Multicast Traffic Grooming in Wavelength-Routed WDM Mesh Networks Using 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 online multicast traffic grooming problem in wavelength-routed WDM mesh networks with sparse grooming capability. We develop a *multicast dynamic light-tree grooming algorithm (MDTGA)* 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; 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 will forward the message to multiple destinations. There are increasing demands for multicast applications, such as real time video conferencing, stock information distribution, and online multi-user 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 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*.

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 which 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 singal source and multiple destinations. Like the lightpath, there is no OEO conversion at any intermediate node on a light-tree.

Using 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] [6] [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 and how to set up their routes and to assign wavelength. This problem is known as the *multicast traffic grooming* problem.

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 SONET over WDM ring networks under static traffic scenarios, where the traffic is known in advance, such as in [8] [9] [10]. Recently, there has been some work on unicast traffic grooming in WDM mesh networks. In [11], the authors formulate the traffic grooming problem as an Integer Linear Programming problem and propose several heuristic algorithms. A generic graph model is presented in [12] to support traffic grooming. This model was further applied to the dynamic traffic grooming scenario [13], in which traffic requests dynamically arrive (or leave) and lightpaths are dynamically set up (or torn down).

Very recently, the multicast traffic grooming problem has begun to attract some research attention [14] [15] [16]. 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 [17]. The authors proposed a unified Integer Linear Programming formulas for static traffic, including unicast,

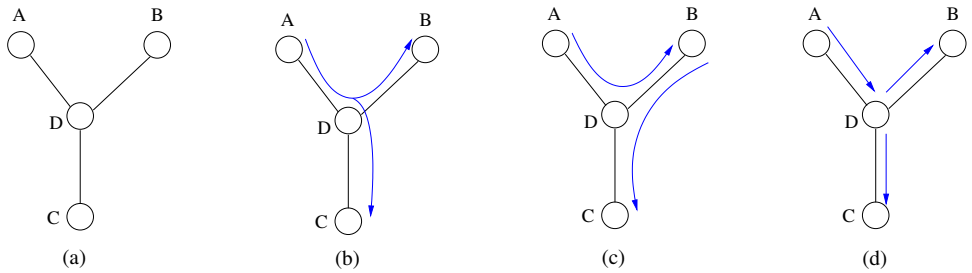

Fig. 1. Multicast routing with traffic grooming: an illustrated example. (a) A part of a network; (b) light-tree based routing; (c) lightpath based routing; (d) link by link routing.

multicast, and mixed traffic. However, to the best of our knowledge, there is no study published on dynamic multicast traffic grooming.

Without traffic grooming, the multicast routing problem can be converted to the well-known *Steiner tree problem*, which is NP-complete [18]. With multicast traffic grooming, the routing and wavelength assignment problem becomes more involved, which makes the existing algorithms for regular multicast routing and wavelength assignment invalid for our new 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 sub-wavelength bandwidth demand, in which node A is the source and node 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 (d). If the bandwidth demand is at the wavelength level, this is the optimal solution. However, with the sub-wavelength bandwidth demand, it is not necessarily the optimal solution. For example, suppose there is another sub-wavelength traffic request from A to B. If the new traffic is carried on the light-tree as in Fig. 1 (d), 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 $\{A, \{B, C\}\}$ 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 (b), 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. This is the essential difference between multicast routing with traffic grooming and without traffic grooming. 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 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; 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. 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]. Fig. 2 (a) is a variance of the SaD switch, using configurable optical splitters [20]. A configurable optical splitter is an active component which 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 non-blocking characteristic of SaD-based cross-connects ensures that no existing connection will be interrupted as light-trees change dynamically. Therefore, there is no blocking of traffic due to optical switching capability.

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

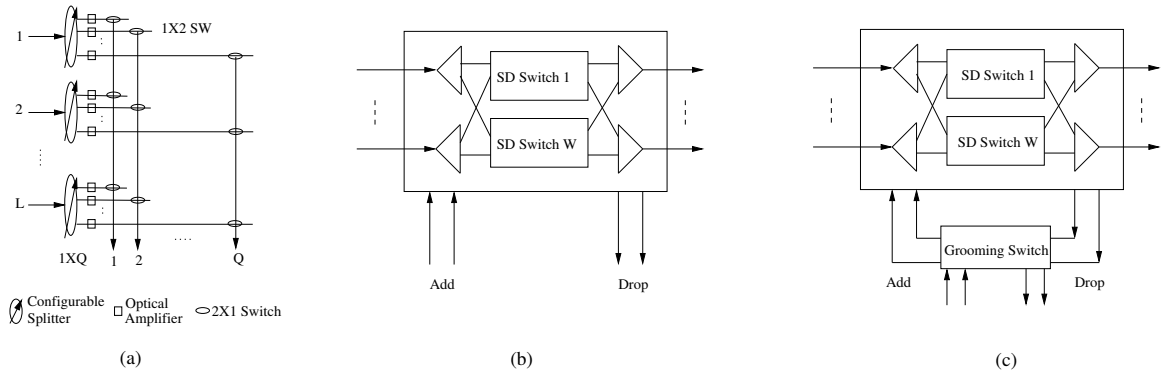

Fig. 2. Node architectures. (a) SaD switch with configurable optical splitter; (b) Node architecture with local grooming capability (MLG-OXC); (c) Node architecture with full grooming capability (MFG-OXC).

traffic with pass-through traffic; 3) pass-through traffic with pass-through traffic.

The first grooming scheme is actually the same as the traditional multiplexing and de-multiplexing function. In the second grooming scheme, the pass-through traffic will first be de-multiplexed and then will be multiplexed with local traffic. Usually, an electronic grooming switch at sub-wavelength 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.

For simplicity, we assume that the second grooming scheme need 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 scheme 2 and scheme 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.

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 equiped with the MFG-OXC and the rest of the nodes are equiped with the MLG-OXC. We refer to the nodes with the MFG-OXC architecture as *grooming hub nodes*.

III. Problem statement

We formulate the multicast traffic grooming problem as follows:

- Given:
  - The topology of the physical network
  - The number of wavelengths on each fiber link
  - The number of transceivers at each node
  - The architecture of each node
- Under the assumptions:
  - Only a small part of the nodes are grooming hub nodes.
  - None of nodes have wavelength converters.
  - All transceivers are tunable to any wavelength.
  - Requests are multicast traffic with sub-wavelength bandwidth demand. Requests 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 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 setup.
- With the constraints:
  - The number of wavelengths per fiber
  - The number of transmitters per node
  - The number of receivers per node
- Find:
  - Routes for arrival requests
- To minimize:
  - 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 our discuss.

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. The route from the source to a destination in the request may go through multiple light-trees, which may be at different wavelength layer. 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 edge*s and *dropping edge*s. 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.

- *adding vertex*: 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.
- *dropping vertex*: 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.
- *transmitting vertex*: 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.
- *receiving vertex*: 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. All vertices in the graph are independent of the state of the network.

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

- *adding edge*: from the adding vertex to a transmitting vertex within a node.
- *dropping edge*: from a receiving vertex to the dropping vertex within a node.
- *pass-through edge*: from a receiving vertex to a transmitting vertex within a node.
- *grooming edge*: from the dropping vertex to the adding vertex within a node if the node is a grooming hub node.
- *wavelength-link edge*: from a transmitting vertex of a node to a receiving vertex of a neighboring node, at the same wavelength layer.

An edge can either be in use by a light-tree or can be freely available. 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 available capacity of an edge which 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. It is the relative values, not the absolute values, of edge weights that makes the difference. We set the edge weights as follows: the weight of a wavelength-link edge is 1; the weight of all other types of edges is 0.01. This weight assignment means that we prefer optical switching over electronic switching. It also means that we prefer using less number of wavelength links for a route, since the wavelength-link edges are the dominate edges in the graph.

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. The traffic injected into a light-tree will travel through all branches on 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 taking account of this effect, 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 we set the weight of all other edges on the light-tree to be zero. The idea is to minimize the total number of wavelength links taken by a route.

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.

- 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. At last, 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.
- 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.

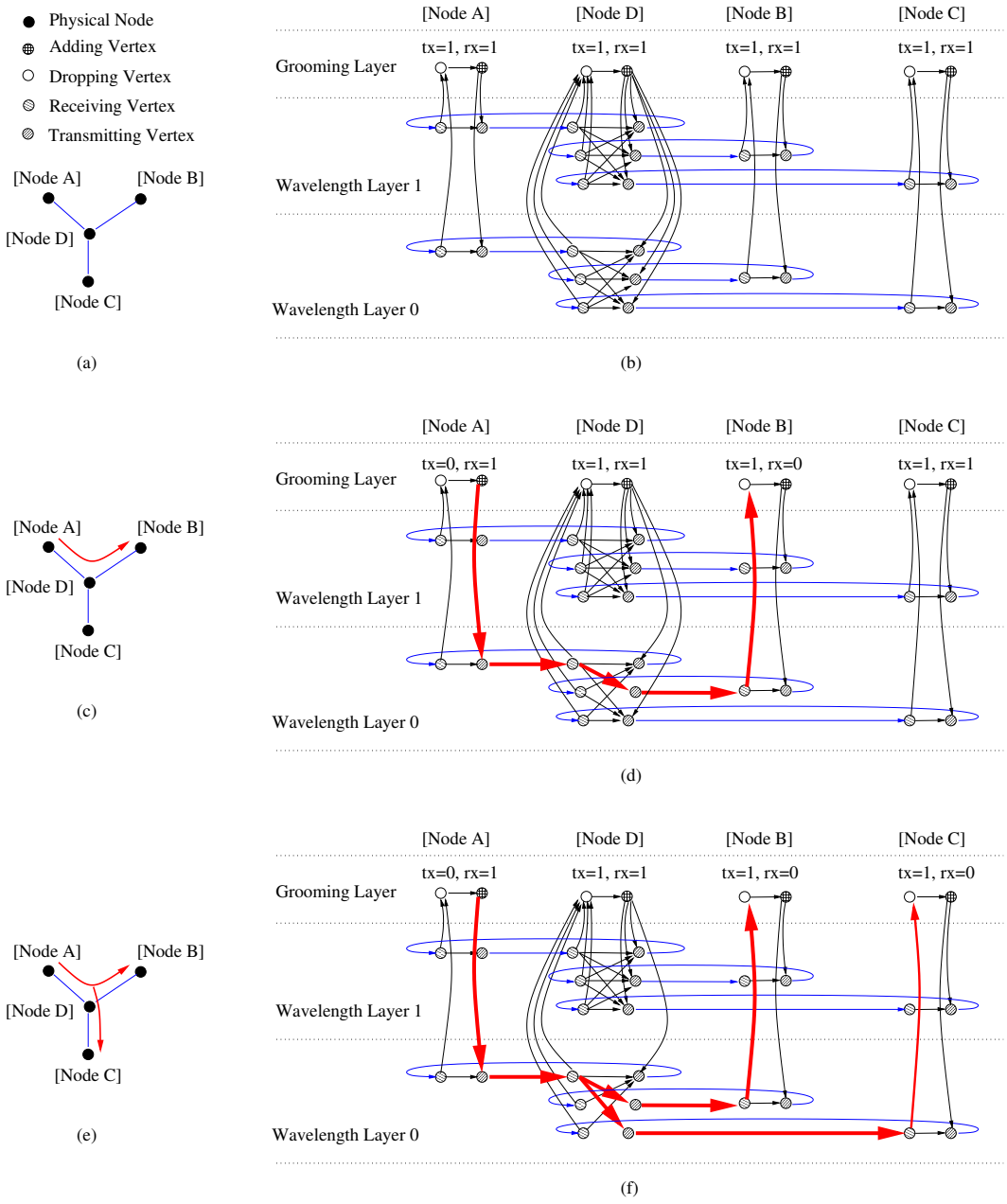

Fig. 3. An illustrative example for implementing the light-tree algorithm on the auxiliary graph. (a) 4-node physical network; (b) the initial auxiliary graph with two wavelength layer i and j; (c) a light-tree established from node A to node B on the physical network; (d) the updated auxiliary graph after the light-tree in (c) is set up; (e) a new branch is extended from the light-tree; (f) the updated auxiliary graph after a new branche is extended as in (e).

- 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 bi-directional. There are 2 wavelengths per fiber and there are 1 transmitter and 1 receiver at each node. Fig. 3 (b) shows the initial auxiliary graph. Let's take node D as an example. There are 3 transmitting ports and 3 receiving ports at node D. Hence, for node D, there are 3 transmitting vertices and 3 receiving vertices at each wavelength layer. Even though there is only 1 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 port to any transmitting port 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 [18]. 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 [18]. The algorithm starts the route tree with the source node. It continues to include the destination which 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 can not simply run a Stainer tree algorithm on the auxiliary graph to find the route tree for a multicast request, since nodes in a network may only have 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 $Req(s, D, b)$, where s is the source, $D = \{d_1, d_2, ..., d_n\}$ is a set of the destination nodes, and b is the bandwidth demand of the request.

- Step 1: Check the residual capacity of each *wavelength-link edge* on each layer and delete it if its residual capacity is less than b. For each destination in the D, repeat Step 2 to Step 4.
- Step 2: Search the shortest paths on the auxiliary graph from the *adding vertex* at the source node to the *dropping vertices* of all remaining destinations. Choose the nearest destination which can be reached by the source. If no such destination exists, go to Step 5; otherwise, continue to Step 3.
- Step 3: Iterate through vertices and edges on the shortest path to establish the route tree on the auxiliary graph.
  - For idle transmitting vertices and receiving vertices, change their state to be *busy*.
  - For an *idle* adding edge, change its state to be *busy* and do the following operation: 1) reduce the number of idle transmitters by 1; 2) delete all idle pass-through edges which is ended at the transmitting vertex of this adding edge.
  - For an *idle* dropping edge, change its state to be *busy* and reduce the number of idle receivers by 1.
  - For an *idle* pass-through edge, change its state to be *busy* and 1) delete all other idle pass-through edges which is ended at the transmitting vertex of this pass-through edge; 2) delete the adding edge which is ended at the transmitting vertex of this pass-through edge.
- Step 4: Check the number of transceivers at each node. If no 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.
- Step 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.
- Step 6: Restore all *wavelength-link edge*s deleted in Step 1.

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

- Step 1: Remove the request's traffic demand from all light-trees on which the request is carried.
- Step 2: Tear down all branches which are no longer carrying effective traffic. If all branches on a light-tree are torn down, remove the entire light-tree.
- Step 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. Let n be group size of the requests. 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* routine and *teardown* routine 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. Search the route on the auxiliary graph shown in Fig. 3 (b), a light-tree can be set up as shown in Fig. 3 (c) at wavelength 0 (actually, 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 1 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 which is connected the node B at wavelength layer 0 is taken by the light-tree. Therefore, that all other pass-through edges going to this transmitting vertex is removed from the auxiliary graph.

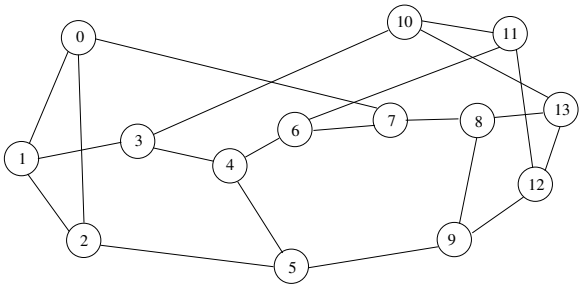

Fig. 4. A network with 14 nodes and 21 bi-directional links.

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. Run 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.

D. Lightpath-based grooming algorithm

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

- 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 which starts from the receiving vertex of this dropping edge.
- For an *idle* pass-through edge, change its state to be *busy* and 1) delete all other idle pass-through edges which is ended at the transmitting vertex of this pass-through edge; 2) delete the adding edge which is ended at the transmitting vertex of this pass-through edge. Delete the dropping edge which 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, MDTA 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 as in Fig. 4, in which each node has the same node degree of 3 and each link represents 2 fibers, one fiber in each direction. We make the following assumptions in our simulation:

- All nodes have full optical splitting capability. Only some of the nodes are grooming hub nodes. There are no wavelength converters in the network.

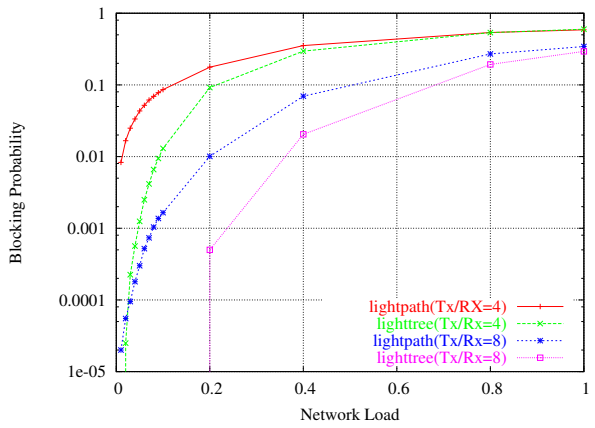

Fig. 5. Blocking probability with lightpath and with light-tree.

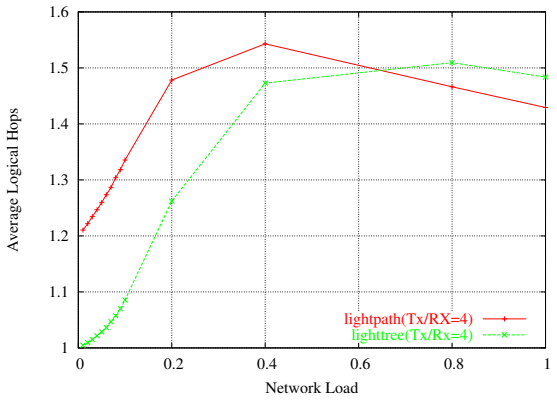

Fig. 6. Average logical hops with lightpath and with light-tree.

- There are 4 wavelengths for each fiber with a capacity of OC-192 per wavelength.
- Traffic requests are generated and terminated dynamically. The traffic arrival is a Poisson process and the duration of requests is a negative exponential distribution.
- All requests are multicast requests with a fixed number of destinations. The sources and destinations are evenly 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 $\{1, 4, 8, 11\}$ have unlimited grooming capability, while the rest nodes have no such capability. This choice is made to ensure that each node is adjacent to at least one grooming hub node. We further assume that each node has 4 transceivers.

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

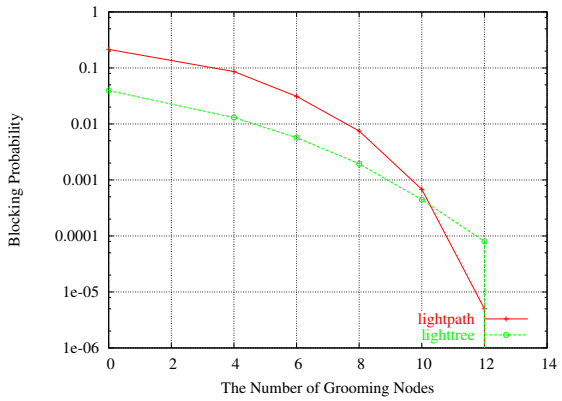

Fig. 7. Blocking probability with variable grooming nodes in the network.

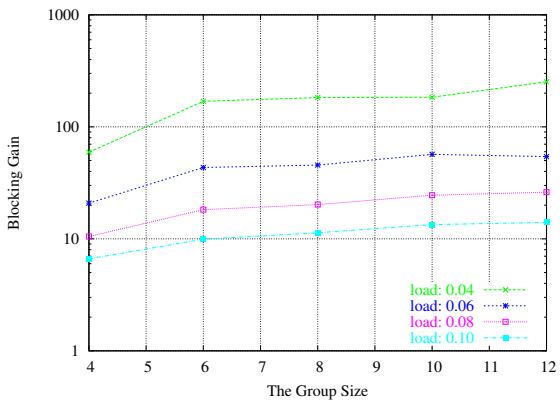

Fig. 8. Blocking probability gain: light-tree over lightpath.

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 logical hops of the two algorithms. The light-tree algorithm has much less average number of logical hop than the lightpath algorithm. The reason is that light-trees also increase the reach of a single optical hop than lightpath.

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. 7. It is clear that increasing the number of grooming hub nodes can reduce the destination 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 multi-hop 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, light-tree topology results in more bandwidth being wasted than the lightpath topology. There exists a trade-off 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 the a lightpath based algorithm, since light-trees are a superset of lightpaths. If we can 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. 8 shows the *blocking gain* of the light-tree algorithm over the lightpath algorithm under different group sizes. 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.

We next investigate the effect of the number of transceivers on the relative performance of the light-tree algorithm. Fig. 9 clearly shows that, with an increase on the number of

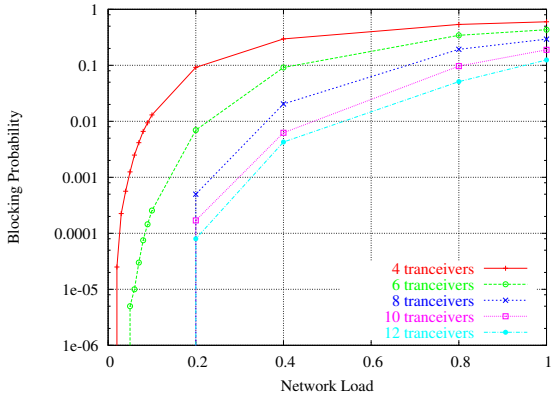

Fig. 9. Blocking probability with variable transceivers per node.

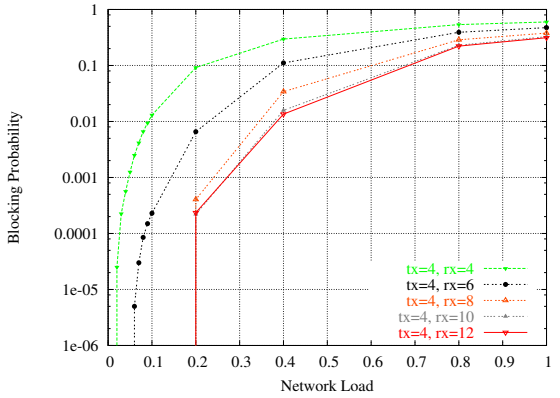

Fig. 10. Blocking probability with variable receivers per node.

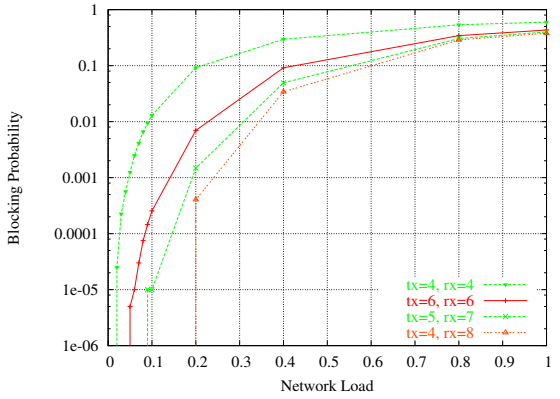

Fig. 11. Blocking probability with variable receivers per node.

transceivers, there are dramatic improvement in the blocking probability. With more transceivers, the blocking due to the lack of transceivers is reduced. Further more, with more transceivers, the algorithm is more likely to set up more light-trees which 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 receivers, it would be better to have more receivers than transmitters. This result is shown in Fig. 10. Increasing the number of receivers, while keeping a fixed number of transmitter, can also greatly reduce the destination blocking probability. A more interesting result is revealed in Fig. 11. The configuration with 5 transmitters and 7 receivers produces better performance than the configuration with 6 transmitters and 6 receivers. The configuration with 4 transmitters and 8 receivers provides even better performance. 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.

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 a auxiliary graph model. Compared with the lightpath algorithm for the problem, the light-tree algorithm has much better performance in terms of blocking probability and average number of logical hops.

References

[1] S. Paul, "Multicasting on the Internet and its applications," *Kluwer Academic*, Boston, 1998.

[2] B. Mukherjee, "Optical communication networks," *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, Communications Magazine*, vol. 37, no. 2, pp. 67-73, Feb. 1999.

[5] G. Sahin and M. Azizoglu, "Multicast Routing and Wavelength Assignment in Wide Area Networks," *Proc. SPIE*, vol. 3531, pp. 196-208, 1998.

[6] R. Malli, X. Zhang, and C. Qiao, "Benefit of Multicasting in All-optical Networks," *SPIE Proc. Conf. All-optical Networking*, vol. 3531, pp. 209-220, Nov. 1998.

[7] R. Libeskind-Hadas and R. Melhen, "Multicast routing and wavelength assignment in multihop optical networks," *Networking, IEEE/ACM Transactions on*, vol. 10 , issue 5, pp. 621-629, Oct. 2002.

[8] A. Chiu and E. Modiano, "Traffic grooming in algorithms for reducing electronic multiplexing costs in WDM ring networks," *J. Lightwave Technology*, vol. 18, pp. 2-12, Jan. 2000.

[9] P. Wan, G. Calinescu, L. Liu and O. Frieder, "Grooming of arbitrary traffic in SONET/WDM BLSRs," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 1995-2003, Oct. 2000.

[10] X. Zhang and C. Qiao, "An effective and comprehensive approach for traffic grooming and wavelength assignment in SONET/WDM rings," *IEEE/ACM Trans. Networking*, vol 8, pp. 608-617, Oct. 2000.

[11] 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.

[12] 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.

[13] 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.

[14] H. Madhyastha, N. Srinivas, G. Chowdhary, and C. Murthy, "Grooming of Multicast Sessions in WDM Ring Networks," *OptiComm2003*, pp.1-12, Nov. 2003.

[15] A. Billah, B. Wang, A. Awwal, "Multicast Traffic Grooming in WDM Optical Mesh Networks", *GlobeCom2003*, pp.2755-2760, Nov. 2003.

[16] A. Kamal and R. Ul-Mustafa, "Multicast traffic grooming in WDM networks," *OptiComm2003*, pp.25-36, Nov. 2003.

[17] D. Yang and W. Liao, "Design of light-tree based logical topologies for multicast streams in wavelength routed optical networks," *Proc. IEEE INFOCOM 2003*, Apr. 2003.

[18] F. Hwang, D. Richards, and P. Winter, "The Steiner tree problem," *Elsevier Science Publishers*, 1992.

[19] 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.

[20] J. Leuthold and C. Joyner, "Multimode interference coupleers with tunable power splitting ratios," *IEEE Journal of Lightwave Technology*, vol. 19, no. 5, pp. 700-707, May 2001.