Cost*, which is directly proportional to the time spent in each state. The transition cost from state $(\mathbf{v_1}, \tau_1)$ to state $(\mathbf{v_2}, \tau_2)$ is a function of τ_1 and τ_2 and is due to the packet loss and the control resources involved in transceiver retuning. Let $\nu(t)$ be the number of times the template had to be changed up to time t under some policy, z. Let $r_k, k = 1, \ldots, \nu(t)$, be the number of packets lost during the k-th reconfiguration, and $l(t)$ be the number of packets generated in the network up to time t. We define the average *reconfiguration cost*, $\mathcal{R}_z$, the fraction of packets lost during the operation of the network under policy z, as:

$$\mathcal{R}_z = \lim_{t\to\infty} \inf \frac{\sum_{k=1}^{\nu(t)} r_k}{l(t)} \tag{2}$$

We consider a state occupancy cost that is proportional to the distance travelled by a packet. Let $(\mathbf{v}(t), \tau(t))$ be the network state at time t, and $h_c(\tau)$ be the distance travelled by packets of connection c when the connectivity is described by template τ. The average *hop* cost incurred by policy z is given by [1]

$$\mathcal{H}_z = \lim_{t\to\infty} \inf \frac{1}{t} \int_0^t \frac{\sum_{c\in\mathbf{v}(t)} h_c(\tau(t))}{\sum_{c\in\mathbf{v}(t)} 1}\, dt \tag{3}$$

We define the *total cost* for policy z as:

$$\mathcal{A}_z = \alpha\mathcal{H}_z + \beta\mathcal{R}_z \tag{4}$$

where α and β are weights assigned to the costs. The basic idea is to use these weights to appropriately define a total performance measure. Consider for example the case when the important performance measure is average packet delay. Let α be $1/v$, where v is the speed of light in the optical medium. Let β be the average time-out interval. Then $\beta\mathcal{R}_z$ is the extra delay experienced by packets that are lost and have to be retransmitted, and $\mathcal{A}_z$ gives the average packet delay. On the other hand, for some loss-sensitive applications the only performance measure may be packet loss, in which case we may set $\alpha = 0, \beta = 1$.

3.2 Optimal Configuration Policies

Howard's *policy-iteration* algorithm [7] is guaranteed to produce a configuration policy that minimizes $\mathcal{A}_z$. Its complexity, however, is directly proportional to the number of states and events, which grows very rapidly with N and $|\mathcal{T}|$, and, in general, it is not possible to obtain the optimal configuration decisions. Our approach was to apply

[1] We will use $c \in \mathbf{v}$ to denote that connection c is "on" in the connection state $\mathbf{v}$.

the algorithm to a small network and identify the properties of optimal configuration policies, which were then used to develop heuristics practical for large networks.

We applied Howard's algorithm to a network with $N = 4, p = 2$ [8] after making the following simplifying assumptions: (a) for any template we considered, if a receiver of station i is tuned to a transmitter of station j, then a receiver of j is also tuned to a transmitter of i, (b) the distance between any two stations was taken to be equal to 1 (c) we assumed that a connection is always routed over a minimum distance path in the current template, and (d) instead of (2) we used $\mathcal{R}_z = \lim_{t\to\infty} \inf \frac{1}{t} \sum_{k=1}^{\nu(t)} s_k$, where s_k denotes the number of transceivers retuned in the k-th reconfiguration instant. We do feel, however, that our conclusions about the relative performance of the various policies are not affected by these simplifications (the effect of different levels of packet loss was captured by adjusting the value of β). Based on our experiments and from various common sense arguments it can be surmised that the basic pattern followed by an optimal configuration policy is:

> When the reconfiguration cost is heavily weighted compared to the hop cost, the decisions most of the time are not to reconfigure. Usually, the network enters a template that provides the minimum average hop cost and stays there forever. As the relative weight of the hop cost is increased the network tends to reconfigure to templates in which it incurs lower hop cost at the expense of incurring some reconfiguration cost. When the weight of the hop cost exceeds a certain threshold, the network, at each transition, reconfigures to one of the templates that provide the minimum hop cost for the next connection state.

The policies at the two ends of the policy "spectrum" (the *no reconfiguration policy* and *configure for minimum hop cost* policy) can be easily determined. However, the points at which these policies become optimal are not easy to determine. In what follows we concentrate on a class of policies for which the decision as to which template to use upon entering a new connection state is only a function of that state, or $d[(\mathbf{v}, \tau), event] = \tau_{next} = f(\mathbf{v_{next}})$. Our examination of these policies is motivated by two factors. First, it is relatively straightforward to compute the cost of such policies. Secondly, this type of policy has been observed in our experiments for a wide range of parameters.

4 Near-Optimal Policies

Our objective is to find, within the class of policies described above, a dynamic configuration policy with low cost. We start with the optimal policy for $\beta = 0$ and modify it to make decisions similar to those of the optimal policy for $\beta > 0$. When $\beta = 0$ the optimal policy dictates that the network be reconfigured to the minimum hop cost template for $\mathbf{v_{next}}$. Since for this class of policies the Markov process consists of a single chain and there are no transient states, we may compute the hop and reconfiguration costs as follows.

$$\mathcal{H}_z = \sum_{\mathbf{v}\in\mathbf{\Omega}} P(\mathbf{v}) \;\; HopCost(\mathbf{v}, f(\mathbf{v})) \tag{5}$$

$$\mathcal{R}_z = \sum_{\mathbf{v}\in\mathbf{\Omega}} P(\mathbf{v}) \;\; ReconfCost(\mathbf{v}, f(\mathbf{v})) \tag{6}$$

$$ReconfCost(\mathbf{v}, \tau) = \sum_{\mathbf{u}\in\mathbf{\Omega}} \gamma_{\mathbf{vu}} RCost(\tau, f(\mathbf{u})) \tag{7}$$

$$P(\mathbf{v}) = \left(\prod_{c\in\mathbf{v}} \frac{\lambda_c}{\lambda_c + \mu_c}\right)\left(\prod_{c\notin\mathbf{v}} \frac{\mu_c}{\lambda_c + \mu_c}\right) \tag{8}$$

$P(\mathbf{v})$ is the probability that the network is in connection state $\mathbf{v}$, $HopCost(\mathbf{v}, \tau)$ is the hop cost and $ReconfCost(\mathbf{v}, \tau)$ is the reconfiguration cost that the network incurs when at state $\mathbf{v}$ and template τ, $RCost(\tau_i, \tau_j)$ is the cost to reconfigure from template τ_i to template τ_j, and $\gamma_{\mathbf{vu}}$ is the transition rate from state $\mathbf{v}$ to state $\mathbf{u}$ [2].

We use Heuristic 1 to obtain good configuration policies.

Heuristic 1

1. *Optimal Policy for $\beta = 0$.* For each connection state $\mathbf{v}$ let τ be the template for which the hop cost of $\mathbf{v}$ is minimized. Set $f(\mathbf{v}) = \tau$.

2. *Local Improvement.* For each $\mathbf{v}$ consider all $\tau \in \mathcal{T}$ as possible candidates for $template f(\mathbf{v})$. Let $\tau' \in \mathcal{T}$ such that

$$\alpha HopCost(\mathbf{v}, \tau') + \beta ReconfCost(\mathbf{v}, \tau') = \min_{\tau\in\mathcal{T}}\{\alpha HopCost(\mathbf{v}, \tau) + \beta ReconfCost(\mathbf{v}, \tau)\} \tag{9}$$

   Set $f(\mathbf{v}) = \tau'$. Repeat for all $\mathbf{v}$ until no further cost reduction is possible.

3. *Template Removal.* For each $\tau \in \mathcal{T}$ do the following: for each $\mathbf{v}$ such that $f(\mathbf{v}) = \tau$, set $f(\mathbf{v}) = \tau' \in \mathcal{T} - \{\tau\}$ and τ' is selected as in Step 2. If the new policy incurs lower cost set $\mathcal{T} = \mathcal{T} - \{\tau\}$, otherwise restore the old policy. Repeat for the new $\mathcal{T}$ until no further cost reduction is possible.

Step 2 of the heuristic goes through the state space and modifies the decisions at each state to improve the initial policy. Step 3 goes through the decision space and removes templates (i.e., the final policy does not consider them as decision alternatives [3]) if the cost of making a transition to these templates is high.

4.1 Policies for Large Systems

As the number of states and alternatives per state grows exponentially with N and $|\mathcal{T}|$, Heuristic 1 becomes inefficient even for networks of moderate size. We now propose a way to manage the state and decision space explosion.

[2] $\gamma_{\mathbf{vu}}$ is equal to λ_c or μ_c for some c, or 0 if no connect/disconnect request can take the connection state from $\mathbf{v}$ to $\mathbf{u}$.

[3] We say that a template τ is "active" if $\exists \mathbf{v} \in \mathbf{\Omega} : f(\mathbf{v}) = \tau$.

β	Policy for $\beta = 0$	Policy After Step 2			Policy After Step 3			Active
	$\alpha\mathcal{H}_z + \beta\mathcal{R}_z$	$\mathcal{H}_z$	$\mathcal{R}_z$	$\alpha\mathcal{H}_z + \beta\mathcal{R}_z$	$\mathcal{H}_z$	$\mathcal{R}_z$	$\alpha\mathcal{H}_z + \beta\mathcal{R}_z$	Templates
0	1222.12	12.22	1.46	1222.12	12.22	1.46	1222.12	15
50	1294.68	12.22	1.41	1292.54	12.22	1.40	1292.51	14
100	1367.44	12.22	1.41	1363.16	12.28	1.33	1361.43	11
150	1440.20	12.22	1.41	1433.78	12.32	1.30	1427.64	9
200	1512.95	12.22	1.41	1504.38	12.37	1.27	1490.81	7
250	1585.71	12.22	1.41	1575.00	12.55	1.19	1553.03	6
300	1685.47	12.22	1.41	1645.66	13.03	0.96	1591.95	4
500	1949.50	12.26	1.39	1921.24	15.98	0.00	1597.85	1

Table 1: Results for $N = 16, \mathcal{P} = .9, M = 15, \alpha = 100, \lambda_c = .194, \mu_c = .1, c = 1, \ldots, 11, \lambda_c = .1, \mu_c = 99.9, c = 12, \ldots, 120$

Managing the Connection State Space. We define a set of "important" connection states, restricting our attention to a small subset, $\mathbf{\Omega}_{\mathcal{P}}$, of the connection state space. To this end we use algorithm ORDER-II [9] to efficiently enumerate the most probable connection states until a desirable degree, $\mathcal{P}, 0 < \mathcal{P} \leq 1$, of coverage of the state space, is obtained. The main justification for doing this lies in the fact that the network will be operating in one of the "important" states most of the time.

Managing the Decision Space. We only consider a small number, M, of templates, selected so as to minimize the hop cost the network will occur while in the states in $\Omega_{\mathcal{P}}$ [8]. Heuristic 2 describes our approach. By adjusting the values of $\mathcal{P}$ and M we can trade the quality of the final policy for speed.

Heuristic 2

1. Given $\mathcal{P}$, use ORDER-II [9] to produce $\mathbf{\Omega}_{\mathcal{P}}$.
2. Given $\mathbf{\Omega}_{\mathcal{P}}$ and M obtain $\mathcal{T}$ such that $\mid \mathcal{T} \mid = M$ [8].
3. Apply Heuristic 1 to obtain $f(\mathbf{v}) \in \mathcal{T}\ \forall \mathbf{v} \in \mathbf{\Omega}_{\mathcal{P}}$.
4. For each $\mathbf{v}$ and $\tau \in \mathcal{T}$, if *event* takes the network to $\mathbf{u} \notin \mathbf{\Omega}_{\mathcal{P}}$, set $d[(\mathbf{v}, \tau), event] = \tau$.

Step 4 ensures that when the network makes a transition to a connection state not in $\mathbf{\Omega}_{\mathcal{P}}$ the decision is not to reconfigure, and no reconfiguration cost is incurred. The hop cost experienced while in states not in $\mathbf{\Omega}_{\mathcal{P}}$ is not expected to constitute a significant part of the total cost. An upper bound on this extra cost is $(1 - \mathcal{P})HopCost_{max}$, where $HopCost_{max}$ is the highest hop cost incurred by any state.

4.2 Numerical Results

We consider a network with $N = 16, p = 2$ and traffic parameters as in Table 1. Using ORDER-II [9] a 90% coverage of the state space is achieved by the 2047 most probable states, a tiny fraction of the total number of states, which is equal to 2^{120}. The results presented in Table 1 are for $M = 15, \alpha = 100$, and various values of β. From the table we observe that for a given value of $\beta > 0$, Steps 2 and 3 of Heuristic 1 improve on the cost of the policy produced by the previous step. As β increases the final policies incur higher hop cost and lower reconfiguration cost; this is desirable as the importance of the reconfiguration cost increases with β. Also, templates for which the reconfiguration cost is prohibitively high are not considered by the policies for high β values; when β exceeds a certain threshold the best policy is to choose one template and never reconfigure.

5 Concluding Remarks

A solution approach based on Markovian Decision Process theory has been presented for the problem of dynamic connectivity reconfiguration in multihop WDM networks. Current research work focuses on alternative models and policies, as well as policy implementation mechanisms.

References

[1] B. Mukherjee. WDM-Based local lightwave networks Part II: Multihop systems. *IEEE Network Magazine*, pages 20–32, July 1992.

[2] J. A. Bannister, L. Fratta, and M. Gerla. Topological design of the wavelength-division optical network. In *Proceedings of INFOCOM '90.* IEEE, 1990.

[3] J-F. P. Labourdette and A. S. Acampora. Logically rearrangeable multihop lightwave networks. *IEEE Trans. Commun.*, 39(8):1223–1230, August 1991.

[4] C. A. Brackett. Dense wavelength division multiplexing networks: Principles and applications. *IEEE JSAC*, SAC-8(6):948–964, August 1990.

[5] J-F. P. Labourdette, A. S. Acampora, and G. W. Hart. Reconfiguration algorithms for rearrangable lightwave networks. In *Proceedings of INFOCOM '92.* IEEE, May 1992.

[6] J. Auerbach and R. Pankaj. Use of delegated tuning and forwarding in WDMA networks. Technical Report RC 16964, IBM Research Report, 1991.

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

[8] G. N. Rouskas and M. H. Ammar. On the design of dynamic configuration policies for multihop lightwave networks. Technical Report GIT-CC-91/47, Georgia Institute of Technology, Atlanta, GA, 1991.

[9] Y. F. Lam and V. O. K. Li. An improved algorithm for performance analysis of networks with unreliable components. *IEEE Trans. Commun.*, COM-34(5):496–497, May 1986.