An Algorithm for Traffic Grooming in WDM Mesh Networks with Dynamically Changing Light-Trees

Xiaodong Huang[†], Farid Farahmand[‡], and Jason P. Jue[†]
[†]Department of Computer Science
[‡]Department of Electrical Engineering
The University of Texas at Dallas
Richardson, Texas 75083-0688
Email: {xxh020100, ffarid, jjue}@utdallas.edu

***Abstract*— In this paper, we address the traffic grooming problem in WDM mesh networks with dynamic unicast traffic. We develop a *dynamic tree grooming algorithm (DTGA)* that can support multi-hop traffic grooming by taking advantage of light-trees. In this algorithm, a light-tree can be dropped, branched, and extended when a route is to be established for a new request. In order to implement the *DTGA*, we develop a layered graph model which can support different routing policies. Extensive simulation shows that *DTGA* has better performance than lightpath-based algorithms when transceivers are limited.**

I. Introduction

Wavelength division multiplexing (WDM) is becoming the dominant technology in backbone networks. To effectively utilize the huge bandwidth of a WDM network, *lightpaths* can be established in the network [1]. A lightpath is an all-optical communication channel that spans one or more links between a source and a single destination.

Generally, the capacity of a lightpath is much higher than the bandwidth requirement of an individual user. Dedicating an exclusive lightpath to each request results in low channel efficiency. One approach to improve the network efficiency is to pack several sub-wavelength traffic requests into a single wavelength channel. The problem of multiplexing and routing low speed traffic requests over lightpaths, as well as determining their routing and wavelength assignments of lightpaths is known as the *traffic grooming problem* [2].

A number of traffic grooming studies have been dedicated to SONET over WDM ring networks under static traffic scenarios, where the traffic is known in advance. Recently, traffic grooming in WDM mesh networks has begun to receive considerable attention. In [3], the authors formulate the traffic grooming problem as an Integer Linear Programming (ILP) problem and propose several simple heuristic algorithms. A generic graph model is presented in [4] to support traffic grooming. This model was further applied to the dynamic traffic grooming scenerio [5], in which traffic requests dynamically arrive (or leave), and lightpaths are dynamically set up (or torn down).

These previous works on traffic grooming adopted lightpaths as their building blocks. An alternative approach is to establish and share *light-trees* to satisfy incoming requests. A light-tree is a wavelength channel that can reach more than one destinations all-optically [6]. A light-tree-based grooming algorithm has been proposed in [7]. In this approach, when a new request arrives, it attempts to extend existing light-trees rooted at the source node of the request in order to reach the new destination. If no such light-tree is found, a new lightpath will be established for the request. This algorithm supports only *single-hop traffic grooming*.

In this paper, we address the traffic grooming problem in WDM mesh networks with dynamic unicast traffic, based on a dynamically changing light-tree model. We develop a *dynamic tree grooming algorithm (DTGA)* that can support *multi-hop traffic grooming* by taking advantage of light-trees. In this algorithm, a light-tree can drop and/or branch at an intermediate node and can extend at leaf nodes as we set up the route for a new request. A branch on a light-tree will be torn down when no more routing passes through it. The algorithm is implemented using a layered graph model. Our graph model can support both light-path and light-tree paradigms while the model in [4][5] supports only lightpath. Extensive simulation shows that *DTGA* has better performance than lightpath-based algorithms when transceivers are limited.

The rest of this paper is organized as follows. In Section II, we describe the node architecture supporting the *DTGA* algorithm. We define the grooming problem in Section III. Details of the *DTGA* algorithm will be discussed in Section IV. In Section V, we will present the simulation results. Section VI concludes the paper.

II. Node Architecture

In this section we describe the architecture for supporting the proposed dynamic tree grooming algorithm. We consider a two-layered node architecture consisting of an electronic and a photonic layer with two distinct capabilities: *traffic grooming* and *optical multicasting*. We refer to this architecture, shown in Fig. 1(a), as a multicast-capable grooming optical cross-connect (MCG-OXC).

In this architecture, traffic grooming capability is offered by the electronic layer, which allows the multiplexing of low speed traffic into high-speed channels as well as the demultiplexing of incoming high-speed traffic. The demultiplexed traffic can be *fully* dropped and switched to local clients or

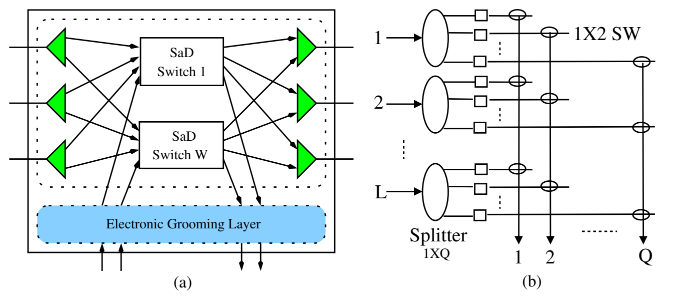

Fig. 1. Multicast-capable grooming optical cross-connect (MCG-OXC) architecture. (a) The node architecture; (b) The SaD switch.

partially aggregated with other incoming and local traffic to be retransmitted optically to the next hop.

Multicast-capable optical cross-connects, residing in the photonic layer of the MCG-OXC, perform optical power splitting and optical switching. One approach to realize the multicasting capability is to employ splitter-and-delivery (SaD) switch architecture [8], shown in Fig. 1(b). In SaD-based cross-connects, each incoming wavelength goes through an optical power splitter and can be sent to any number of output ports. The strictly non-blocking characteristic of SaD-based cross-connects ensures that no existing connection will be interrupted as light-trees change dynamically.

III. Problem definition

We formulate the traffic grooming problem as follows:

- Given:
  - The topology of the physical network
  - The number of wavelengths on each fiber link
  - The number of transceivers on each node
- Under the assumptions:
  - Each node is an MCG-OXC with full splitting and full grooming capability and with no wavelength converters.
  - All transceivers are tunable to all wavelengths.
  - Requests are unicast with sub-wavelength bandwidth demand. They arrive and depart randomly. A request can not be split at the source node or any intermediate nodes.
  - The routing of a request could be single hop or multi-hop. If a routing can not be found for a request, the request will be blocked and discarded immediately.
- Find:
  - The routing for an incoming request
- Minimize:
  - The average blocking probability.

IV. Algorithm

In this section, we first propose an approach to generate an auxiliary graph according to the state of the network. Based on this auxiliary graph, we propose a new dynamic tree grooming algorithm. A shortest path found on the auxiliary graph can be mapped back to a routing in the network. By changing the weight of different edges in the auxiliary graph, different routing policies can be applied to the algorithm. The basic principles behind the algorithm are that light-trees are adopted to groom low speed traffic flows and that the light-trees can extend or contract dynamically as needed.

A. Definitions and notations

The physical network can be represented by $G_0 = (V_0, E_0)$. There are w wavelengths on each fiber. We will generate an auxiliary graph $GG = (V, E)$, which has $w + 1$ layers: w *wavelength layers* and one *grooming layer*. Wavelength layers are used to map the network state on each wavelength. The grooming layer is used to abstract the grooming capability of the network. All wavelength layers are connected to the grooming layer by *AddEdge*s and *DropEdge*s (defined in the next subsection). Note that there is no direct connection between wavelength layers.

We define three types of vertices to abstract the capability of an MCG-OXC as follows.

- *Grooming Vertex (GVT)*: represents the grooming capability of an MCG-OXC. Any wavelength can be added to or dropped on a *GVT* if there are available transmitters or receivers, respectively. In *GG*, there is a *GVT* for each node.
- *Transmitting Vertex (TVT)*: abstracts a transmitting port for a specific wavelength on an MCG-OXC. The *TVT* is connected to a remote *RVT* (see below) on a neighbor node according to the physical network topology. There are w *TVT*s in *GG* for each transmitting port on a node, one for each wavelength.
- *Receiving Vertex (RVT)*: abstracts a receiving port for a specific wavelength on an MCG-OXC. The *RVT* is connected to a remote *TVT* on a neighbor node according to the physical network topology. There are w *RVT*s in *GG* for each receiving port on a node, one for each wavelength.

Therefore, for each node u on the physical network, there will be one *GVT*, wd_u *TVT*s and wd_u *RVT*s on the auxiliary graph, where d_u denotes the degree of node u.

B. Generation of the auxiliary graph

The auxiliary graph *GG* is initially generated as follows.

- Generate a wavelength layer for each wavelength. First, for each node on G_0, add a *TVT* and a *RVT* to the layer for each transmitting and receiving port, respectively. Then, for each node on G_0, add a *pass-through edge (PTEdge)* from each *RVT* to each *TVT* within the node. At last, for each fiber link on G_0, add a *wavelength link edge (WLKEdge)* from the *TVT* at a node to the *RVT* at the neighboring node.
- Generate the grooming layer. For each node on G_0, add a *GVT* to the layer. Note that there are no edges between *GVT*s.
- Connect wavelength layers and the grooming layer. Within each node, add an *Adding Edge (AddEdge)* from the *GVT* to the *TVT*s at each layer; add a *Dropping Edge (DropEdge)* from the *RVT*s at each layer to the *GVT*.

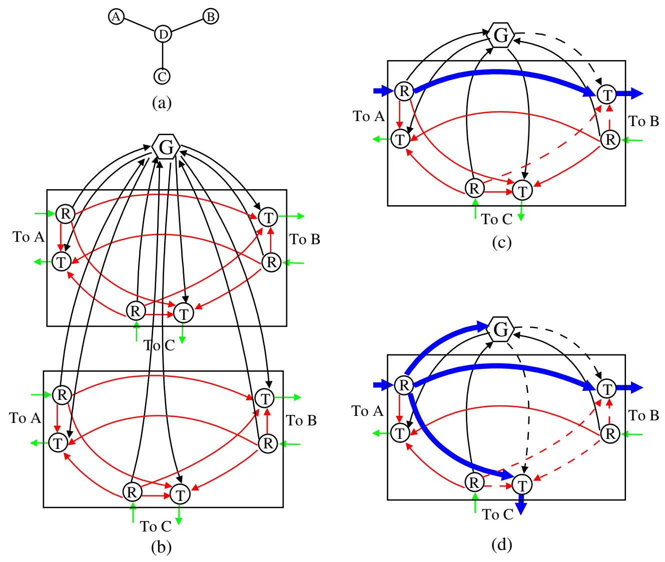

Fig. 2. An illustrative example for implementing the *DTGA* algorithm. T, R, and G represent *TVT*, *RVT*, and *GVT*, respectively. Dash-lines indicate edges which have been deleted from the graph. Thick lines represent light-tree. (a) Physical network; (b) Layered representation of node D; (c) A light-tree passing through node D; (d) A light-tree dropping and branching at node D.

As an example, consider a four-node network shown in Fig. 2(a), in which all physical links are bi-directional and there are two wavelengths on each fiber. Fig. 2(b) shows the initial graph representation of node D. There is a *GVT* at the grooming layer; there is a *RVT* and a *TVT* connecting to nodes A, B and C, at wavelength layer 1 and wavelength layer 2. On both wavelength layers, *RVT*s can reach all *TVT*s through *PTEdge*s. All *RVT*s can reach the *GVT* through *DropEdge*s and the *GVT* can reach all *TVT*s through *AddEdge*s, since initially node D has available transceivers.

An edge can either be used by a light-tree or be freely available. Each edge has an associated *weight*, *capacity*, and *residual capacity*. The weight will be assigned according to the routing policy (see next subsection). The capacity is the maximum traffic an edge can carry. Note that the capacity of a *WLKEdge* is equal to the capacity of the wavelength channel, whereas, all other edges have unlimited capacity. The residual capacity is the available capacity of an edge which can be used to carry new traffic. The *WLKEdge* is the only edge type that has limited residual capacity which is initially equal to its capacity and changes dynamically.

C. Dynamic Tree grooming algorithm (DTGA)

The *DTGA* algorithm has two routines, *DTGA.SETUP* and *DTGA.TEARDOWN*. Once the auxiliary graph *GG* is initially constructed, the *DTGA.SETUP* routine will be executed each time a new request arrives. If the request can be satisfied, the *DTGA.SETUP* will set up the route and update the auxiliary graph to reflect the current state of the network; otherwise, the request will be blocked. On the other hand, each time a request terminates, the *DTGA.TEARDOWN* will be executed and the auxiliary graph will be updated.

We describe the details of *DTGA.SETUP* for a new request $Req(s,d,B)$, where s and d are the source and destination nodes, respectively, and B is the bandwidth demand of the request.

- Step 1: Check the residual capacity of each *WLKEdge* on each layer and delete it if its residual capacity is less than B.
- Step 2: Search the shortest path on *GG* from the *GVT* at the source node to the *GVT* at the destination node by running Dijkstra's shortest path algorithm. If no such a path exists, discard the request and restore all *WLKEdges* deleted in Step 1. Otherwise, continue to Step 3.
- Step 3: Iterate through each edge on the shortest path to establish the route.
  - For each optical hop (starting with and ending at a *GVT*), if none of *TVT*s and *RVT*s is on any light-tree, set up a new light-tree along the vertices on the optical hop. On the other hand, if some of *TVT*s and *RVT*s are on a light-tree, extend the light-tree to cover the remaining vertices.
  - For each *TVT* on the shortest path, delete all incoming edges, except the *AddEdge* or the *PTEdge* that is on the path.
  - Update the residual capacity of each *WLKEdge* along all light-trees on the shortest path.
- Step 4: Check the number of transceivers at each node. If no transmitter is available, delete unused *AddEdge*s on all wavelength layers at the node. Similarly, if no receiver is available, delete unused *DropEdge*s on all wavelength layers at the node.
- Step 5: Restore all *WLKEdge*s deleted in Step 1.

Every time a request is terminated the *DTGA.TEARDOWN* routine operates as follows.

- Step 1: Remove the request's traffic demand from all light-trees along the request path.
- Step 2: Tear down all *inactive branches* which are no longer carrying effective traffic. If all branches on a light-tree are inactive, remove the entire tree.
- Step 3: Update the network state accordingly. We omit the details due to space limitation.

The complexity of the *DTGA.SETUP* routine results primarily from Dijkstra's shortest path algorithm, which is implemented on the auxiliary graph. Let n and d be the number of nodes and the maximum node degree in the network, respectively. We get $|V|$=$O(dwn)$ and $|E|$=$O(dwn)$. Since GG is a very sparse graph, the complexity of Dijkstra's algorithm is $O(E \log V)$. The complexity of all other operations in the algorithm is equivalent to $O(dwn)$. Therefore, the complexity of the *DTGA.SETUP* and *DTGA.TEARDOWN* is $O((dwn)\log(dwn))$ and $O(dwn)$, respectively.

We continue our illustrative example in Fig. 2 to demonstrate these concepts. For simplicity, we show only a single wavelength layer in Fig. 2(c) and Fig. 2(d). Fig. 2(c) demonstrates the status of node D as a portion of a light-tree (shown by the thick line) entering node D from node A and continuing to node B. The traversing light-tree in the figure is carried over three distinct edges: the *WLKEdge* from node A to node D, the *PTEdge* from *RVT* vertex to *TVT* vertex within node D, and the *WLKEdge* from node D to node B. Consequently, since the *TVT* connected to node B is being used at node D, the *AddEdge* and the other two *PTEdge*s directed to the *TVT* vertex must be deleted, shown by dashed lines. Note that *DropEdge*s and *PTEdge*s allow light-trees to drop and branch, respectively. For example, Fig. 2(d) shows the status of node D when the light-tree is dropped at node D and branched to node C.

D. Lightpath-based grooming algorithm

For the purpose of performance comparison, we implement a lightpath-based algorithm using our proposed auxiliary graph model by adding the following operation to Step 3 in the *DTGA.SETUP* routine:

- For each *RVT* on the shortest path, delete all outgoing edges, except the *DropEdge* or the *PTEdge* that is on the path.

In *DTGA*, when a light-tree is set up for the first request, the light-tree is actually a lightpath. With the above rule, the algorithm eliminates the possibility of dropping or extending existing light-trees. Therefore, it implements the lightpath-based grooming algorithm.

E. Routing policies

In the *DTGA* algorithm, the shortest path for a request depends on the weight assignment of edges, which in turn depends on the *routing policy*. A routing policy is a criterion used to select the best possible route.

We consider the following four routing policies for the *DTGA* algorithm. 1) Minimum physical hops (*MPH*); 2) Minimum logical hops (*MLH*); 3) Minimum extra transceivers (*MTR*); 4) Minimum total on-tree physical hops (*MTH*). The MTR policy implies that, if existing light-trees can satisfy a request, we should never set up new light-trees. This approach attempts to utilize the minimum number of transmitter used by each request. The MTH is different from the MPH policy in the sense that the weight for all edges on a light-tree is the same as the sum of all physical hops on the light-tree.

Routing polices can be implemented in *DTGA* by assigning different weights to different types of edges in *GG*. Since the shortest path algorithm always tries its best to avoid edges with high weight values, we can simply assign a huge weight value to those edges which we desire to be used as infrequently as possible. In our algorithm, for MPH and MTH, we make *WLKEdge*s as the highest weight edges; for MLH, we make *AddEdge*s and *DropEdge*s as the highest weight edges; for MTR, we make *unused AddEdge*s as the highest weight edges.

V. Numerical results and analysis

In this section we present the performance results obtained by implementing the *DTGA* algorithm. We have chosen the NSFNet backbone, shown in Fig. 3, as our test network, and consider the following assumptions for the simulation environment:

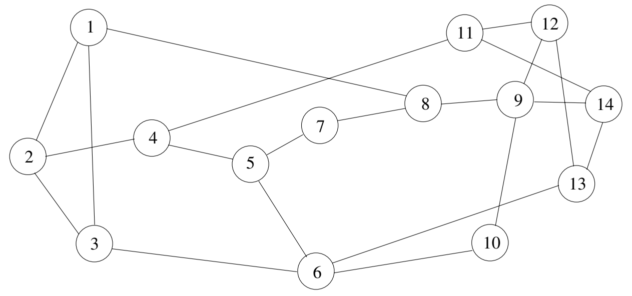

Fig. 3. The NSF network with 14 nodes and 21 bi-directional links.

- Traffic requests are generated and terminated dynamically. The traffic arrival is a Poisson process and the request's duration is a negative exponential distribution. Traffic is uniformly distributed among all node pairs.
- Each link on the NSFNet is bi-directional with a fiber in each direction. There are 4 wavelengths in each fiber with the capacity of OC-192.
- All nodes have full splitting and full grooming capacity but have no wavelength converters.
- The number of transmitters and receivers per node are 4 and 6, respectively.
- Requests are unidirectional unicast with bandwidth demand at OC-12, OC-48, or OC-96 rates with the proportion of $8 : 1 : 1$.

Fig. 4 and Fig. 5 show the performance of the *DTGA* algorithm under different routing policies. Regardless of the policies, the *DTGA* algorithm outperforms the lightpath-based approach, both in terms of blocking probability and average number of logical hops. The reason is that the *DTGA* algorithm can use transmitters more efficiently, as much more leaf nodes on a light-tree can share the transmitter on the root of the light-tree than on a lightpath. On the other hand, under either very low or very high network load, *DTGA* can still improve the performance, but the improvement is not so significant.

Further analysis shows that the *DTGA* algorithm can achieve the lowest blocking probability under the MTH routing policy. Since any traffic carried on a light-tree will travel over all branches on the tree, the MTH routing policy is more accurate than any other polices in describing the amount of wavelength link resources used by the request that will be routed through the light-tree. On the other hand, the MLH policy results in the lowest average number of logical hops, since, under MLH, the *DTGA* algorithm will first attempt to satisfy requests by using single-hop optical paths. On the contrary, the MTR policy results in the worst performance either in terms of blocking probability or average number of logical hops. Under the MTR policy, existing light-trees extend to a greater number of nodes so that the *non-effective* traffic carried on *WLKEdge*s increases significantly.

Next, we examine the performance of the *DTGA* algorithm under the MTH routing policy with different number of receivers. For each node, the number of transmitters is set to 4, and the number of receivers varies from 4 to 12. The results are shown in Fig. 6 and Fig. 7. The results indicate that,

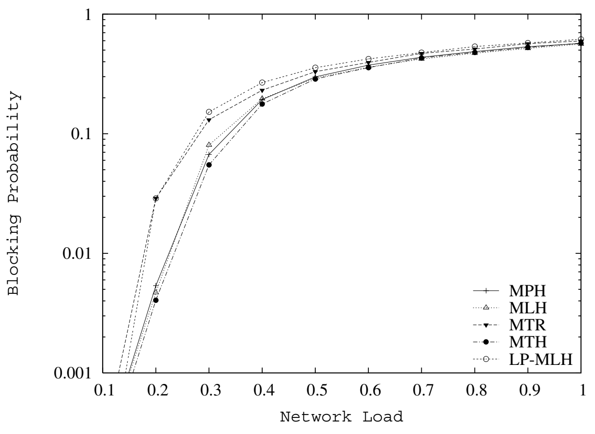

Fig. 4. Blocking probability under different routing policies.

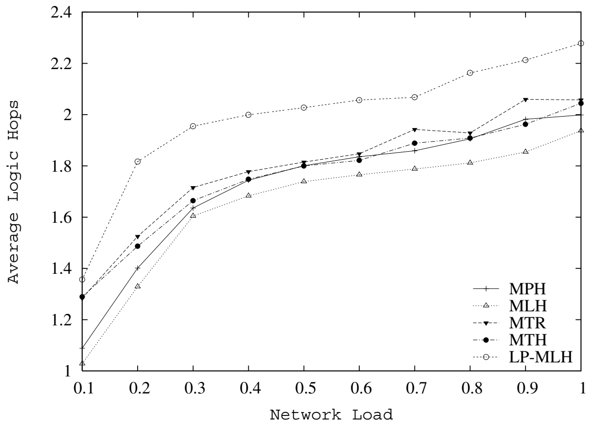

Fig. 5. Average logical hops under different routing policies.

as the number of receivers increases, the blocking probability and average number of logical hops decrease. The decrease is significant when the number of receivers is changed from 4 to 6. The decrease becomes less apparent as the number of receivers continues to increase beyond 8. In fact, hardly any improvement can be observed when the number of receivers is changed from 10 to 12. Note that a similar trend can be observed when other routing policies are utilized. These results imply that, in general, the *DTGA* algorithm is sensitive to the number of receivers and is able to utilize transceivers efficiently.

VI. Conclusion

In this paper, we have presented a grooming algorithm, *DTGA*, for the traffic grooming problem in WDM mesh networks which can support light-trees. This algorithm adopts dynamically changing light-trees as the building block and is implemented by using a graph model. Compared with previous algorithms for the problem, our algorithm has much better performance in terms of blocking probability and average number of logical hops. Our graph model can also be used without modification to support multicast traffic grooming. This topic may be addressed in future work.

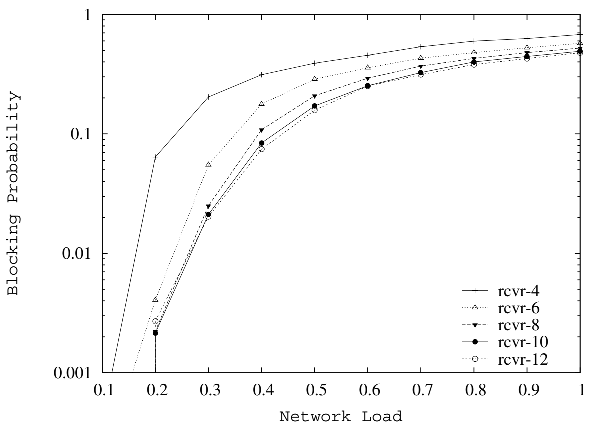

Fig. 6. Blocking probability under MTH with different number of receivers.

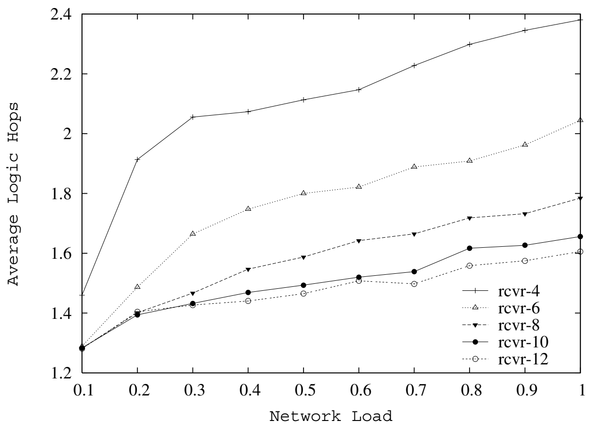

Fig. 7. Average logical hops under MTH with different number of receivers.

References

[1] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath communications: an approach to high bandwidth optical WAN's,"*IEEE Trans. Commun.* vol. 40, no. 7, pp. 1171-1182, Jul. 1992.

[2] R. S. Barr and R. A. Patterson, "Grooming telecommunication networks," *Optical Network Magazine*, vol. 2, no. 3, pp. 20-23, May/Jun. 2001.

[3] K. Zhu and B. Mukherjee, "Traffic grooming in an optical WDM mesh network," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 1, pp. 122-133, Jan. 2002.

[4] H. Zhu, H. Zang, and B. Mukherjee, "A novel generic graph model for traffic grooming in heterogeneous WDM mesh networks," *IEEE/ACM Trans. Networking*, vol. 11, no. 2, pp. 285-299, Apr. 2003.

[5] H. Zhu, H. Zang, K. Zhu, and B. Mukherjee, "Dynamic traffic grooming in WDM mesh networks using a novel graph model," *IEEE GLOBECOM '02*, vol. 3, pp. 2681-2685, Nov. 2002.

[6] L.H. Sahasrabuddhe and B. Mukherjee, "Light trees: optical multicasting for improved performance in wavelength routed networks," *IEEE, Communications Magazine*, vol. 37, no. 2, pp. 67-73, Feb. 1999.

[7] K. Zhu, H. Zang, and B. Mukherjee, "A comprehensive study on next-generation optical grooming switches," *IEEE Journal on Selected Areas in Communications* , vol. 21, no. 7, pp. 1173-1186, Sep. 2003.

[8] W.S. Hu and Q.J. Zeng, "Multicasting optical cross connects employing splitter-and-delivery switch," *IEEE Photonics Technology Letters*, vol. 10, no. 7, pp. 970-972, Jul. 1998.