Avoidance of Multicast Incapable Branching Nodes for Multicast Routing in WDM Networks

Fen ZHOU, Miklós MOLNÁR
Department of Computer Science
INSA de Rennes / IRISA
Rennes, FRANCE

Bernard COUSIN
University of Rennes I / IRISA
Rennes, FRANCE

***Abstract*— Although many multicast routing algorithms have been proposed in order to reduce the total cost in WDM optical networks, the link stress and delay are two parameters which are not always taken into consideration. This paper proposes a novel wavelength routing algorithm, which tries to avoid the multicast incapable branching nodes (*MIB*, branching nodes without splitting capability) to diminish the link stress for the shortest path based multicast tree and maintains good parts of the shortest path tree to reduce the end-to-end delay. Firstly a DijkstraPro algorithm with priority assignment and node adoption is introduced to produce a shortest path tree with up to 38% fewer *MIB* nodes, and then critical articulation and deepest branch heuristics are used to process the *MIB* nodes. Finally distance based reconnection algorithm is proposed to create the multicast tree or forest.**

***Keywords*- Multicast Routing, Light Splitting, Light-Tree Computation, WDM Networks**

I. INTRODUCTION

Multicast is a very efficient way for one-to-many or many-to-may communication. A multicast session typically involves a source and a set of destinations. In traditional data networks, usually, a multicast tree rooted at the source is constructed with branches spanning all the destinations to accommodate a multicast session. In order to be able to multicast data in WDM optical networks, optical switch needs to have splitting capability. Note that optical switches with splitting capability are always much more expensive to build than those without. Consequently, only a few nodes can support splitting. Hence, multicast routing in WDM optical networks is greatly different from that in traditional data networks and one must consider the constraint on splitting capability of nodes in a practical optical network. Due to these physical constraints, supporting multicast routing in all optical network is a challenging work.

For multicast routing in WDM optical networks, many multicast tree formation algorithms have been proposed to reduce the total cost (i.e. the total number of wavelength channels), but the link stress and delay in the network are also two very important factors, which should be taken into account of, especially for the time sensitive and bandwidth intensive multicast applications such as HDTV, VOIP and Video Conference. It is known that if the data is transmitted via shortest path from the source to the destination in a WDM optical network, then the delay is minimal. Unfortunately, most of the nodes cannot support slitting and the splitting nodes are very rare in optical networks due to its high cost. If most of destinations communicate with the source through their shortest path, there is a big probability that many of the shortest paths will traverse the same node without splitting capability. Then, more wavelengths should be utilized and the link stress will be even high. From the point of view of link stress, shortest paths cannot be used for all the destinations, and some destinations could find longer paths to the source. While, from the point of view of delay, longer paths should not be used for all destinations also. So, a tradeoff should be found between the link stress and the delay in order to obtain the best general performance.

In this paper, a multicast routing algorithm considering sparse light splitting, which tries to avoid the multicast incapable branching nodes in the multicast light-tree, is proposed to resolve the wavelength routing problem in WDM optical networks. It aims to reduce both the link stress and the delay. The significant aspects of this paper lie at: (i) DijkstraPro algorithm with priority assignment and node adoption is introduced to construct a shortest path tree with fewer multicast incapable branching nodes; (ii) critical articulation and deepest branch heuristics are used to process the *MIB* nodes with the purpose of reducing both the link stress and delay; (iii) distance based reconnection algorithm is proposed to create the multicast forest with smaller delay while keeping the same link stress and cost.

II. RELATED WORK

The hardness of the multicast routing in WDM networks with sparse light splitting capability has been discussed in many papers [1~8] and different algorithms has been proposed. There are mainly three main categories according to the routing approach they employ: Source-Based Routing (e.g. Reroute-to-Source, Reroute-to-Any and Member-First [1]), Steiner-Based Routing (e.g. Member-Only [1] and Virtual-Source Capacity-Priority algorithm [2]) and Core-Based Routing (e.g. Virtual Source based algorithm [3]) [4]. Essentially, the Source-Based Routing approach constructs the multicast tree by connecting the source to each destination individually using the appropriate shortest path in order to minimize the per source–receiver path cost. The objective of the Steiner-Based Routing schemes, however, is to minimize the overall cost of the multicast tree. The core structure, connects a subset of nodes, called core nodes, which have both light-splitting and wavelength conversion capacities. The multicast session is then established with the help of this core structure [3, 4]. As far as we know from literature [1], in the

all optical network with sparse splitting and without wavelength conversion, Member-Only algorithm can get the approximate minimal cost, while Reroute-to-Source algorithm yields the optimal delay.

In Reroute-to-Source, firstly a multicast tree is generated to span all the destinations, by pruning the shortest path tree built by the Dijkstra algorithm. Then, it checks the light splitting capability for each node in the multicast tree. If a node is a branching node with splitting capability, then no modification is needed. But, if it is a multicast incapable branching node (i.e. it has at least 2 direct children while has no splitting capability), then only one direct child could be kept, which is chosen arbitrarily. And all the other children should be connected to the source through the reverse shortest path each with a different wavelength. It is obvious that the average delay of the Reroute-to-Source is minimal. However, the stress of the link is very high; because different downstream branches of a multicast incapable branching node should be connected to the source using the same shortest path on different wavelengths. In fact, there may find some longer paths to reach the source using the same wavelength.

In Reroute-to-Any, firstly it also constructs a multicast tree by pruning unnecessary nodes in the shortest path tree for all the nodes in the network. Then, for each multicast incapable branching node, one downstream branch is kept and the others are cut. Finally, the cut destinations can be connected to multicast tree via a multicast capable node or a leaf multicast incapable node in the tree. Although its link stress and total cost are better than the Reroute-to-Source and its average delay is superior to Member-only, it is still not satisfying and should be improved in order to adapt the QoS required traffic. It seems no algorithm has been proposed to decide which branch of the multicast incapable branching node should be kept and what kind of reconnection algorithm can be used to reconnect the cut destinations.

In Member-Only algorithm, at each iteration, the nearest destination is added to the multicast tree using the shortest path. But, this shortest path should not include any non-leaf multicast incapable nodes in the tree. It is a modification of Takahashi-Matsuyama heuristic [5]. Although its total cost is approached to the optimal one, there is a big possibility that most of the destinations are connected to the source via a node far away from the source. As a result, its delay is big and the diameter of the multicast tree is always very large.

The rest of the paper is organized as follows. Section III gives some necessary definitions, and section IV discusses the wavelength routing problem. The multicast routing algorithm based on avoidance of multicast incapable branching nodes is proposed and simulated respectively in section V and VI. Finally, a summary of results is made in section VII.

III. SOME DEFINITIONS

We assume wavelength conversion capability is not available in our WDM optical networks. And, the nodes with splitting capability are also sparse because of their complicate architecture and expensive cost. Hence, there are only two kinds of nodes in optical networks: multicast incapable nodes, multicast capable nodes. Without lack of generality, the splitting capability of multicast capable nodes is assumed to be infinite and with no constraint, which is a very ideal situation. In addition, the hop counts are used as a metric to calculate the cost as well as the delay. In order to well describe the problem, firstly some necessary definitions are introduced below.

Definition 1: *MI* and *MC* nodes

MI nodes: Multicast incapable nodes are nodes which can't split, but have *TaC* [9] capability. That is to say, it can tap a small amount of optical power from the wavelength channel while forwarding it to only one output link.

MC nodes: Multicast capable nodes are nodes which are capable of splitting the incoming message to all of the outgoing ports.

In all the graphs of this paper, a *MI* node is denoted by a *rectangle* while a *MC* node is denoted by a *circle*.

Definition 2: Multicast Incapable Branching Node (*MIB* node)

MIB nodes have no splitting capability, but lead to several downstream branches in the multicast tree. Its out degree in the multicast tree is no less than 2. Once it forwards the message to one branch, it could not forward it to another branch using the same wavelength.

Definition 3: Set *MC_SET*, *MI_SET* and *D*

For a multicast tree,

MC_SET: includes the *MC* node and the leaf *MI* nodes in the multicast tree. They may be used to span the multicast tree.

MI_SET: includes only the non-leaf *MI* nodes, which are not able to connect a new destination to the multicast tree.

D: includes unvisited multicast members which are not yet joined to the multicast trees.

Definition 4: Constraint Path (*CP*) and Shortest Constraint Path (*SCP*)

The constraint path between a node u and a tree T is a shortest path from node u to a node v in the *MC_SET* for T. And, this shortest path could not traverse any node in the *MI_SET* for T. That is:

$$CP(u,T) = \{p(u,v) | v \in MC\_SET \ \& \ \forall x \in p(u,v), x \notin MI\_SET\} \quad (1)$$

where $p(u, v)$ denotes the shortest path from u to v in the graph. And the constraint path with the minimum length is called the shortest constraint path (*SCP*).

$$min\{dist[CP(u,T)]\} = dist\{SCP(u,T)\} \quad (2)$$

Accordingly, node v is called the connector for u to T. There may be several *SCP*s from u to T with the same length and so do the connectors.

Definition 5: Connection Constraint Node (*CC* node) and Critical Articulation Node (*CAN*)

If node u is a *CC* node, there must be an intermediate node, which is included in all the paths from u to the source. This intermediate node is called the critical articulation node: $CAN(u, s)$. In other words, u could not reach the source without it. For example, in Figure 1, node *AN* separates the

network into 2 parts: node *d* and source *s* are in different parts.

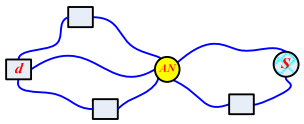

Figure 1. Critical Articulation node

Without node *AN*, *d* is not able to communicate with *s*. So *d* is a *CC* node, and node *AN* is the *CAN*(*d*, s).

IV. Problem Description

In a WDM optical network, a multicast session is assumed to be required with the source *s* and destinations d_1, d_2… d_n. In order to accommodate the multicast group, a multicast tree or multicast forest should be constructed. There are many algorithms proposed to construct a multicast tree with minimum cost. However, nowadays multimedia services such as HDTV, VOIP, Video Conference and Video on Demand are largely used in Internet. They are delay sensitive and bandwidth intensive. Consequently, the link stress and the delay are two important parameters for the multicast tree in WDM optical networks. When the link stress is very high, fewer wavelengths can be used for the other multicast sessions. That means the bandwidth for other multicast sessions is limited. Besides this, power loss and noise are two other challenging problems in all-optical networks. Although power loss can be compensated by appropriate placement of all-optical amplifiers in fibers and cross-connects, noise coming with amplification can be cascaded and is hard to clear without electronic processing. It is practical to limit the length of a path (equals to its delay) in order to decrease the number of amplifiers [6]. In addition, the optical network is more and more used in the Internet Backbone. Although optical messages are transmitted from the source to the destination at a very high speed, the nodes in WDM optical networks are distributed over the world. In this case, the end-to-end delay can't be negligible especially for the delay sensitive traffic.

What is more, the average delay and link stress cannot be minimized simultaneously. If the pruned shortest path tree is used as the multicast tree, although its delay is optimal, its link stress is very high. If approximated Sterner tree is employed as the multicast tree (using Member-Only algorithm [1]), although its link stress is good, its delay cannot be tolerant. So, a tradeoff should be found between them. In order to minimize the average delay, shortest path tree could be used to construct a multicast tree with optimal delay. In order to reduce the link stress, the *MIB* nodes in the shortest path tree could be diminished by making some destinations communicate with the source using longer paths. Based on this main idea, an avoidance of *MIB* nodes based multicast routing algorithm in WDM network with sparse light splitting is proposed in the following section.

V. Avoidance Of *MIB* Nodes For Multicast Routing

The avoidance of *MIB* nodes based multicast routing algorithm can be viewed as a post-processing of the shortest path tree (*SPT*). Just because of *MIB* nodes in the *SPT*, more wavelengths are required to accommodate the multicast group. Hence, they should be avoided in order to decrease the link stress. If there is no *MIB* node in the shortest path tree, then the *SPT* is an optimal multicast tree with both minimum delay and minimum link stress. Otherwise, some process should be done on the *MIB* nodes. This algorithm mainly consists of three steps: the shortest path tree construction step, *MIB* nodes process step and the multicast tree or forest reconstruction step. In the first step, an enhanced DijkstraPro algorithm is introduced to construct a multicast tree with fewer *MIB* nodes and smaller link stress, which makes use of the priority method and node adoption. In the second step, the *MIB* nodes in the shortest path tree are processed, where the deepest branch and the critical articulation heuristics are proposed to keep only one downstream branch of the *MIB* nodes aiming to reduce both the link stress and delay. In the last step, distance based reconnection algorithm is presented to create the multicast forest, which can also reduce the delay.

A. Construction of SPT

First of all, a shortest path tree rooted at the source is constructed for all the nodes in the network. Then, according to the multicast session, non-destination nodes and the nodes that don't lead to any destination should be pruned from the tree.

Generally, Dijkstra algorithm is employed to build the shortest path tree. In Dijkstra algorithm, a node is said to be labeled permanently [7] if its shortest path to the source is found. Otherwise it is said to be tentatively labeled [7]. Initially, only the source *s* is permanently labeled and all the other nodes are tentatively labeled. In each iteration, the node with the shortest distance to the source among all the tentatively labeled nodes is chosen and labeled permanently. What is worth noting is that, in one iteration, there may be several nodes that have the same shortest distance to the source, here we call them as ***candidate*** nodes and the distance is named as their ***level***. However, according to the Dijkstra algorithm, we should label only one of the *candidate* nodes permanently in order to update the distances of the other nodes. But, how to choose it? In traditional Dijkstra algorithm, it is chosen arbitrarily. Think about this situation: there are two *candidate* nodes at the same *level*; one is a *MI* node and another is a *MC* node; they share the same two adjacent nodes. If the *MI candidate* node is firstly chosen to be permanently labeled, then the two adjacent nodes will update their distances to the source, and thus will be connected to the source via this *MI candidate* node. The problem is that the *MI candidate* node cannot split the incoming signal into more than one outgoing port. As a result, it will become a *MIB* node in the shortest path tree. In contrary, if the *MC candidate* node is firstly chosen to be permanently labeled, then those two adjacent nodes update their distances to the source also, and thus they will be connected to the source via this *MC candidate* node. Subsequently, the *MI candidate* node is chosen to be permanently labeled. At this moment, no adjacent node needs to update its distance and no adjacent node is left to be connected to the source via this *MI candidate* node. So, the risk for a *MI candidate* node to become a *MIB* node is reduced or even avoided.

Due to the constraint on splitting capability, the traditional

Dijkstra algorithm may not yield a favorable result. It could be improved and some modifications are required. Hence, DijkstraPro algorithm with priority and node adoption is presented. When building the shortest path tree using Dijkstra, in case of several *candidate* nodes at the same *level*:

❖ *Giving higher priority to MC Candidate node*

The *candidate* node with multicast splitting capability (*MC candidate* node) should be given higher priority than the *MI candidate* nodes. Since, they can connect as many destination nodes as possible to the tree without producing any *MIB* node. In other words, the possibility for a *MI candidate* node in latter iterations to connect more than one destination to the tree is greatly decreased.

❖ *Giving high priority to MI Candidate node with smaller degree*

Moreover, if there is no *MC candidate* node, then the *candidate* node with smaller degree is given a higher priority. That is because, the possibility for a *MI candidate* node with smaller degree (especially for the *candidate* nodes which has a degree of 2) to be a *MIB* node is very low. That is to say, the number of nodes left, which should be connected to the source through other *MI candidate* nodes with higher degree, is very small. Consequently, the possibility for a *candidate* node with higher degree to become a *MIB* node is reduced. So, the average probability for a node to be a *MIB* node is slightly decreased.

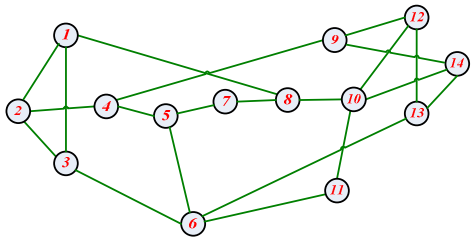

Figure 2. NSF Network

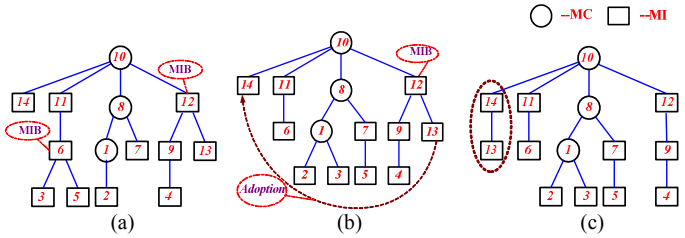

Figure 3. (a) The *SPT* for $m_1$constructed by Dijkstra. (b) The *SPT* for m_1 constructed by offering higher priority to *MC* nodes. (c) The *SPT* after node adoption from (b).

Look at the NSF network in Figure 2, node 1, 8 and 10 are assumed to be *MC* nodes. A multicast session comes: m_1= {source: 10| members: 1~14}. If Dijkstra is used, then we can get the shortest path tree in Figure 3(a). There are 2 *MIB* nodes in this shortest path tree. But, we can see that, node 1, 6, 7, 9 and 13 have the same shortest distance to the source node 10. So, they can be viewed as *candidate* nodes at the same *level*. And, if node 1 (*MC* node) is offered higher priority and firstly chosen to be permanently labeled, followed by 7, 9, 13 and 6, then we can get a new shortest path tree in Figure 3(b), which has only one *MIB* node.

❖ *Node Adoption*

Just at the moment that all *candidate* nodes at the same *level* have been permanently labeled, the following situation may occur. Some *MI candidate* nodes connect only 2 direct children to the tree (i.e. *MIB candidate* nodes) while some *candidate* nodes are leaf nodes in the created tree. So, why doesn't the leaf *candidate* node adopt one child from the *MIB candidate* node at the same *level*, if this child can reach the source through the leaf *candidate* node also? Doing so, this *MIB* node could be avoided. Through node adoption between the *candidate* nodes at the same *level*, the number of *MIB* nodes in the shortest path tree can be reduced or the load of a *MIB* node can be balanced. And the destination node should be given a higher priority to be adopted.

Still see the example in Figure 3(b). It is obvious that node 11, 12 and 14 have the same least distance to the source node 10. Hence, they can be viewed as *candidate* nodes. After all of they have been permanently labeled, we can see node 12 is a *MIB* node and node 14 is a leaf node. Note that, node 13 or 9 can reach the source node 10 by the shortest path through both of node 12 and 14. Thus, one of them could be adopted by node 14, and a new shortest path tree without *MIB* node is obtained in Figure 3(c).

B. Processing of the MIB nodes

Due to the fact that the *MIB* nodes in the shortest path tree can forward the incoming message to one and only one outgoing branch, the existence of the *MIB* nodes is the most important cause of high link stress. So, they should be processed and avoided. In Reroute-to-Source algorithm, all downstream links of *MIB* nodes are connected to the source through the reverse shortest path on different wavelengths, which result in high link stress. Although Reroute-to-Any algorithm is also proposed in literature [1], there is no detail about how to keep one branch when processing the *MIB* nodes. So, in this paper, the deepest branch and critical articulation heuristics are employed to decide which branch should be kept in order to decrease the link stress and the delay.

*1) **MIBPro***

❖ *Critical Articulation Heuristic*

A *CC* node u can only communicate with the source through its $CAN(u, s)$. In a multicast tree, if $CAN(u, s)$ is unfortunately a *MIB* node, then the branch containing u should be assigned a higher priority and kept, when processing this *MIB* node. Since, there is no alternative path for u to reach the

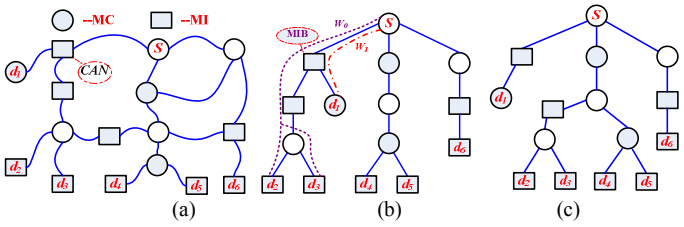

Figure 4. (a) An example network with a *CAN*. (b) The *SPT* for m_2. (c) Process of *MIB* nodes using the critical articulation heuristic.

Processing of MIB nodes Using Critical Articulation and Deepest Branch Heuristics

Step 1: Search all the *MIB* nodes in the shortest path tree

Step 2: For each *MIB* node

Case 1: No downstream branch contains a *CC* node

Keep the deepest branch

Case 2: Only one downstream branch contains a *CC* node

&& *MIB* node = $CAN(CC, s)$

Keep the branch with the *CC* node

Case 3: Several downstream branches contain *CC* nodes

&& *MIB* node = $CAN(CC_i, s)$, i =1, 2, …

Keep the deepest branch with a *CC* node

Step 3: delete the downstream branches of *MIB* nodes which are not kept

source without traversing its $CAN(u, s)$. However, destinations in the other branches may find another path to the source, which will not traverse this *MIB* node. In fact, *CC* and $CAN(CC, s)$ nodes are very rare in a real optical network. However, in case that some nodes in the network have failed, they may exist, and this heuristic will be very practical.

In the network of Figure 4(a), node d_1 is a *CC* node. The shortest path tree for multicast session m_2 = {source: s| destinations: d_1~ d_6} is given by Figure 4(b). We can see $CAN(d_1, s)$ is a *MIB* node. Hence it should be processed. If disconnect node d_1 from $CAN(d_1, s)$ and keep the branch leading to node d_2 and d_3, then 2 wavelengths w_0 and w_1 are required as shown in Figure 4(b). But, if the *CC* node d_1 is kept and cut the other one, then only one wavelength is needed as shown in Figure 4(c).

❖ *Deepest Branch Heuristic*

The deepest branch should also be assigned a higher priority. This is because it seems difficult for a destination far away from the source to find another path to the source without traversing any non-leaf *MI* node in the tree. On the other hand, the delay for a destination node far away from the source will be limited by the length of its shortest path to the source, which is very useful to reduce the average delay. To implement this step, breadth-first traversal algorithm can be employed. So, the worst case time complexity is $O(N)$, where N is the number of nodes in the network.

2) MIBPro2

In addition, another method is also proposed to process the *MIB* nodes in the shortest path tree. For all the *MIB* nodes, it suggests delete all the downstream branches without employing any heuristic. These two methods will be compared in section VI.

C. Reconnection of Multicast Forest

After the *MIB* nodes processing step, the shortest path tree is divided into a subtree plus several separated destinations, which is a disconnected forest and should be reconnected in order to accommodate all the multicast members. The Member-Only [1] like algorithm is a very good method to reconnect the multicast forest. It always adds the nearest destination to the multicast tree using the shortest path, but

Distance Based Reconnection Algorithm

Step 1: For all the nodes in the subtree obtained after the *MIB* nodes process step ,

T = subtree

MC_SET: *MC* nodes and leaf *MI* nodes

MI_SET: non-leaf *MI* nodes

D: separated destinations cut from the tree

Step 2: Find the closest destination $d \in D$ to tree T, and its path should not traverse any node in *MI_SET*:

$dist\{SCP(d, T)\} = \min\{ dist[SCP(d_i, T)] | d_i \in D\}$ (*)

Step 3: If there are several destinations satisfying (*), Select the destination nearest to the source as d

Step 4: For d , if there are several connector nodes in tree T satisfy:

$dist\{SCP(d, T)\} = dist\{SCP_i(d, connectori)\}, i$=1,2,…

Select the connector node nearest to the source and Choose the corresponding *SCP*

Step 5: Connect d to T using its *SCP*.

Step 6: For all nodes in the new tree T,

MC_SET: add d and all *MC* nodes on *SCP* (d, T), remove non-leaf *MI* nodes

MI_SET: add all non-leaf *MI* nodes on *SCP* (d, T)

D: remove d.

Step 7: Go to ***step 2*** until no destination can be added to T

Step 8: Otherwise, set *MC_SET* = {s}, *MI_SET* = Ø, and begin a new tree T = {s}, using the same procedure from ***step 2*** to ***step7***, until D is empty.

this shortest path should not use any non-leaf *MI* node in that tree. That is to say, in each iteration, only the destination with the shortest *SCP* is connected to the tree through its *SCP*. As far as we know member-only algorithm can get the best link stress and the minimum cost. However, its delay is very large. What is worth noting that some improvements can be done to reduce the delay in some extent while obtaining the same cost and the same link stress. The example below will explain how to improve the delay.

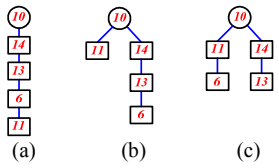

Figure 5. An example of two hints for the reconnection

For instance multicast session m_3 = {source: 10| destinations: 6, 11, 13, 14} is required in NSF network in Figure 2. We assume the first tree only contains the source node 10. According to the member-only like method, the destination with the shortest *SCP* should be added to this tree firstly. The shortest path for node 11 to source and that for node 14 to the source have the same length 1. Without loss of generality, node 14 is chosen to be connected. Then, on the new tree, we can see that both *SCP* for node 11 and 13 have the same length. Also without loss of generality, node 13 is chosen to be connected. After that node 6 is chosen, finally

node 11 is chosen. Following these steps, the multicast tree can be gotten in Figure 5(a). But it is not difficult to notice that node 11 can be connected to the tree via node 10 or node 6. Why don't we connect it through node 10 like in Figure 5(b)? The difference is that the connector node has different distances to the source (for node 10 the distance is 0 while that for node 6 is 3). In addition, it is even more interesting to see the Figure 5(c). All of these three multicast trees have the same cost of 4 while have different average delays: 10/4, 7/4 and 6/4. It is also easy to find that, after node 14 is added to the tree, if node 11 is added to tree earlier than node 13 then, we can get the result in Figure 5(c).

So, from this simple example, we can get two hints in order to reduce the average delay while maintaining the same cost and the same link stress. From this point of view, distance based reconnection algorithm is developed. If there are several nodes, whose *SCP*s to the multicast tree have the same length, and then these nodes should be added in the order of their distance to the source (the distance in the network): the nearer, the earlier. What is more, when the destination with the shortest *SCP* has at least 2 connector nodes in the subtree, it is better to use the connector node nearest to the source (the distance in the multicast tree). Otherwise its delay will be too long.

VI. Performance Evaluation and Simulation

Optical fibers, which can provide high bandwidth, are always used in the backbone of Internet. The NSF Network in Figure 2 is employed as a test bed to simulate the proposed algorithms. This network has been used as a reference topology in many papers [4][6][8], that is why we select it.

A. Performance of DijkstraPro algorithm

In TABLE1, Dijkstra and DijkstraPro algorithm are compared. SOUECE ID denotes the source of the shortest path tree being built, *N* denotes the number of *MIB* nodes in the shortest path trees and *S* denotes the link stress of the shortest path tree (the number of wavelengths required). In CONDITION 1, where only the source is a *MC* node, the average number of *MIB* nodes in the shortest path tree constructed by DijkstraPro is ***0.85*** less (23%) than the original Dijkstra and the link stress is ***0.36*** smaller. This verifies that the node adoption in DijkstraPro algorithm can really work. In the NSF network, node 6 and node 10 have high connectivity (both of their degree are 4), so they can be assumed to be *MC* nodes which are very useful for multicast sessions. In CONDITION 2, where node 6, 10 and the source are *MC* nodes, the DijkstraPro algorithm can also produce a shortest path tree with fewer *MIB* nodes and smaller link stress. The average number of *MIB* nodes is ***0.93*** less (38%) and the link stress is ***0.15*** smaller. Moreover, it is easy to think out that when all the nodes in NSF network are *MC* nodes, all the shortest path tree constructed by the Dijkstra or the DijkstraPro algorithm will not have any *MIB* nodes and their link stress are always 1.

So, it is obvious that, when the ratio of *MC* nodes in the network is very high the improvement of DijkstraPro algorithm is not significant, but when the *MC* nodes are very sparse its performance is much better than the original Dijkstra algorithm.

TABLE 1 COMPARISON OF DIJKSTRA AND DIJKSTRAPRO

***SPT* IN NSFNET**	**CONDITION1** MC: SOURCE MEMBERS: 1~14				**CONDITION2** MC: 6,10 AND SOURCE MEMBERS: 1~14			
SOURCE	DIJSTRA		DISKSTRAPRO		DIJSTRA		DISKSTRAPRO	
ID	***N***	***S***	***N***	***S***	***N***	***S***	***N***	***S***
1	3	4	3	3	1	2	1	2
2	4	3	3	3	3	3	2	3
3	3	4	2	4	2	2	1	2
4	4	3	3	2	4	3	3	2
5	4	4	2	3	3	2	1	2
6	3	2	3	2	3	2	3	2
7	5	5	4	4	3	3	2	2
8	4	4	2	3	3	2	1	2
9	4	3	3	3	3	3	2	3
10	3	2	1	2	2	2	1	2
11	3	4	3	4	1	2	1	2
12	4	3	4	3	2	2	1	2
13	3	4	2	4	2	2	1	2
14	4	3	4	3	2	2	1	2
AVERAGE	3.64	3.43	2.79	3.07	2.43	2.29	1.5	2.14

B. Performance of the Avoidance of MIB Nodes based Multicast Routing algorithm

There is no literature which describes Reroute-to-Any [1] algorithm keeps which branch of *MIB* nodes and which algorithm is used to reconnect the cut destinations. In our simulation, arbitrary branch is assumed to be kept and Member-Only [1] like reconnection method is employed in Reroute-to-Any algorithm.

To evaluate the performance of the proposed avoidance of *MIB* nodes based multicast routing algorithm (MIBPro/MIBPro2), four metrics are used: link stress, average delay, maximum delay and total cost of the multicast forest. The total cost is calculated by the total hop-counts in the multicast forest. And the delay is calculated by the hop-counts from the destination to the source. The link stress is the maximum stress of links in the forest, which equals to the number of wavelengths required. The members of a multicast session are assumed to be uniformly distributed. And the source of the multicast session is chosen from node 1 to 14 in sequence. For each source, 100 random multicast sessions are generated. Hence, the result of each point in the graph is the average of 1400 computations. In addition, Reroute-to-Source (R2S), Reroute-to-Any (R2A) and Member-Only (MO) are also implemented for the comparison.

❖ *Effect of Group Size (Number of Multicast Members)*

Figure 6, 7, 8 and 9 have compared the link stress, average delay, maximum delay and total when the group size varies in two situations: (a) all the nodes are *MI* nodes except the source; (b) node 6, 10 and the source are *MC* nodes while the others are *MI* nodes. In Figure 6(a)(b), we can see when the group size is below 6, the links stress is almost the same for R2A, MIBPro, MIBPro2 and MO; but just when the group size grows larger than 6, MIBPro and MIBPro2 become better

than R2A, whose link stresses are very nearer to MO. In Figure 7(a) and 8(a), when the group size is larger than 10, both the average delay and maximum delay of MIBPro is lower than R2A. We can also see in Figure 7(b) and 8(b) when the group size is larger than 7 the advantage of MIBPro in terms of average delay and the maximum delay become significant. Moreover, the total cost of R2S and MIBPro are almost the same in Figure 9(a) (b).

From these results, we can see MIBPro can get better performance in terms of link stress, average delay and maximum delay when the group size is large. This is because the priority, node adoption and distance based reconnection are not operative when the group size is too small.

❖ *Effect of Splitting Capability (Number of MC nodes)*

The performances when the number of *MC* nodes varies are compared in Figure 10, 11, 12 and 13 in three conditions: group size respectively equals to 4, 8 or 12. At each point, the *MC* nodes are selected arbitrarily. In Figure 10(a), when the group size is 4, and the number of *MC* nodes is between 2 and 7, we notice MIBPro has the best link stress among these five algorithms. And it is also very clear in Figure 10(b)(c) that when the *MC* nodes are spare in the network, the link stress of MIBPro and MIBPro2 are much nearer to the best one (MO) than R2A. In Figure 11(a)(b) and 12(a)(b), the average delay and maximum delay of MIBPro are almost the same as R2A. However, when the group size is 12 in 11(c) and 12(c) the average delay and maximum delay of MIBPro is smaller than R2A.

Form these results, we can deduce when the number of *MC* nodes are sparse, the performance of MIBPro in terms of link stress is always better then R2A and near to MO. This is because the deepest branch and critical heuristics performance well. And when the number of *MC* nodes is large, *MIB* nodes in the shortest path tree is fewer. As a result, the advantage of MIBPro is not significant. We can also see that only when the group size is large, the average delay and maximum delay of MIBPro is smaller than R2A, which corresponds to the previous part (effect of group size).

VII. CONCLUSION

Due to the physical constraint, supporting multicast routing in optical network with spare light splitting and without wavelength conversion is not easy. Many multicast routing algorithms have been proposed to attempt to construct the Steiner tree in all optical networks to get the minimum cost, but this problem is proved to be *NP*-hard. In fact, QoS required applications become more and more popular in Internet nowadays. The bandwidth (or link stress in WDM optical network) and delay are two important parameters for QoS. Hence, a multicast routing algorithm based on avoidance of *MIB* nodes is presented for QoS required traffic in WDM networks in order to decrease the link stress and delay. It keeps the good parts of the shortest path tree which results in the optimal delay for if not may at least some multicast members. In order to reduce the number of *MIB* nodes and link stress in the construction of the shortest path tree step, DijkstraPro algorithm is presented, where higher priority is assigned to *MC candidate* node and node adoption are conducted between the *candidate* nodes. To keep one branch of *MIB* nodes in the shortest path tree, critical articulation and deepest branch heuristics are introduced. And the distance based reconnection algorithm is also developed to create the multicast forest. The first part of the simulation in section VI shows that DijkstraPro algorithm is a better tool for the shortest path tree construction in optical network than Dijkstras. It can really reduce the *MIB* nodes and link stress of the shortest path tree. Moreover, the second part of the simulation proves that, the proposed MIBPro algorithm can yield much better link stress than R2A algorithm when *MC* nodes are very sparse. In addition, when the group size is large it can also get smaller average delay and smaller maximum delay, which approaches to the optimal one.

REFERENCES

[1] Xijun Zhang, John Wei, Chunming Qiao. Constrained Multicast Routing in WDM Networks with Sparse Light Splitting. Journal of Lightware Technology, 18(12):1917-1927, 2000.

[2] N. Sreenath, K. Satheesh, G. Mohan, C. Siva Ram Murthy. Virtual Source Based Multicast Routing in WDM Optical Networks. Photonic Network Communications, 3(3): 213-226, 2001.

[3] N. Sreenath, K. Satheesh, G. Mohan, C. Siva Ram Murthy. Virtual Source Based Multicast Routing in WDM Networks with Sparse Light Splitting. IEEE Workshop on High Performance Switching and Routing'01, p141-145, 2001.

[4] Ashraf Hamad, Tao Wu, Ahmed E. Kamal, Arun K. Somani. On multicasting in wavelength-routing mesh networks. Computer Networks, 50: 3105-3164, 2006.

[5] Aniko Zsigri, Alexandre Guitton, Miklos Molnar. Construction of light-trees for WDM multicasting under splitting capability constraints. International Conference on Telecomunication'03, p171-175, 2003.

[6] Shuguang Yan, Jitender S. Deogun, Maher Ali. Network Routing in sparse splitting optical networks with multicast traffic. Computer Network, 41: 89-113, 2003.

[7] C. Siva Ram Murthy, Mohan Gurusamy. WDM Optical Networks: concepts, design and algorithms. Prentice Hall, 2002.

[8] Lin, Peng Shi, Shuai Li, Jian Zhang, Jie Gu, Wanyi. Novel Virtual Source Based Multicast Routing Algorithm in WDM Networks with Sparse Light Splitting. International Conference on Communication Technology'06, p1-4, 2006.

[9] Maher Ali, Jitender S. Deogun. Cost-Effective Implementation of Multicasting in Wavelength-Routed Networks. IEEE Journal of Lightwave Technology, 18(12): 1628—1638, 2000.

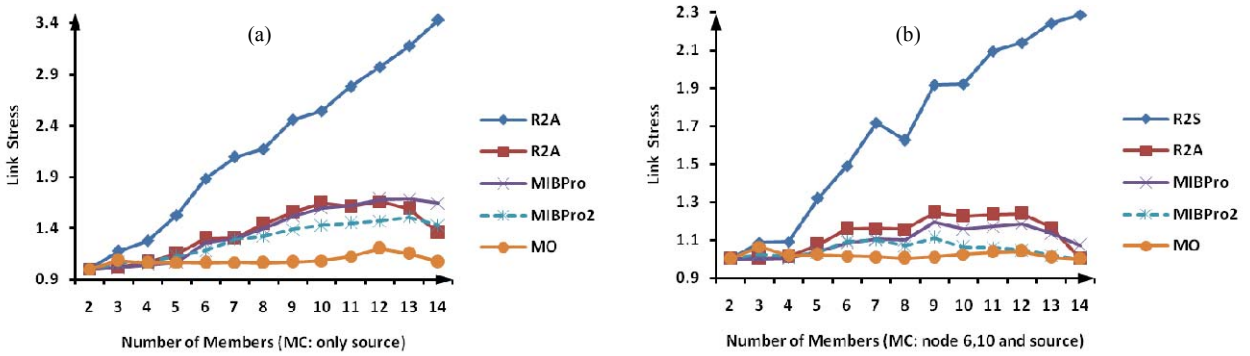

Figure 6. Comparison of Link Stress when the number of multicast members varies

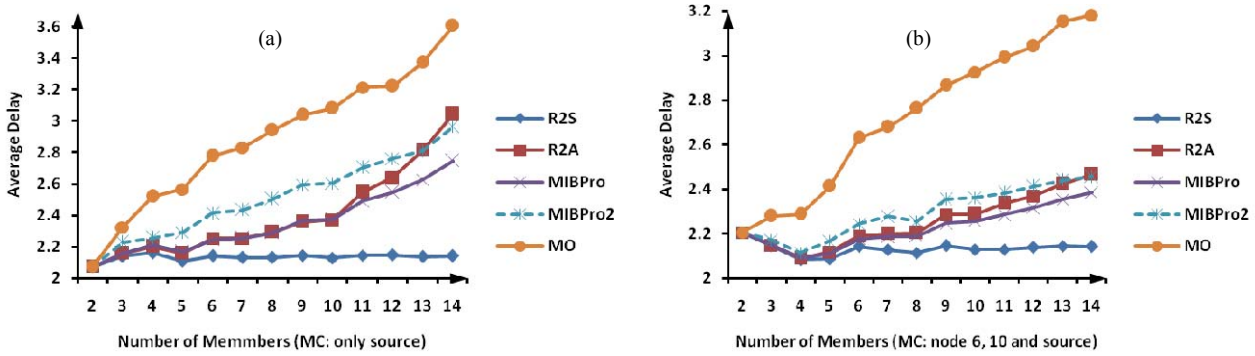

Figure 7. Comparison of Average Delay when the number of multicast members varies

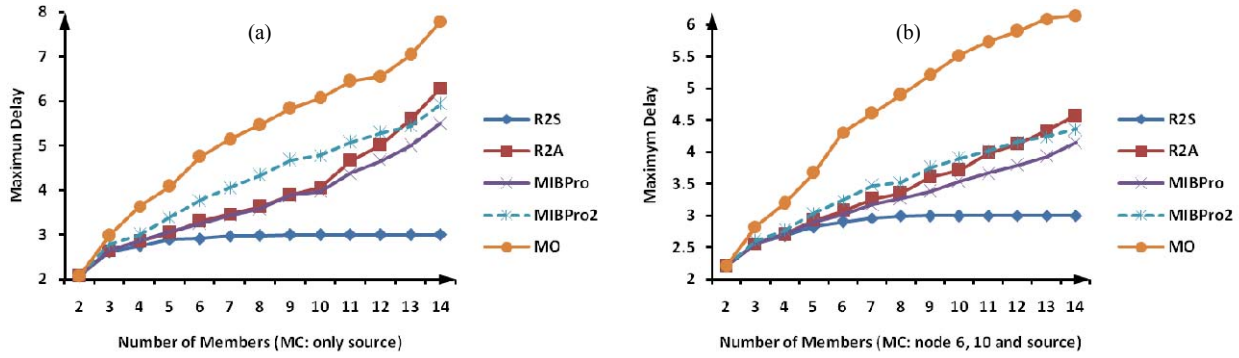

Figure 8. Comparison of Maximum Delay when the number of multicast members varies

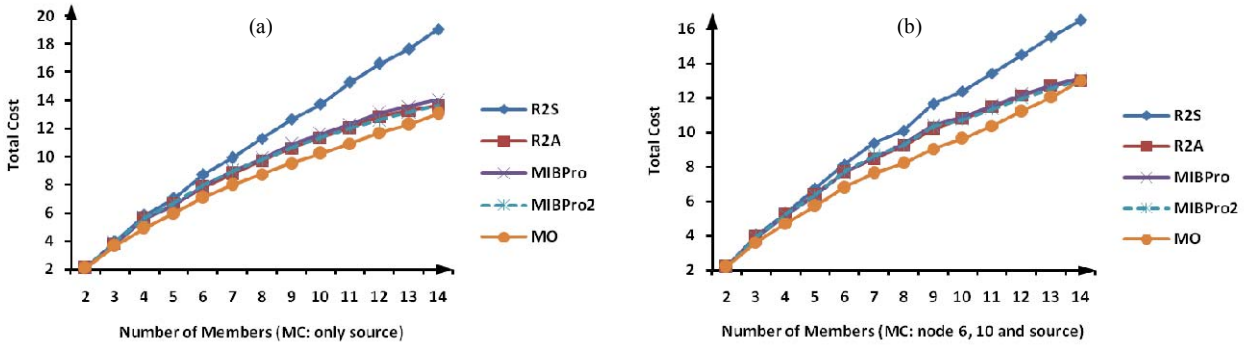

Figure 9. Comparison of Total Cost when the number of multicast members varies

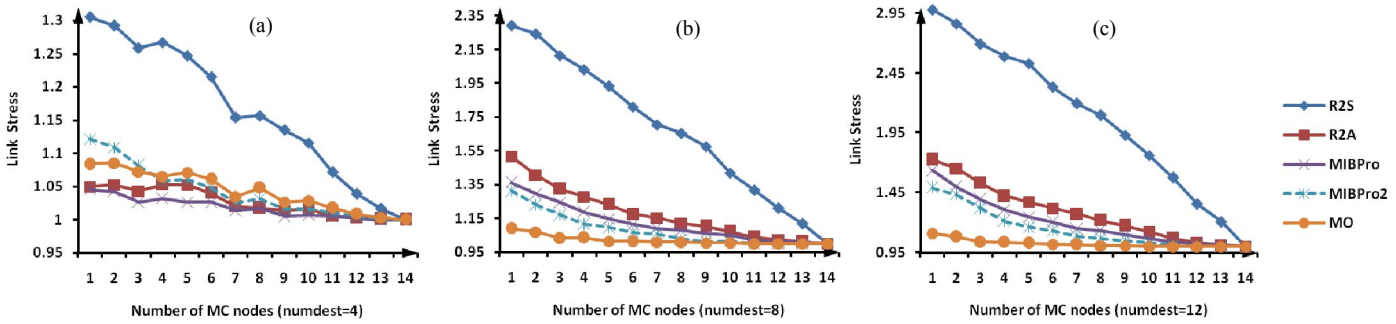

Figure 10. Comparison of Link Stress when the number of *MC* nodes varies

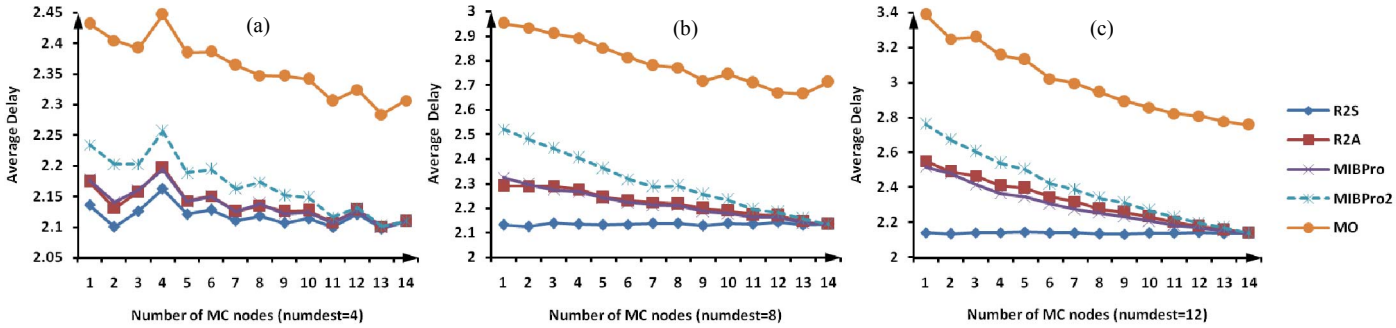

Figure 11. Comparison of Average Delay when the number of *MC* nodes varies

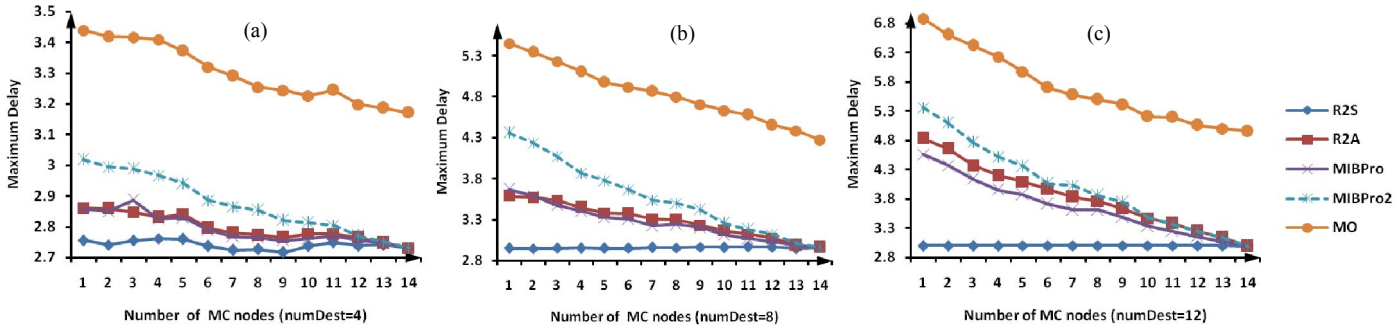

Figure 12. Comparison of Maximum Delay when the number of *MC* nodes varies

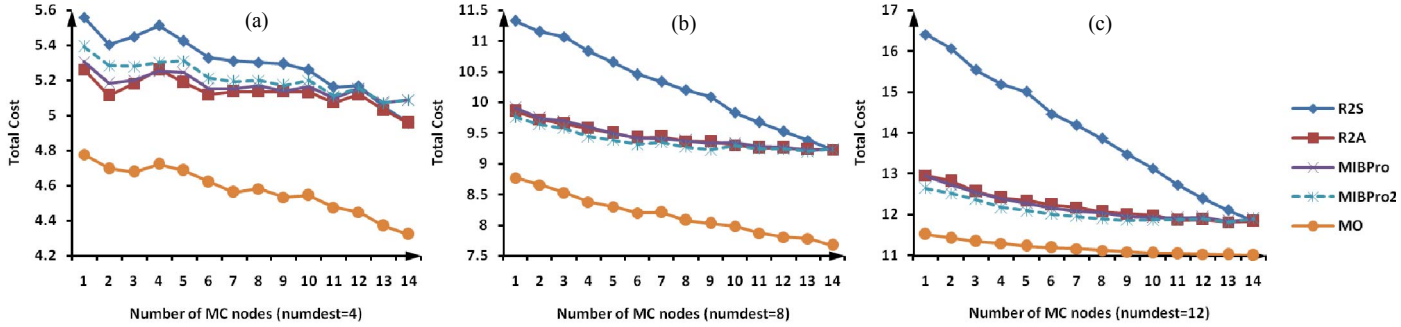

Figure 13. Comparison of Total Cost when the number of *MC* nodes varies