Multicast Traffic Grooming in Wavelength-Routed WDM Mesh Networks Using Dynamically Changing Light-Trees

Xiaodong Huang, Farid Farahmand, and Jason P. Jue, *Senior Member, IEEE*

***Abstract*—In this paper, we address the online multicast traffic grooming problem in wavelength-routed wavelength division multiplexing (WDM) mesh networks with sparse grooming capability. We develop a multicast dynamic light-tree grooming algorithm (MDTGA) that can support multihop 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; a light-tree can also be contracted when some branches carry no effective traffic after requests depart from the network. The difficulty of the algorithm lies in the dynamic characteristic of light-trees. We implement MDTGA on a new auxiliary-graph model. For the purpose of comparison, we also implement a lightpath-based grooming algorithm by putting a constraint on the optical-splitting capability of the nodes. Through extensive simulations, we find that MDTGA has much better performance than the lightpath-based algorithm.**

***Index Terms*—Blocking probability, graph model, light-tree, multicast, routing, traffic grooming, wavelength division multiplexing (WDM).**

I. Introduction

MULTICAST is a communication paradigm in which a source sends out a single message to a network and the network forwards the message to multiple destinations. There are increasing demands for multicast applications, such as real-time video conferencing, stock information distribution, and online multiuser games [1]. Multicast applications demand more and more bandwidth on backbone networks.

Fortunately, wavelength division multiplexing (WDM) is becoming the dominant technology in backbone networks, enabling an optical fiber to carry more than one wavelength simultaneously. To further reduce the cost of electronic–optical–electronic (OEO) conversion and the cost of electronic processing, all-optical communication channels can be established in WDM networks with the help of an optical cross connect (OXC), through which optical signal on a wavelength could be switched in the optical domain [2]. Since an all-optical communication channel is assigned a specific wavelength, it is also referred to as wavelength channel.

Manuscript received December 16, 2004; revised July 20, 2005. This work was supported in part by the National Science Foundation (NSF) under Grants ANI-0133899 and CNS-0435105.

X. Huang and J. P. Jue are with the Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75083-0688 USA (e-mail: xxh020100@utdallas.edu; jjue@utdallas.edu).

F. Farahmand is with the Department of Electrical Engineering, The University of Texas at Dallas, Richardson, TX 75083-0688 USA (e-mail: ffarid@utdallas.edu).

Digital Object Identifier 10.1109/JLT.2005.856244

In WDM networks, there are two typical all-optical communication channels, lightpaths [3] and light-trees [4]. A lightpath is an all-optical communication channel that passes through all intermediate nodes between a source and a single destination without OEO conversion. A light-tree is an all-optical channel between a single source and multiple destinations. Like the lightpath, there is no OEO conversion at any intermediate node on a light-tree.

Using a light-tree to carry multicast traffic is a natural choice in WDM mesh networks. Much research has addressed the very fundamental multicast routing and wavelength-assignment problem, such as in [5]–[7]. In these studies, it is assumed that the bandwidth demand of a multicast request is at the level of the capacity of a wavelength channel. However, in practice, the bandwidth demand of a multicast request is usually much less than the capacity of a wavelength channel. If a wavelength channel is dedicated to only one multicast request, most of the capacity of the wavelength channel is wasted. Then, the problem is how to pack multiple multicast requests into wavelength channels, how to set up their routes, and how to assign wavelengths. This problem is known as the multicast traffic grooming problem.

Recently, the multicast traffic grooming problem has begun to attract research attention [8]–[10]. All of these papers studied the multicast traffic grooming problem based on the lightpath wavelength channel model and approached the problem using integer linear programming (ILP) and heuristics. A more comprehensive study on this topic can be found in [11]. The authors proposed a unified ILP formula for static traffic, including unicast, multicast, and mixed traffic. This previous literature studied a static multicast traffic grooming problem in which a traffic matrix is given in advance. However, to the best of the authors' knowledge, there is no study published on the dynamic multicast traffic grooming problem in which multicast requests randomly arrive at and depart from a network.

Without traffic grooming, the multicast routing problem can be converted to the well-known Steiner-tree problem, which is NP-complete [12]. With multicast traffic grooming, a wavelength channel will be shared among multiple multicast requests such that the routing and wavelength-assignment problem becomes more involved.

On the other hand, although the multicast traffic grooming problem is relatively new, a very related topic, the unicast traffic grooming problem, has received much attention. Many unicast traffic grooming studies have been dedicated to synchronous optical network (SONET) over WDM ring networks under

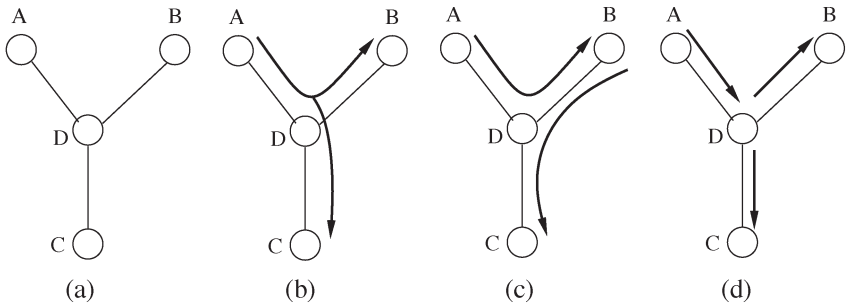

Fig. 1. Multicast routing with traffic grooming: an illustrated example. (a) A part of a network; (b) light-tree-based routing; (c) lightpath-based routing; and (d) link-by-link routing.

static traffic scenarios, where the traffic is known in advance, such as in [13]–[15]. Recently, there has been some work on unicast traffic grooming in WDM mesh networks. In [16], the authors formulate the traffic grooming problem as an ILP problem and propose several heuristic algorithms. A generic graph model is presented in [17] to support traffic grooming. This model was further applied to the dynamic traffic grooming scenario [18]. In these studies, the lightpath wavelength channel model is assumed.

Although it is easy to see that the light-tree is a better choice over lightpath for multicast requests with a bandwidth requirement of wavelength granularity, it is far from clear which is a better choice for the multicast traffic grooming problem. An illustrated example is shown in Fig. 1.

Fig. 1(a) is a small part of a backbone network. Suppose that there is a new incoming multicast request $\{A, \{B, C\}\}$ with subwavelength bandwidth demand, in which node A is the source and nodes B and C are the destinations. It is not difficult to find a minimum-cost Steiner tree for the request, as shown in Fig. 1(b). If the bandwidth demand is at the wavelength level, this is the optimal solution. However, with the subwavelength bandwidth demand, it is not necessarily the optimal solution. For example, suppose there is another subwavelength traffic request $\{\{A, \{B\}\}$. If the new traffic is carried on the light-tree, as in Fig. 1(b), the new traffic will also go through branch DC of the light-tree, which is a waste of bandwidth.

In this case, we might set up a lightpath-based route for the request $\{A, \{B, C\}\}$ and the request $\{A, \{B\}\}$, as in Fig. 1(c), which will eliminate the bandwidth waste for the new request. The disadvantage is that it consumes more bandwidth for the current request than the light-tree-based scheme. The most bandwidth-efficient approach is the link-by-link routing scheme, as shown in Fig. 1(d), either for the current request or for future requests. However, the lightpath-based routing scheme and the link-by-link routing scheme might need more electronic traffic grooming processing and consume more transceivers than a light-tree-based routing scheme. Since the link-by-link routing scheme does not take advantage of optical switching, we do not further consider it.

In general, a light-tree-based routing scheme has the trend to reduce the bandwidth consumption for the current request, but at the potential cost of waste on bandwidth for future requests, while a lightpath-based routing scheme basically has the opposite property of the light-tree scheme. Intuitively, it is not clear which routing scheme will give better performance. The above reasons make the multicast traffic grooming problem a new problem.

In this paper, we address the online multicast traffic grooming problem in wavelength-routed WDM mesh networks. We develop a multicast dynamic light-tree grooming algorithm (MDTGA) that can implement multihop 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; a light-tree can also be contracted when some branches carry no effective traffic after requests depart from the network. The difficulty of the algorithm lies in the dynamic characteristic of light-trees. We implement MDTGA on a new auxiliary-graph model. We also propose a lightpath-based online multicast traffic grooming algorithm. Through extensive simulations, we find that MDTGA has much better performance than the lightpath-based algorithm. Some other interesting observations are also presented.

The rest of this paper is organized as follows. In Section II, we describe the node architecture. We define the grooming problem in Section III. Details of the MDTGA 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 node architecture for supporting the proposed dynamic light-tree grooming algorithm. The node architectures should provide two basic functions: optical multicasting and electronic grooming.

One approach to realize the optical multicast is to employ a splitter-and-delivery (SaD) switch [19]. To reduce power loss, we take advantage a variance of the SaD switch by using configurable optical splitters [20], shown in Fig. 2(a). A configurable optical splitter is an active component that can adjust the splitting rate of its output ports as needed. Since we adopt dynamically changing light-trees, the number of child nodes of a node on the light-tree may change dynamically. With configurable splitters, unnecessary power loss could be avoided.

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 nonblocking characteristic of SaD-based cross connects [19] ensures that no existing connection will be interrupted as light-trees change dynamically. In addition, there is no blocking of traffic due to optical switching capability. It is also possible to adopt the shared

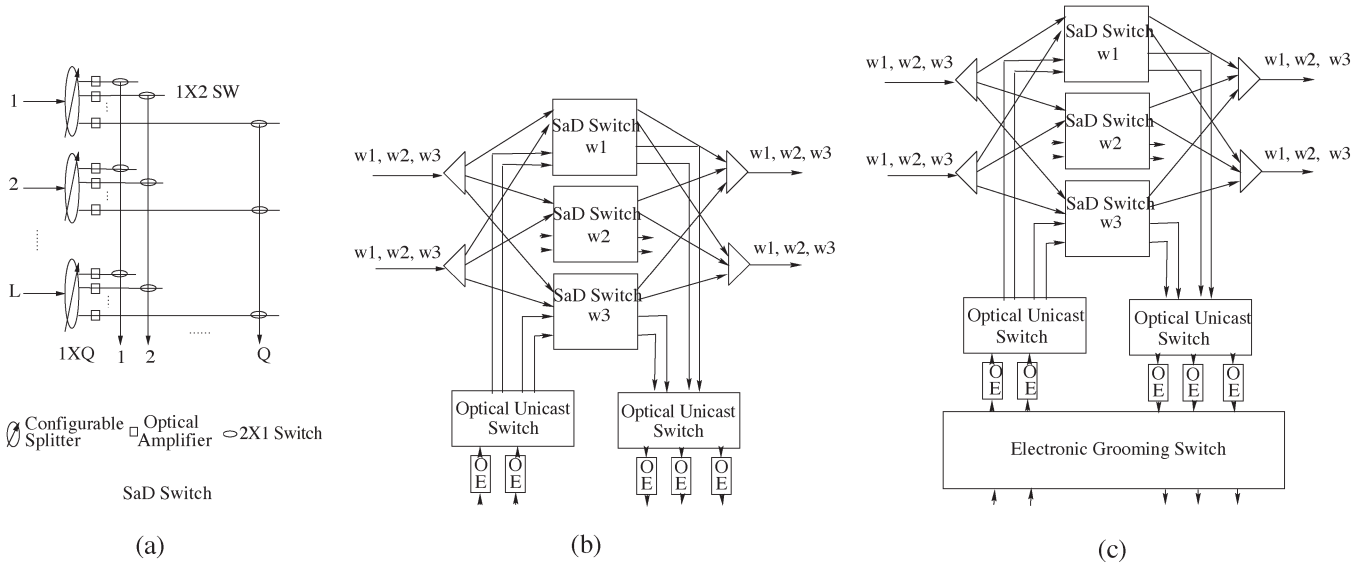

Fig. 2. Node architectures. (a) SaD switch with a configurable optical splitter; (b) node architecture with local grooming capability (MLG-OXC); and (c) node architecture with full grooming capability (MFG-OXC).

splitter bank architecture as in [4]. However, some restrictions need to be placed on the structure of some light-trees and blocking may also happen due to insufficient splitters. In this paper, we adopt the SaD-based cross-connect architecture for its simplicity.

At the same time, electronic grooming capability is necessary at some nodes in order to implement traffic grooming. From the traffic perspective, there could be three schemes of traffic grooming: 1) local traffic with local traffic; 2) local traffic with pass-through traffic; and 3) pass-through traffic with pass-through traffic.

The first grooming scheme is actually the same as the traditional multiplexing and demultiplexing function. In the second grooming scheme, the pass-through traffic will first be demultiplexed, and then will be multiplexed with local traffic. Usually, an electronic grooming switch at subwavelength granularity is needed to make this approach flexible enough for practical usage. In the third grooming scheme, there is switching between traffic of different wavelength channels. Hence, an electronic grooming switch is necessary. In general, this could be implemented by opaque OXCs, or with the help of dedicated electronic grooming switches. Since we need to set up light-trees, opaque OXCs cannot be used here.

Without loss of generality, we assume that the second grooming scheme needs an electronic grooming switch. We use two types of node architectures in this paper. Both have optical-multicast capability. The difference between the two architectures is the grooming capability. The first node architecture is shown in Fig. 2(b), which has no electronic grooming switch. It supports only grooming scheme 1. We refer to it as Multicast OXC with local grooming (MLG-OXC). The second node architecture has a dedicated electronic grooming switch, as shown in Fig. 2(c). The second architecture supports grooming schemes 2 and 3. We refer to it as Multicast OXC with full grooming (MFG-OXC). To efficiently support multicast, we also assume the grooming switch in an MFG-OXC has electronic multicast capability. Through OEO conversion and electric grooming, MFG-OXC also implements indirectly a kind of wavelength conversion function.

An MFG-OXC is much more expensive than an MLG-OXC, because the electronic grooming switch is expensive. Therefore, in this paper, we assume that only some of the nodes in the network are equipped with the MFG-OXC and the rest of the nodes are equipped with the MLG-OXC. We refer to the nodes with the MFG-OXC architecture as grooming hub nodes.

As we will see in Section V, by using the light-tree channel model, it is more effective to deploy more receivers than transmitters at each node. Since the optical unicast switch for receiving is independent of the optical unicast switch for transmitting, there is no special difficulty in implementing such an asymmetric node architecture as shown in Fig. 2(b) and (c).

III. Problem Statement

We formulate the multicast traffic grooming problem as follows:

- Given:
  a) the topology of the physical network;
  b) the number of wavelengths on each fiber link;
  c) the capacity of a wavelength channel;
  d) the number of transceivers at each node;
  e) the architecture of each node;
- Under the assumptions:
  a) only a small part of the nodes are grooming hub nodes;
  b) none of the nodes have wavelength converters;
  c) all transceivers are tunable to any wavelength;
  d) requests are multicast traffic with subwavelength bandwidth demand; requests arrive and depart randomly; a request cannot be split at the source node or any intermediate nodes;
  e) the routing of a request could be of single hop or multiple hop; the destinations for which no route could

be found will be blocked, while the routes for other reachable destinations will be set up.

- With the constraints:
  a) there is the same wavelength assignment for all wavelength links on a light-tree;
  b) traffic carried by a light-tree cannot exceed the capacity of a wavelength channel.
- Find:
  a) routes for arrival requests.
- To minimize:
  a) the average blocking probability.

IV. Multicast Grooming Algorithm

In this section, we will discuss the details of the proposed multicast grooming algorithm. We will first present the basic idea of the algorithm. Then, we will discuss the auxiliary graph for the algorithm. Next, we will describe the details of the algorithm. Finally, an illustrative example is given along with our discussion.

When a new multicast request arrives at the network, a route tree is going to be set up if there are enough resources available. The route tree itself may consist of multiple light-trees, which may each be on a different wavelength. The route from the source to a destination in the request may go through multiple light-trees. A light-tree may carry traffic from more than one request. When a multicast request departs from the network, the route tree for the request is released, as well as the resources taken by the route tree. Light-trees can be changed dynamically. Light-trees can extend by growing new branches or contract by pruning old branches as needed.

In order to take advantage of Steiner-tree algorithms, we design an auxiliary graph to represent the state of the network. We can search the route for a request on the auxiliary graph and we can also map the route tree in the graph back to the real network. The difficulty in designing the auxiliary graph results from the dynamic characteristic of the light-tree. Our proposed auxiliary graph model can overcome this difficulty. The idea is to model the connection relationship among ports within each node, since the port connections are the most fundamental dynamic factor in WDM networks.

A. Definitions and Notations

There are w wavelengths on each fiber. For a physical network, an auxiliary graph G will be generated, 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 adding edges and dropping edges. Note that there is no direct connection between wavelength layers.

We define four types of vertices to abstract the capability of an MLG-OXC or MFG-OXC as follows.

1) *Adding vertex*: This represents the ports from which traffic is injected into the network. In the auxiliary graph, there is one adding vertex for each node. Adding vertices are in the grooming layer.
2) *Dropping vertex*: This represents the ports from which traffic is dropped from the network. In the auxiliary graph, there is one dropping vertex for each node. Dropping vertices are in the grooming layer.
3) *Transmitting vertex*: This represents the physical transmitting ports to which an outgoing optical fiber is connected. In the auxiliary graph, there is one transmitting vertex at each wavelength layer for each physical transmitting port at a node.
4) *Receiving vertex*: This represents the physical receiving ports to which an incoming optical fiber is connected. In the auxiliary graph, there is one receiving vertex at each wavelength layer for each physical receiving port at a node.

For each node u in the physical network, there will be one adding vertex, one dropping vertex, $w \cdot d_u$ transmitting vertices, and $w \cdot d_u$ receiving vertices in the auxiliary graph, where d_u denotes the degree of node u. An adding vertex and a dropping vertex could be, respectively, the roots and the leaves of multiple light-trees. On the contrary, each transmitting vertex and receiving vertex can be used by one light-tree or be freely available. We refer to their status as busy and idle, respectively.

Next, we enumerate the following five types of edges in the auxiliary graph:

1) *adding edge*—from the adding vertex to a transmitting vertex within a node;
2) *dropping edge*—from a receiving vertex to the dropping vertex within a node;
3) *pass-through edge*—from a receiving vertex to a transmitting vertex within a node;
4) *grooming edge*—from the dropping vertex to the adding vertex within a node if the node is a grooming hub node;
5) *wavelength-link edge*—from a transmitting vertex of a node to a receiving vertex of a neighboring node, at the same wavelength layer.

Each edge in the auxiliary graph can either be used by one light-tree or can be freely available. We also refer to their status as busy and idle, respectively. Each edge has an associated capacity, residual capacity, and weight. The capacity is the maximum traffic an edge can carry. Note that the capacity of a wavelength-link edge is equal to the capacity of the wavelength channel, whereas all other edges have unlimited capacity. The residual capacity is the freely available capacity of an edge that can be used to carry new traffic. The wavelength-link edge is the only edge type that has limited residual capacity, which is initially equal to its capacity and changes dynamically. The weight assignment of edges is tightly coupled with the algorithm. We will explain the weight assignment after we discuss the algorithm.

A light-tree on the auxiliary graph will be rooted at an adding vertex, followed by an adding edge as the only edge adjacent to the root. A light-tree may span several pass-through edges and wavelength-link edges before it reaches dropping vertices through dropping edges at the destination nodes.

B. Initialization of the Auxiliary Graph

Initially, there is no multicast request carried on the network. All resources are available, including transmitters, receivers, and optical fibers. Therefore, the auxiliary graph G can be initially generated as follows.

1) *Generate a wavelength layer for each wavelength*. First, for each node on the physical network, add a transmitting vertex and a receiving vertex to the wavelength layer for each transmitting and receiving port, respectively. Then, for each node, add a pass-through edge from each receiving vertex to each transmitting vertex within the node. Finally, for each fiber link on the physical network, add a wavelength-link edge from the transmitting vertex at a node to the receiving vertex at the neighboring node.
2) *Generate the grooming layer*. For each node on the physical network, add an adding vertex and a dropping vertex to the grooming layer. If a node is a grooming hub node, add a grooming edge from the dropping vertex to the adding vertex of the node.
3) *Connect wavelength layers and the grooming layer*. Within each node, add an adding edge from the adding vertex at the grooming layer to each transmitting vertex at each wavelength layer; add a dropping edge from each receiving vertex at each wavelength layer to the dropping vertex at the grooming layer.

When light-trees are set up and torn down or light-trees are changing dynamically, resources will be taken and released. The auxiliary graph will be maintained to reflect the resource state of the network.

As an example, consider a four-node network shown in Fig. 3(a). All links in the network are bidirectional. Suppose there are two wavelengths per fiber and there is one transmitter and one receiver at each node. Fig. 3(b) shows the initial auxiliary graph. Let us take node D as an example. There are three transmitting ports and three receiving ports at node D. Hence, for node D, there are three transmitting vertices and three receiving vertices at each wavelength layer. Even though there is only one transmitter at each node, there is an adding edge from the adding vertex to all transmitting vertices within each node at both wavelength layers, since traffic can be sent from the adding vertex to any transmitting vertex at any wavelength.

C. Multicast Dynamic Tree Grooming Algorithm (MDTGA)

After the auxiliary graph is set up, the route for a multicast request needs to be found. From the graph theory point of view, this is the classic Steiner-tree problem, which is well known to be NP-complete [12]. There are many approximation algorithms to the Steiner problem. We may use some of these approximation algorithms to solve our problem.

One approximation algorithm is the shortest path tree algorithm [12]. The algorithm starts the route tree with the source node. It continues to include the destination that is the nearest to the partially established tree until all destinations are included in the tree. The algorithm has the running complexity of $O(nN^2)$ and an approximation upper bound of $O(2-2/n)$, where N is the number of vertices in the graph and n is the number of the destinations.

However, we cannot simply run a Steiner-tree algorithm on the auxiliary graph to find the route tree for a multicast request, since nodes in a network may only have a limited number of free transmitters. Traditional Steiner-tree algorithms, including the shortest path tree algorithm, may set up more light-trees rooted at a node than the number of available transmitters at the node, which is a violation of the transmitter resource constraint.

We take the following action to address this difficulty. After one new destination is added to the route tree, we introduce one additional operation: check the availability of transmitters at each node. If there is no transmitter available, all free adding edges are removed from the node. The algorithm adds the nearest destination to the route tree one by one until all destinations are included or no more destinations can be reached. Hence, simply speaking, our MDTGA algorithm is a variance of the shortest path tree algorithm, running on the proposed auxiliary graph. The building block to support traffic grooming is the dynamically changing light-trees.

The MDTGA algorithm has two routines, setup and teardown. Once the auxiliary graph G is initially constructed, the setup routine will be executed each time a new request arrives. The setup routine will establish a route to as many destinations as possible and update the auxiliary graph to reflect the current state of the network. On the other hand, each time a request terminates, the teardown routine will be executed to release resource occupied by the request, and the auxiliary graph will be updated accordingly.

We describe the details of the setup routine for a new request $\mathrm{Req}(s, D, b)$, where s is the source, $D = \{d_1, d_2, \ldots, d_n\}$ is a set of the destination nodes, and b is the bandwidth demand of the request.

1) Check the residual capacity of each wavelength-link edge on each layer and delete it if its residual capacity is less than b. Initially, the route tree for the request includes only the adding vertex at the source node. For each destination in D, repeat 2) to 4).
2) Search the shortest paths on the auxiliary graph from the partially established route tree to the dropping vertices of all remaining destinations. Choose the destination which is nearest to the partially established route tree. If no such destination exists, go to 5); otherwise, continue to 3).
3) Iterate through vertices and edges on the shortest path to the chosen destination to augment the route tree.
   a) For any idle transmitting vertex or receiving vertex, change its state to be busy.
   b) For an idle adding edge, change its state to be busy and do the following operations: i) reduce the number of free transmitters by 1; and ii) delete all idle pass-through edges that end at the transmitting vertex of this adding edge.
   c) For an idle dropping edge, change its state to be busy and reduce the number of free receivers by 1.
   d) For an idle pass-through edge, change its state to be busy and i) delete all other idle pass-through edges

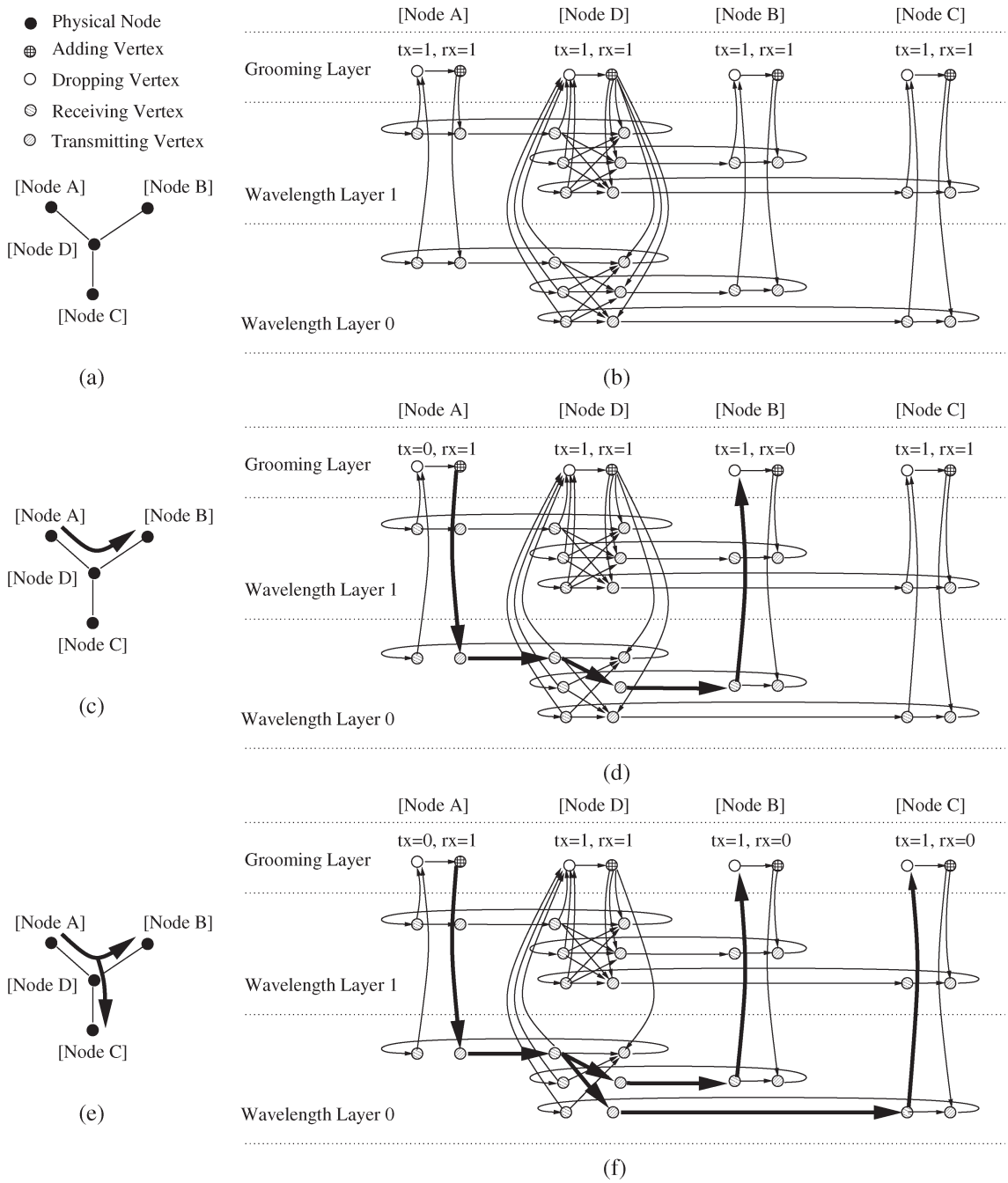

Fig. 3. Illustrative example for implementing the light-tree algorithm on the auxiliary graph. (a) Four-node physical network; (b) initial auxiliary graph with wavelength layer 0 and wavelength layer 1; (c) light-tree established from node A to node B on the physical network; (d) updated auxiliary graph after the light-tree in (c) is set up; (e) new branch extended from the light-tree; and (f) updated auxiliary graph after a new branch is extended, as in (e).

that end at the transmitting vertex of this pass-through edge and ii) delete the adding edge that ends at the transmitting vertex of this pass-through edge.

4) Check the number of transceivers at each node. If no free transmitter is available, delete all idle adding edges at the node. Similarly, if no receiver is available, delete all idle dropping edges at the node.
5) Iterate through all light-trees on which the request is carried. Deduct b from the residual capacity of wavelength-link edges on these light-trees.
6) Restore all wavelength-link edges deleted in 1).

Every time a request is terminated, the teardown routine operates as follows.

1) Remove the request's traffic demand from all light-trees on which the request is carried.
2) Tear down all branches that are no longer carrying effective traffic. If all branches on a light-tree are torn down, remove the entire light-tree.
3) Update the network state and the auxiliary graph accordingly. We omit the details due to space limitation.

The complexity of the 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 refer to the number of destinations in a request as group size, denoted by n. We obtain $|V| = O(dwN)$ and $|E| = O(dwN)$. Since the auxiliary graph is a very sparse graph, the complexity of Dijkstra's algorithm is $O(|E| \log |V|)$ by using the Fibonacci heap. The complexity of all other operations in the algorithm is equivalent to $O(|V|)$. Therefore, the complexities of the setup and teardown routines are $O(n(dwN)\log(dwN))$ and $O(ndwN)$, respectively.

We continue our illustrative example in Fig. 3 to demonstrate the operation of the setup routine. Suppose there is a new request $\{A, \{B\}\}$, demanding 1/4 of the capacity of a wavelength channel. Searching on the auxiliary graph shown in Fig. 3(b), a light-tree could be set up, as shown in Fig. 3(c), at wavelength 0 (at this time, the light-tree is a lightpath). The auxiliary graph, after the light-tree is set up, is shown in Fig. 3(d). Note that, as the light-tree uses one transmitter at node A, there is no free transmitter available at node A. Therefore, the adding edge to wavelength 1 is removed from the auxiliary graph at node A. Similarly, there is no free receiver at node B and the receiving edge from wavelength layer 1 is removed at node B. For node D, the transmitting vertex that is connected to the node B at wavelength layer 0 is taken by the light-tree. Therefore, the adding edge and all other pass-through edges that are going into this transmitting vertex are removed from the auxiliary graph.

Continue the example. Suppose, at a later time, there is a new request $\{A, \{B, C\}\}$, demanding 1/4 of the capacity of a wavelength channel. Running the setup routine on the auxiliary graph shown in Fig. 3(d), we can find a light-tree, as shown in Fig. 3(e), which is obtained by extending a new branch DC on the previous light-tree ADB. The auxiliary graph, after the new light-tree is set up, is shown in Fig. 3(f). Please note that the optical splitting capability of node D is kept in the auxiliary graph shown in Fig. 3(d) so that the existing light-tree gets a chance to extend a new branch. After the route is set up for the new request, the light-tree is shared by both requests.

It is worth noting that the proposed algorithm need not depend on the specific node architecture as in Section II. The only assumption is that nodes should support light-trees. The proposed algorithm also works for networks with tap-and-continue architecture [21] after some minor revisions.

D. Weight Assignment of Edges

In the proposed algorithm, the actual structure of a route tree is determined by the shortest path tree algorithm, which in turn depends on the weight assignment of edges. In the weight assignment, we make two decisions: 1) use as few wavelength links as possible for a route; and 2) prefer optical switching over electronic switching. The idea behind the decisions are twofold. Reducing consumption of wavelength links is very likely to decrease blocking, and optical switching is more cost effective than electronic switching.

We set the default weights of the five types of edges as in Table I. That is, the weight of a wavelength-link edge is 1.00; the weight of all other types of edges is 0.01. It is worth noting that, for the shortest path tree algorithm, it is the relative values, not the absolute values, of edge weights that make a difference. Therefore, other weight assignments are also possible only if the relative weight relationship of edges is kept. Since the wavelength-link edges are the dominant edges in the graph, the weight assignment obviously supports our first decision.

TABLE I
DEFAULT WEIGHT ASSIGNMENT OF EDGES IN THE AUXILIARY GRAPH

Edge Type	Weight Assignment
adding edge	0.01
dropping edge	0.01
pass-through edge	0.01
grooming edge	0.01
wavelength-link edge	1.00

For optical switching at a node, the route only traverses one pass-through edge at the node. On the other hand, for electronic switching, the route will first traverse a dropping edge, then a grooming edge, and finally an adding edge. By the weight assignment scheme, optical switching has a weight of 0.01, while electronic switching has a weight of 0.03. Therefore, the shortest path tree algorithm will prefer optical switching if such a connection is available.

After edges are used in some light-tree, we need to adjust their weights because the traffic injected into a light-tree will travel through all branches of the light-tree. Therefore, paths from the root of a light-tree to any leaf node of the light-tree consume the same amount of bandwidth. In order to take this effect into account, we set the weight of the adding edge of a light-tree to be the sum of weights of all branches on the light-tree and the weight of all other edges on the light-tree to be zero. After edges are released from a light-tree, their weights are restored to the default values.

E. Lightpath-Based Grooming Algorithm

For the purpose of performance comparison, we implement a lightpath-based algorithm using our proposed auxiliary graph model by changing the operation rule regarding adding edge and pass-through edge in 3) of the setup routine.

a) For an idle dropping edge, change its state to be busy and reduce the number of idle receivers by 1. Delete all pass-through edges that start from the receiving vertex of this dropping edge.
b) For an idle pass-through edge, change its state to be busy and 1) delete all other idle pass-through edges that ends at the transmitting vertex of this pass-through edge; and 2) delete the adding edge that ends at the transmitting vertex of this pass-through edge. Delete the dropping edge that starts at the receiving vertex of this pass-through edge.

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

V. NUMERICAL RESULTS AND ANALYSIS

In this section, we evaluate the performance of the algorithm by extensive simulation. We run the simulation on the network

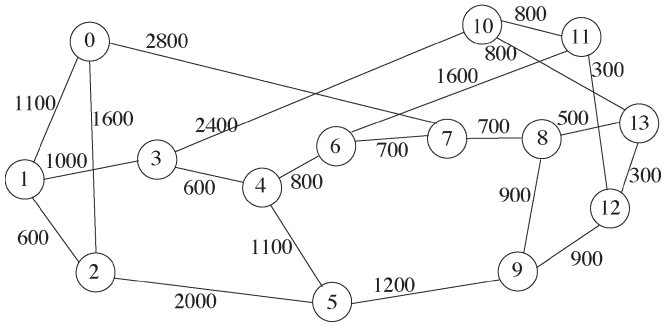

Fig. 4. Network with 14 nodes and 21 bidirectional links (distance in kilometers).

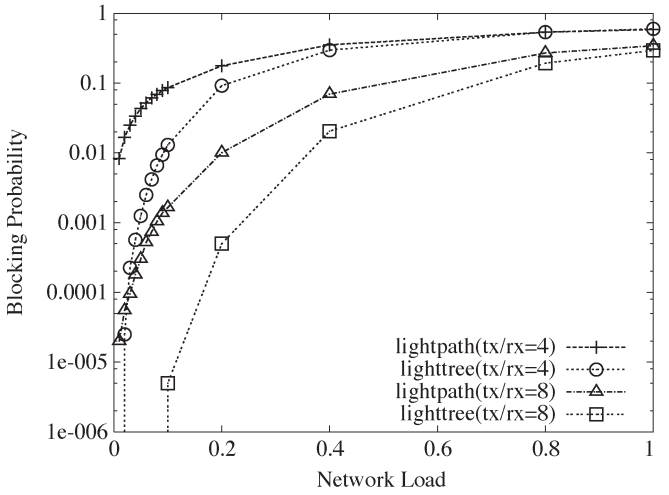

Fig. 5. Blocking probability with lightpath and with light-tree.

as in Fig. 4, in which each node has the same node degree of 3, and each link represents two fibers, one fiber in each direction. We make the following assumptions in our simulation.

1) All nodes have full optical splitting capability. Only some of the nodes are grooming hub nodes. There are no wavelength converters in the network.
2) There are four wavelengths for each fiber, with a capacity of OC-192/OC-48 per wavelength.
3) Traffic requests are generated and terminated dynamically. The traffic arrival is a Poisson process and the duration of requests is a negative exponential distribution.
4) All requests are multicast requests with a fixed number of destinations. The sources and destinations are uniformly distributed across the network. Each request demands a fixed bandwidth of OC-48.

We first compare the performance of the light-tree algorithm with the lightpath algorithm, with sparse grooming capability. We assume that nodes $\{1, 4, 8, 11\}$ are grooming hub nodes that have unlimited grooming capability, while the rest nodes have no grooming capability. This choice is made to ensure that each node is adjacent to at least one grooming hub node. We further assume that each multicast request has 4 destination members.

Fig. 5 depicts the destination blocking probability as a function of the network load with the two algorithms. As seen from the figure, the light-tree algorithm significantly outperforms the lightpath algorithm, either in the case with 4 transceivers

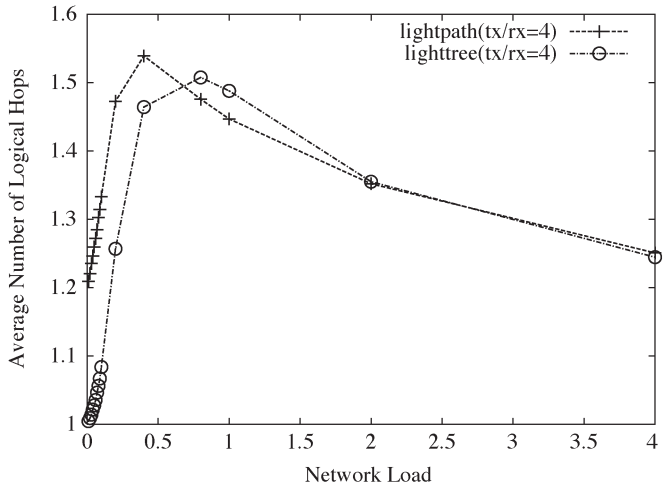

Fig. 6. Average number of logical hops with lightpath and with light-tree.

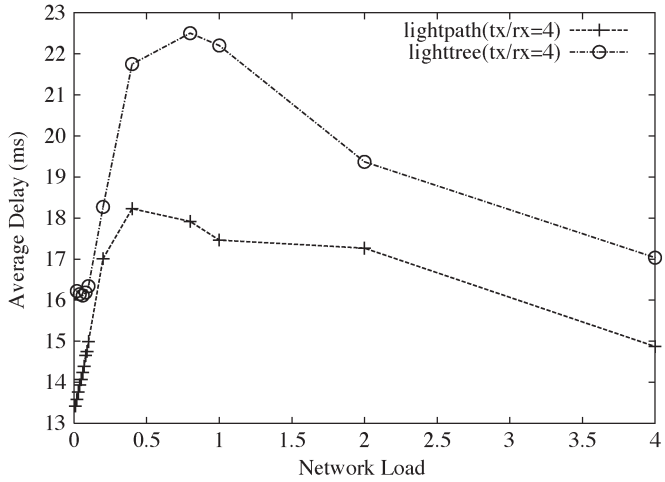

Fig. 7. Average delay with lightpath and with light-tree.

or in the case with 8 transceivers. With the introduction of optical multicast capability, more destinations could be reached through optical splitting at intermediate nodes in the light-tree algorithm. However, the lightpath algorithm cannot take advantage of the optical multicast capability. When comparing the two cases with different number of transmitters and receivers, it can be seen that, with more transceivers at each node, the improvement of the light-tree algorithm over the lightpath algorithm is more significant. Fig. 6 illustrates the average number of logical hops of the two algorithms. The light-tree algorithm has much less average number of logical hops than the lightpath algorithm, which means that light-tree-based approach has less demand on electronic grooming processing. The reason is that light-trees also increase the reach of a single optical hop than lightpath. However, this property of light-trees results in fewer freely available wavelength links in the network such that a request has a higher chance to be carried on established light-trees (either by using existing light-trees or by extending existing light-trees). This results in a slightly higher average propagation delay, which is shown in Fig. 7. If we take into account the delay due to traffic grooming in intermediate nodes, the delay difference between the two approaches is even less because the light-tree model has less grooming, as previously mentioned.

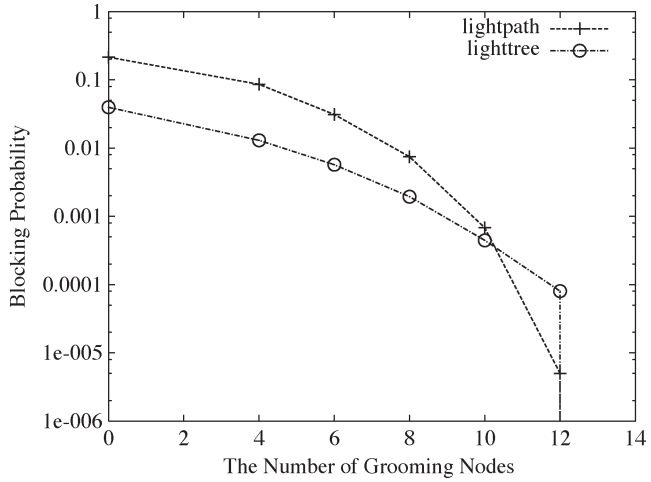

Fig. 8. Blocking probability with variable grooming nodes in the network.

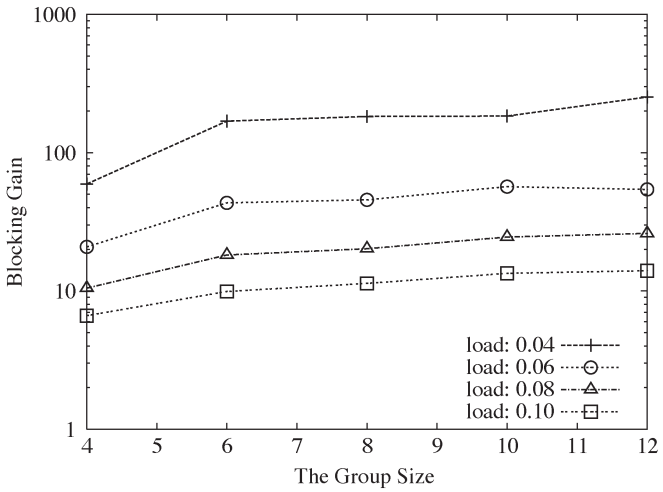

Fig. 9. Blocking probability gain: light-tree over lightpath.

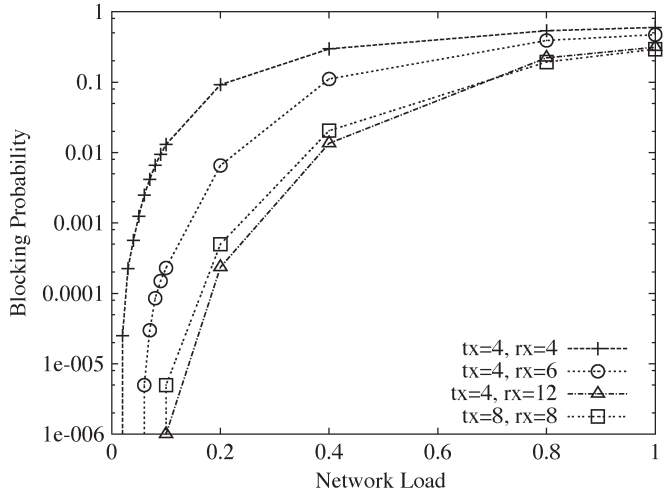

Fig. 10. Blocking probability with variable transceivers per node.

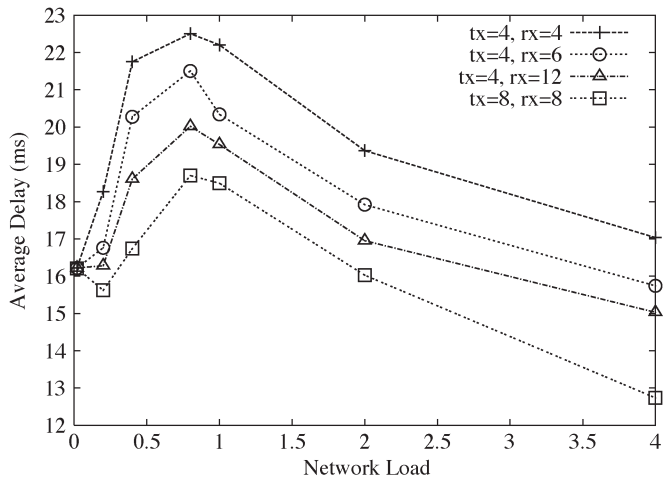

Fig. 11. Average delay with variable transceivers per node.

In the above simulation, we assume that there are only 4 nodes with grooming capability. Next, we investigate the performance trend with a variable number of grooming hub nodes in the network. This is shown in Fig. 8, with a network load of 0.1. It is clear that increasing the number of grooming hub nodes can significantly reduce the blocking probability for the light-tree algorithm as well as for the lightpath algorithm. This is because, with more grooming hub nodes, the algorithm is more likely to set up multihop routes for the requests.

It is somewhat interesting to note that, when the number of grooming hub nodes is more than 10, the lightpath algorithm has better performance than the light-tree algorithm. When most of nodes have grooming capability, there is almost no blocking due to the lack of grooming capability. However, the light-tree topology results in more bandwidth being wasted than the lightpath topology. There exists a tradeoff between these two factors in our algorithm. If our algorithm could be more intelligent, we would expect that a light-tree-based algorithm would always have better performance than a lightpath-based algorithm, since light-trees are a superset of lightpaths. If we could find the optimal solution in both cases, a light-tree-based solution should be at least as good as a lightpath-based solution. However, for dynamic traffic, it is very hard to find the optimal solution in either case.

Next, we investigate the effect of multicast group size on the performance of the two algorithms. Fig. 9 shows the blocking gain of the light-tree algorithm over the lightpath algorithm under different group sizes. We define the blocking gain as the ratio between the blocking probability of the lightpath model and that of the light-tree model. It is observed that the blocking gain increases with the increase of the multicast group size, especially at low network load. This is expected since, with more destinations in a multicast request, light-trees are more efficient in saving bandwidth.

Finally, we study the MDTGA light-tree algorithm in cases with variable number of transceivers. Fig. 10 clearly shows that, with an increase on the number of transceivers ($\mathrm{tx}/\mathrm{rx} = 4$ versus $\mathrm{tx}/\mathrm{rx} = 8$), there is a dramatic decrease in the blocking probability. With more transceivers, the blocking due to the lack of transceivers is reduced. Furthermore, with more transceivers, the algorithm is more likely to set up more light-trees that occupy fewer wavelength links, which further reduces the blocking due to limited bandwidth resources.

In general, as a light-tree takes one transceiver and more than one receiver, it would be better to have more receivers

than transmitters. This result is also shown in Fig. 10. Increasing the number of receivers, while keeping a fixed number of transmitters, can also greatly reduce the destination blocking probability. A more interesting result can also be observed in Fig. 10. The configuration with 4 transmitters and 12 receivers has better performance than the configuration with 8 transmitters and 8 receivers. This observation is of practical importance. We can greatly improve the performance by only increasing the number of receivers at each node instead of increasing the number of both transmitters and receivers. Since receivers are less expensive than transmitters, this will greatly reduce the cost of the network. Such asymmetric node architecture could be implemented by our proposed architectures. Fig. 11 shows the delay with variable transceivers per node. The difference in delay is only marginal when the load is low. We also observe that the lightpath-based grooming algorithm has lower blocking probability even when only the number of receivers is increased (not shown due to space limitation). Although there is exactly one transmitter and one receiver for each lightpath, it may happen that a lightpath cannot be established because of lack of receivers.

VI. Conclusion

In this paper, we study the online multicast traffic grooming problem in WDM networks with sparse grooming capability. We present a grooming algorithm, MDTGA, which adopts dynamically changing light-trees as the building block and is implemented by using an auxiliary-graph model. Compared with the lightpath algorithm for the problem, the light-tree algorithm has much better performance in terms of blocking probability, with only a slight increase in delay. We observe that an asymmetric node architecture (with more receivers than transmitters) is more cost effective for multicast traffic grooming than the regular symmetric node architecture (with equal number of receivers and transmitters).

References

[1] S. Paul, *Multicasting on the Internet and Its applications*. Boston, MA: Kluwer, 1998.

[2] B. Mukherjee, *Optical Communication Networks*. New York: McGraw-Hill, 1997.

[3] 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.

[4] L. H. Sahasrabuddhe and B. Mukherjee, "Light trees: Optical multicasting for improved performance in wavelength routed networks," *IEEE Commun. Mag.*, vol. 37, no. 2, pp. 67–73, Feb. 1999.

[5] G. Sahin and M. Azizoglu, "Multicast routing and wavelength assignment in wide area networks," in *Proc. SPIE*, Boston, MA, 1998, vol. 3531, pp. 196–208.

[6] R. Malli, X. Zhang, and C. Qiao, "Benefit of multicasting in all-optical networks," in *Proc. SPIE Conf. All-Optical Networking*, Boston, MA, Nov. 1998, vol. 3531, pp. 209–220.

[7] R. Libeskind-Hadas and R. Melhen, "Multicast routing and wavelength assignment in multihop optical networks," *IEEE/ACM Trans. Netw.*, vol. 10, no. 5, pp. 621–629, Oct. 2002.

[8] H. Madhyastha, N. Srinivas, G. Chowdhary, and C. Murthy, "Grooming of multicast sessions in WDM ring networks," in *Proc. Optical Networking and Communications (OptiComm)*, Dallas, TX, Oct. 2003, pp. 1–12.

[9] A. Billah, B. Wang, and A. Awwal, "Multicast traffic grooming in WDM optical mesh networks," in *IEEE Global Telecommunications Conf. (GLOBECOM)*, San Francisco, CA, Dec. 2003, pp. 2755–2760.

[10] A. Kamal and R. Ul-Mustafa, "Multicast traffic grooming in WDM networks," in *Proc. Optical Networking and Communications (OptiComm)*, Dallas, TX, Oct. 2003, pp. 25–36.

[11] D. Yang and W. Liao, "Design of light-tree based logical topologies for multicast streams in wavelength routed optical networks," in *Proc. IEEE Information Communications (INFOCOM)*, San Francisco, CA, Apr. 2003, pp. 32–41.

[12] F. Hwang, D. Richards, and P. Winter, *The Steiner Tree Problem*. Amsterdam, The Netherlands: Elsevier, 1992.

[13] A. Chiu and E. Modiano, "Traffic grooming algorithms for reducing electronic multiplexing costs in WDM ring networks," *J. Lightw. Technol.*, vol. 18, no. 1, pp. 2–12, Jan. 2000.

[14] P. Wan, G. Calinescu, L. Liu, and O. Frieder, "Grooming of arbitrary traffic in SONET/WDM BLSRs," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 1995–2003, Oct. 2000.

[15] X. Zhang and C. Qiao, "An effective and comprehensive approach for traffic grooming and wavelength assignment in SONET/WDM rings," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 608–617, Oct. 2000.

[16] K. Zhu and B. Mukherjee, "Traffic grooming in an optical WDM mesh network," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 1, pp. 122–133, Jan. 2002.

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

[18] H. Zhu, H. Zang, K. Zhu, and B. Mukherjee, "Dynamic traffic grooming in WDM mesh networks using a novel graph model," in *Proc. IEEE Global Telecommunications Conf. (GLOBECOM)*, Taipei, Taiwan, Nov. 2002, vol. 3, pp. 2681–2685.

[19] W. S. Hu and Q. J. Zeng, "Multicasting optical cross connects employing splitter-and-delivery switch," *IEEE Photon. Technol. Lett.*, vol. 10, no. 7, pp. 970–972, Jul. 1998.

[20] J. Leuthold and C. Joyner, "Multimode interference coupleers with tunable power splitting ratios," *J. Lightw. Technol.*, vol. 19, no. 5, pp. 700–707, May 2001.

[21] M. Ali and J. S. Deogun, "Cost-effective implementation of multicasting in wavelength-routed networks," *J. Lightw. Technol.*, vol. 18, no. 12, pp. 1628–1638, Dec. 2000.

Xiaodong Huang received the B.S. degree in electrical engineering and the M.S. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 1995 and 1998, respectively. He is currently working towards the Ph.D. degree in computer science at the University of Texas at Dallas.

His research interests include optical networks and wireless networks, with a focus on multicast and traffic grooming.

Farid Farahmand was born in Tehran, Iran. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Texas at Dallas, in 1993, 1995, and 2005, respectively.

He started his professional career in 1993, when he joined Alcatel USA as a Hardware Design Engineer. In January 2000, he moved to Alcatel Corporate Research and was involved with the development of terabit optical routers. In 2005, he became the Technical Advisor at Emtec Solution in Dallas, TX. His research interests are in high-speed packet switching and all-optical networks focusing on architecture and protocol designs for optical burst-switched networks.

Jason P. Jue (M'99–SM'04) received the B.S. degree in electrical engineering from the University of California, Berkeley, in 1990, the M.S. degree in electrical engineering from the University of California, Los Angeles, in 1991, and the Ph.D. degree in computer engineering from the University of California, Davis, in 1999.

He is an Associate Professor in the Department of Computer Science at the University of Texas at Dallas. His research interests include optical networks, network control and management, and network survivability.