Performance Analysis of All-Optical Multicast Routing Algorithms with Sparse Splitting

Dinh Danh Le, Miklós Molnár, Jérôme Palaysi

Performance Analysis of All-Optical Multicast Routing Algorithms with Sparse Splitting

Dinh Danh Le
LIRMM
Université Montpellier 2, France
Email: dinhdanh.le@lirmm.fr

Miklós Molnár
LIRMM
Université Montpellier 2, France
Email: miklos.molnar@lirmm.fr

Jérôme Palaysi
LIRMM
Université Montpellier 2, France
Email: jerome.palaysi@univ-montp2.fr

Abstract—**In this paper we study the multicast routing problem in all-optical WDM networks with sparse splitting capacity. In the literature, there are several proposals on finding the best way to construct multicast light-trees (or light-forests) with the objective of minimizing the number of wavelengths (*link stress*), the maximum end-to-end delay from the source to the destinations (*maximum delay*) and/or the total number of wavelength channels used (*total cost*) of the light-forests. However, balancing all these criteria is very difficult. Particularly, end-to-end delay and link tress cannot be minimized simultaneously. Hence, it is interesting to find a good approach that can provide a trade-off solution. Putting this into practice, we propose a comparative study of the most known algorithms and introduce a new one which can provide a good trade-off among those three criteria. Simulation results and comparison point out that our proposal provides the best link stress, a low maximum delay and low total cost of the multicast forest among considered algorithms. Especially, our proposal works best in dense networks, and/or with a large multicast group size in comparison to the classical algorithms.**

Keywords: **all-optical WDM networks, sparse splitting, light-tree, light-forest, multicast routing.**

I. Introduction

In the network evolution, optical network is considered as a modern type of networks that can provides with a huge bandwidth (about Tb/s) and low communication latency. It is getting mature to be deployed world-wide in the Internet core network in order to serve a larger number of high performance applications coming from various kinds of devices such as computers, smart-phones, TVs, etc.

Wavelength Division Multiplexing (WDM) is a type of Frequency Division Multiplexing technology in optical frequency domain, where each fiber consists of several communication channels at different wavelengths. At any time, a pair of nodes in the network can be connected to each other via a channel supported by a single wavelength. Different channels must use different wavelengths, and they can only use the same wavelength if they do not have any common links [5].

Multicasting in WDM networks relates to transmitting optical data from a single source to multiple destinations concurrently. This type of communication is getting more important in today Internet due to an increasing number of multimedia and real time applications concerning several destinations, such as video and phone conference, distance e-learning, online television, etc. There are several ways of multicasting in WDM networks. In the approach based on unicast communication, data packets are first duplicated at the source as many as the number of destinations, then each of these copies is routed to each of destinations. This simple way is proved to be bandwidth consuming [10] and do not utilize available optical splitting capacity. In the other approach, multicasting at WDM layer can be operated with the support of Multicasting at WDM layer can be operated with the support of splitters, i.e. Multicast Capable Optical Cross-Connects (MC OXCs, or MC in short) [11]. In this scheme, incoming light signal can be duplicated by splitting data at intermediate MC nodes of a *light-tree*. Since communication can use common links, WDM muticast saves considerably bandwidth consumed compared to the first scheme, and it is more desirable for multicasting in all-optical networks.

However, the implementation of WDM multicasting is not easy due to not only the difficulty of multicast routing problem itself but several optical constraints also. The first constraint comes from sparse light splitting capacity, i.e. the small number of MC nodes in the networks. Many studies showed that, about below 50% of the OXCs in WDM networks can be MC OXCs due to their costly fabrication and because they require a large number of amplifiers [1], [2]. This constraint makes WDM multicasting challenge, because lack of MC OXCs induces more light trees for a given multicast session, and so more bandwidth consumption.

The second constraint is the limitation of number of wavelengths can be supported on optical fibers. Wavelength is the smallest transmission unit in WDM networks, each wavelength corresponds to one channel. The more the wavelengths supported in each fiber, the more the bandwidth available. However, the number of wavelengths that can be supported is limited by optical device technology [7]. Thus, in order to serve an increasing number of multicast applications, multicast routing algorithms should be carefully designed with the awareness of minimizing number of wavelengths used.

The third constraint relates to the *distinct wavelength constraint* (i.e. two light-paths or light-trees can not be assigned with the same wavelength if they have some common links). If there is wavelength conversion (WC) [7] presented in the network, the light-trees can be constructed more flexibly. However, WC has not been deployed widely in reality due to its expensive fabrication and immature technology [3], [8].

For this reason, we make the assumption that there is no WC in the network. Thus, the distinct wavelength constraint must be respected.

In this paper, we focus on the well-known algorithms for constructing multicast light-forests proposed in [10] in the consideration of optical constraints mentioned above. Previous studies show that these algorithms are not always effective in term of balancing multiple criteria (i.e. cost, delay and link stress). We therefore recommend a new algorithm that can provide a better trade-off solution. The rest of this paper is organized as follows. We first review the studies related to WDM multicast routing in sparse splitting networks in Section II. In Section III, the problem is modelled and performance metrics for the evaluations are also introduced. In Section IV, we focus on our proposed algorithm and describe it in detail. The simulation results with evaluations are showed in Section V before the conclusion.

II. Related work

As mentioned above, MC OXC is the key element in supporting multicast in WDM networks. However, equipping full splitting capacity in WDM switches is not realistic due to its complicated fabrication and high expense. Besides, wavelength conversion is also helpful for constructing more flexible multicast light-tress, but it is also expensive and immature enough to deploy in reality. Thus, sparse splitting capacity and without wavelength conversion are supposed in many studies [1], [2], [9]–[11]. For this reason, our research is also carried out in sparse splitting WDM networks with the absence of WC. In addition, for simplicity, we also assume that any MC nodes in the networks can split incoming light signal to all of their outgoing ports. With these assumption, in this section, we review the well-known multicast routing algorithms in sparse splitting constraints proposed in the literature, including the four algorithms in [10], and Avoidance of MIB node [11] and briefly evaluate their performance.

In [10], four multicast light-forest computation algorithms were presented, including Reroute-to-Source, Reroute-to-Any, Member-First and Member-Only. The first three algorithms are based on *Shortest Path Tree* (SPT), in which Member-First is quite different when taking membership information into account, and Member-Only is based on *Steiner Tree Approximation*.

In the first two algorithms (i.e. Reroute-to-Source and Reroute-to-Any), a spanning tree is first created from a source to all destinations by employing a shortest path algorithm (e.g. Dijkstra's algorithm). Then, the algorithm checks the light splitting capability of each branching node in the SPT. If the number of its children is greater than its splitting capacity, only some children are kept, and the other must be cut and re-routed. For example, if the branching node is a multicast incapable branching node (MIB node), then only one child can be kept, which is chosen arbitrarily. All the other children (and sub-trees rooted at them) must be re-routed to the current tree either at an MC node along the shortest paths to the source (Reroute-to-Source), or re-routed to any other node on a tree (which can be an MC node or a leaf MI node) if possible (Reroute-to-Any). Obviously, the end-to-end delay of Reroute-to-Source is minimal. However, the link stress can be very high, because downstream branches of an MI node have to communicate with the source using the same shortest path but on different wavelengths. In contrast, the end-to-end delay of Reroute-to-Any is higher, and the number of wavelengths is less compared to Reroute-to-Source.

Member-First is also based on SPT while taking membership information into consideration. The algorithm manages adjacent fringe links and constructs the light-tree iteratively, by adding the link with the highest priority at each step. The constructions begins by the source. When a destination (u) is added, it checks every MIB node (x) from u back to the source, keep only one branch and cut all other branches rooted at x. It stops when the current set of possible fringe links is empty, then it constructs another multicast tree from the source until all the destinations are included (a more detailed description can be found in Section IV). According to [10], Member-First achieves a better link stress and cost in compared to Reroute-to-Any, and produces a good trade-off among performance metrics (i.e. link stress, total cost, maximum delay, and average delay).

On the Steiner Tree Approximation based approach, Member Only begins to build a multicast light-tree by connecting the destinations to the source one by one just through the shortest paths (the closest, the first). At each step, it tries to find the shortest path from the destinations to the current multicast light-tree so that the shortest paths do not traverse any MIB nodes. If it is found, the corresponding destination and the path are added to the light-tree. Otherwise, the current tree is terminated and a new one on another wavelength is started until all the destinations have been covered. According to [10], the light-trees computed by Member-Only algorithm have the best total cost among the other. However, because the distance from the destinations to the source is not taken into account, many destinations might be connected to the light-tree via a node far away from the source. Consequently, the diameter of the multicast tree is often very large (and hence, the end-to-end delay is very high).

In [11], the authors proposed an algorithm called Avoidance of MIB Nodes for multicast routing, and it can be viewed as the improvement based on Reroute-to-Any algorithm. First, it uses a modified Dijkstra algorithm taking some priorities of candidate nodes into account to construct an SPT rooted at the source. Then it processes MIB nodes of the SPT by special heuristics to resolve conflicts. Finally, it uses a distance based post-treatment to create the final light-forest. According to [11], the algorithm is better than Reroute-to-Any and provides a good trade-off between link stress and end-to-end delay when MC nodes are very sparse in the network.

Reroute-to-Source connects the destinations to the source via shortest paths. The cost and the link stress of its solution can not significantly be improved. Member-Only tries to minimize the total cost at the expense of delays. The improvement of these algorithms is not trivial. Avoidance of MIB Nodes for multicast routing is a good improvement of Reroute-to-Any. Member-First is better than Reroute-to-Any in general and especially, its framework (the incremental construction of light-trees using fringe links) is promising to design efficient algorithms. Thus, we select Member-First to design a more efficient optical multicast routing algorithm. Our objective is to analyse the effect of priority definition on the performance of the algorithms. In the next sections, our proposed algorithm is described in detail, and some analysis and comparison between the mentioned algorithms are also given.

III. Problem Modelling and Performance Metrics

The network is modelled by a pair (G, S) where G is a connected undirected graph and S is a subset of vertices of G representing the MC nodes (V_G presents network nodes and E_G presents network links[1]). We suppose that there is a pair of fibers between the connected nodes, so the graph is undirected. Any link $e \in E$ is associated with a cost $c(e)$ and a propagation delay $d(e)$. We consider (s, D) as the multicast session triggered from the source node s to the set of destinations $D = \{d_1, d_2, ..., d_n\} \subset V_G$, $s \notin D$. Let $S \subset V_G$ be the set of MC nodes in the network. Because of sparse splitting capacity of the network, it is likely that a single light-tree may not be sufficient to span all the destinations. Therefore, we assume that k light-trees will be built composing a light-forest $F = \{LT_i, i = 1, \ldots, k\}$. Since these k light-trees are not edge-disjoint, they must be assigned with different wavelengths (due to the *distinct wavelength constraint*). The number of wavelengths required for the multicast session (s, D) is equal to the number of light-trees in F, and it is called *link stress*. Accordingly, the first performance metric is defined as $LinkStress(F) = k$.

Another metric, the *total cost* of the forest F is the sum of cost of all the links on all the light-trees of the light-forest: $TotalCost(F) = \sum\limits_{i \in [1,k]} \sum\limits_{e \in LT_i} c(e)$.

Besides, although optical fibers can support a very high speed in optical networks, there is a propagation delay on each link. When the network size becomes large, the additive end-to-end delay will be considerably along the long distance. Thus, end-to-end delay is a metric that should be paid attention. Let LP_{s,d_i} be the light-path from the source s to destination d_i. Two metrics relating to end-to-end delay (i.e. *maximum delay* and *average delay*) can be calculated as follow:

$$MaxDelay(F) = \max_{d_i \in D} \sum_{e \in LP_{s,d_i}} d(e)$$

$$AvgDelay(F) = \frac{1}{|D|} \sum_{d_i \in D} \sum_{e \in LP_{s,d_i}} d(e)$$

With all the notations mentioned above, the all-optical multicast routing problem can be stated as follow:

- **Input:** A connected undirected graph G, a set of MC nodes S, a multicast session (s, D)
- **Output:** A light-forest F for the multicast session (s, D) satisfying the constraints: the leaves of the light-trees are destinations and the branching nodes are in S.
- **Objective:** A trade-off solution among following criteria: *link tress*, *total cost*, and *end-to-end delay* (i.e. *maximum delay* and *average delay*).

[1]For any graph G, we denote V_G the set of its vertices and E_G the set of its edges.

As many studies (e.g. [10], [11]), for simplicity in calculating performance metrics, we assume that all the wavelengths are equally expensive (or cheap), the bandwidth consumed using a wavelength on different links is the same as well. In addition, the propagation delay is also the same on each link. With this assumption, for every link e, $c(e) = 1$ *unit cost* and $d(e) = 1$ *unit delay*. Performance metrics can now be briefly defined as follow: the *link tress* is the number of wavelengths needed in the forest, the *total cost* is the total number of branches of the forest, the *maximum delay* and the *average delay* is the maximum number of hops, and the average number of hops from the source to the destinations, respectively.

IV. Member-Splitter First algorithm

Before describing our proposed algorithm, we first give some more details of Member-First algorithm, then analyse it for possible improvements.

A. Member-First algorithm

This heuristic constructs light-trees incrementally by examining the possible fringe links of the trees. In [10], a *fringe link* is defined as a link adjacent to the tree without forming a cycle with edges of the tree as well as of the fringe link set. These fringe links are managed by a priority queue. Accordingly, each fringe link e is associated with a priority p_e, and the fringe link set is represented by a set $L = \{e, p_e\}$. The smaller the value of the fringe link, the higher the priority of it. The tree construction begins at the source and can be briefly described as follows.

```
1: Initialize the fringe link set $L$ with the adjacent links of the source
2: Select the fringe link with the highest priority from $L$
3: Add the selected link (and its end-node $n$) to the tree and remove it from $L$
4: Remove impossible multiple children of MI nodes from $n$ back to the source
5: Update $L$ with the adjacent links of $n$
6: If there are fringe links in $L$, go to step 2
7: Prune branches that do not lead to any destinations
8: If there are destinations not yet covered, then go to step 1 to construct another tree
```

Keeping the basic sequence of Member-First as a framework, the algorithm can be improved to construct more favourable light-trees by changing the priority of fringe links and the procedure to update them.

B. Priority Definition

One of the primary factor that affects a lot on algorithm's performance is the priority of links in the fringe link set.

In Member-First, the priority of a fringe link (v, u) can be briefly presented as the order of (*h, member*), in which h is the number of hops (or length) from the source to u, and *member* stands for the multicast membership of u. Accordingly, the link (v_i, u_i) have higher priority than (v_j, u_j) if $h(u_i) < h(u_j)$, or when $h(u_i) = h(u_j)$, u_i is a member, but u_j is not [10].

In fact, there are more than two factors mentioned above that can affect the performance of the algorithm as analysed in following sections. Moreover, Member-First provides a good framework for easily alternating the possible combinations of these factors. In this section, we analyse several possible alternatives and study their influence on the algorithm's performance.

Apart form the two factors taken by Member-First $(h, member)$, the other ones that can affect the algorithm performance are MC (the node is MC or not) and $degree$ (the degree of a node). Thus, there are totally four factors must be taken in to account: $(h, member, MC, degree)$. Different combinations of them can lead to different results, and to choose the best combination needs careful theoretical analysis as well as realistic basis. Naturally, a good combination must be corresponding to the order of the importance of each parameter. Let us analyse each of these parameters in detail.

For the first factor, the number of hops h (or the length) from a node to the source directly impacts the end-to-end delay. In fact, Reroute-to-Source is taking only this metric into account resulting in the best delay but the high total cost. The second factor $(member)$ can affect the total cost and the link stress of the light-forest as the case of Member-Only. Recall that Member-Only connects the destinations to the current tree one by one, just considering the membership information, so the cost is best but the delay is too high.

Member-First takes two of these factors (with sequence of $h, member$) in order to give a more trade-off between the end-to-end delay and the total cost. However, it does not regard to MC nodes available in the network. Thus, it leaves a question that when all the previous factors $(h, member)$ are the same for two fringe links, which link should be given higher priority. Obviously, in such a case, the link leading to an MC node should be the first choice because MC nodes can connect to many of its children (probably including destinations) with only a single wavelength. From this set of children, the connection to other destinations seems to be more probable. Thus, we propose to give higher priority to links leading to MC nodes.

Finally, the $degree$ of nodes also has a significant effect on the quality of the multicast tree. Indeed, the probability that a high degree node can lead to other destinations not yet spanned is higher. However, high degree MI nodes are likely to produce more MIB nodes in the resultant light-tree, resulting in more wavelengths needed to cover all the destinations. That means the link stress can be high. Thus, giving higher priority to the higher-degree nodes can lead to lower end-to-end delay. In contrast, MI nodes with smaller degree probably induce higher delay but lower link stress.

From this fact, we give higher priority to links leading to

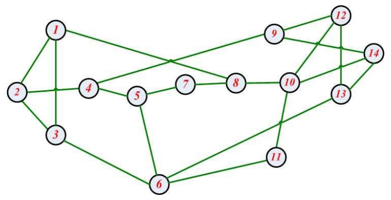


Fig. 1. NSF network topology

the MC nodes with higher degree, and then MI nodes with smaller degree (when they have the same h and $member$). By this way, the end-to-end delay and the link stress can be balanced.

Combination of these aforementioned elements, the new priority of fringe links can be defined in the order of: *h, member, MC, degree*. In particular, a link (v_i, u_i) have higher priority than (v_j, u_j) if:

- $h(v_i) < h(v_j)$;
- or when $h(v_i) = h(v_j)$, u_i is a *member*, but u_j is not;
- or when $h(v_i) = h(v_j)$ and if their memberships are the same, u_i is MC and u_j is MI;
- or when all the above criteria are the same, if both are MC nodes, then u_i has higher degree; otherwise, if both are MI nodes, then u_i has smaller degree.

To verify our analysis above, we carry out several simulations employing Member-First with NSFNET network (Figure 1) in which the order of these factors are alternated and compared together. The selected alternative orders are:

1) h, member (H-Member)
2) member, h (Member-H)
3) h, member, MC (H-Member-MC)
4) h, MC, member (H-MC-Member)
5) h, member, MC, small degree (H-Member-MC-sDeg)
6) h, member, MC, high degree (H-Member-MC-hDeg)
7) h, member, MC, high degree for MC nodes and small degree for MI nodes (H-Member-MC-aDeg)

In the simulations, each node of the network is in turn selected as the source for a multicast session. For a given source, a given multicast group size, and a given fraction of MC nodes, 100 random multicast sessions are generated. (The destinations and MC nodes are distributed randomly through the network.) Hence, the result of each point in the simulation figures is the average of $100 \times |V|$ computations on the four metrics mentioned above: *link stress*, *total cost*, *maximum delay* and *average delay*.

In our simulations, two types of network configurations are set. The performance of the algorithm versus multicast group size and performance of the algorithm versus the number of MC nodes. For the first configuration, the number of MC nodes is set fixed while the group size varies. In this case, the number of MC nodes is set at 3 nodes (˜23% for the sparse splitting capacity) and the group size is varied from 1 to 13 nodes. MC nodes and destinations are distributed randomly in the network. (In all cases, the source is not counted in number of MC nodes as well as the group size.) The performance metrics are calculated and plotted in Figure 2. For the second configuration, the group size is set at 10 nodes (˜77%) while the number of MC nodes varies from 1 to 13 nodes. The simulation results are shown in Figure 3.

Firstly, we give the comparative between the two pair of first sequences: *H-Member vs. Member-H* and *H-Member-MC vs. H-MC-Member*.

H-Member versus Member-H

Now let us take a look at the first pair *H-Member* vs. *Member-H*. As it is shown in Figures 2 and 3, *H-Member* provides a low end-to-end delay, but high link stress and total cost. In contrast, *Member-H* helps Member-First achieve lower link stress and cost but high delay. That is exact what we talked about these two factors above. Consequently, choosing one or the other depends on what objective of the problem. That is, one favours of better link stress and cost should choose *Member-H*, and other supports a better delay should choose *H-Member*. For our improved algorithm, we choose the combination *H-Member* as the first order in later combinations.

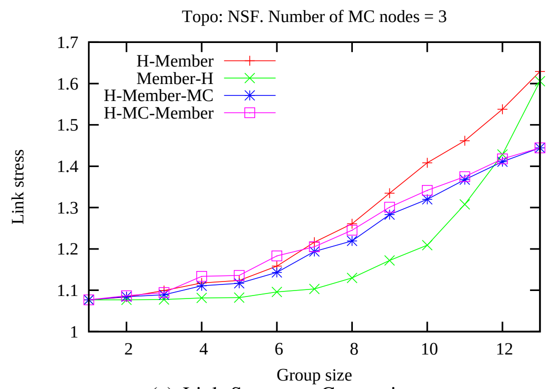
H-Member-MC versus H-MC-Member

Let us now focus on the second pair *H-Member-MC vs. H-MC-Member*. We can see that, when taking MC parameter into account, these combination can help to provide a balance solution compared with the two first ones. In deed, these can reduce link stress and cost of *H-Member*, concurrently reduce end-to-end of *Member-H*. Between the two, *H-Member-MC* is better than *H-MC-Member* in all the performance metrics. Thus, *H-Member-MC* is selected for the last combinations that we will investigate.

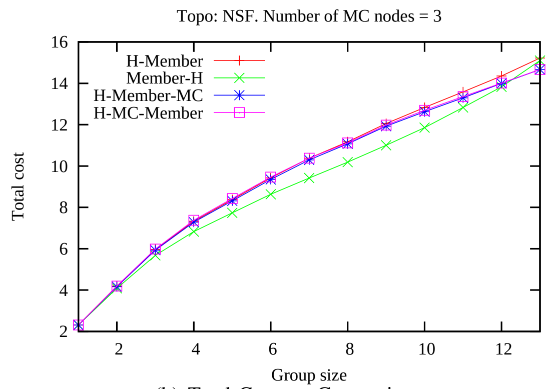
H-Member-MC-sDeg, H-Member-MC-hDeg and H-Member-MC-aDeg

To verify the correctness of our selection of the final priority combination, we also operate simulations to compare the three last sequences mentioned above. The simulaions are also carried out by using Member-First with NSFNET as the same two network configurations as we did with the four first sequences before. The results are shown in Figures 4 and 5.

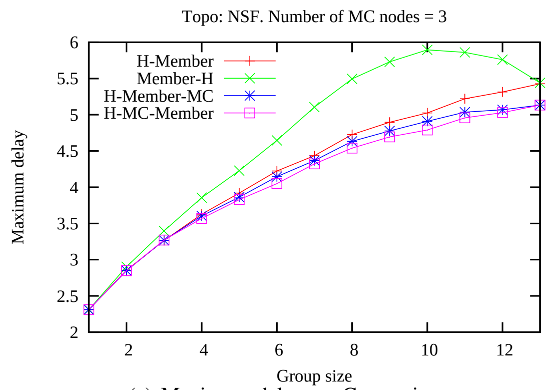
We can see that *H-Member-MC-aDeg* is always ranked the second among the other two, meaning that it can provides a good trade-off between the two. As the result, this combination can be a good candidate for our algorithm as described in the following sections.



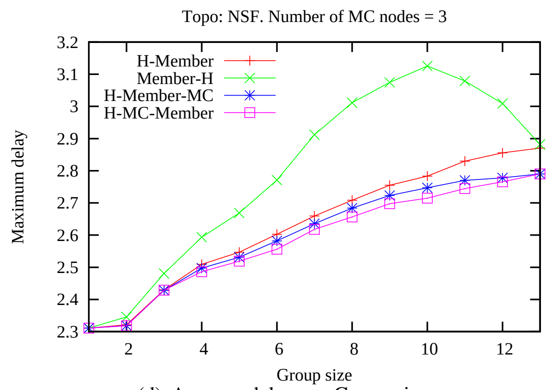
(a) Link Stress vs. Group size



(b) Total Cost vs. Group size

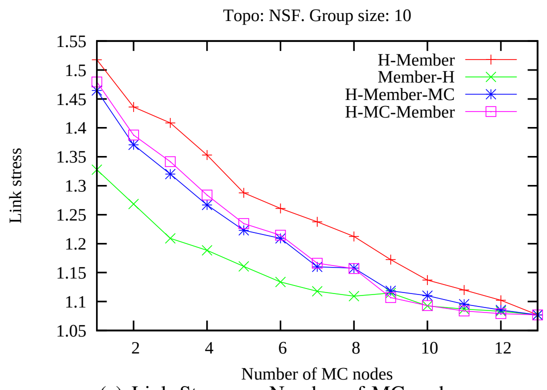


(c) Maximum delay vs. Group size

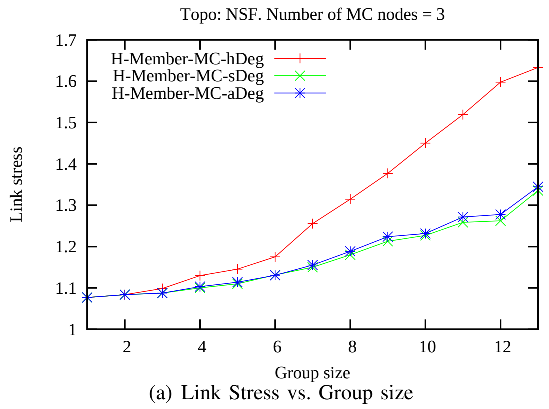


(d) Average delay vs. Group size

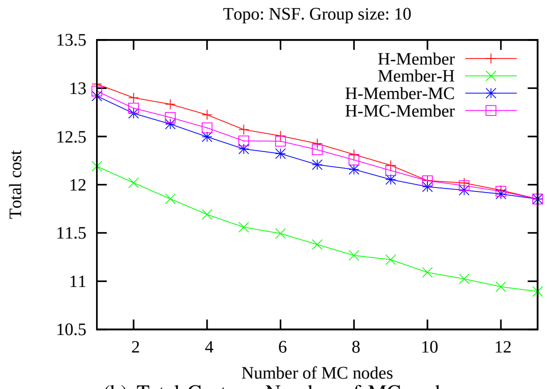
Fig. 2. H-Member vs. Member-H and H-Member-MC vs. H-MC-Member when Group size varies in NSF network with 3 random MC nodes



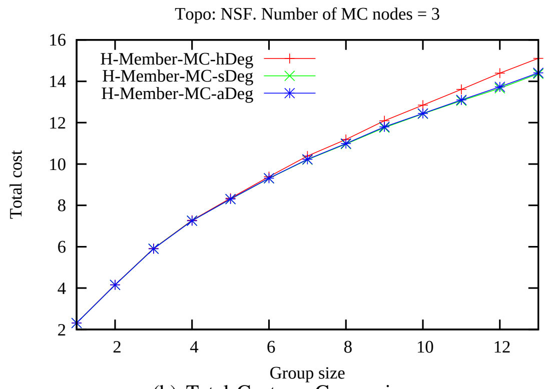
(a) Link Stress vs. Number of MC nodes



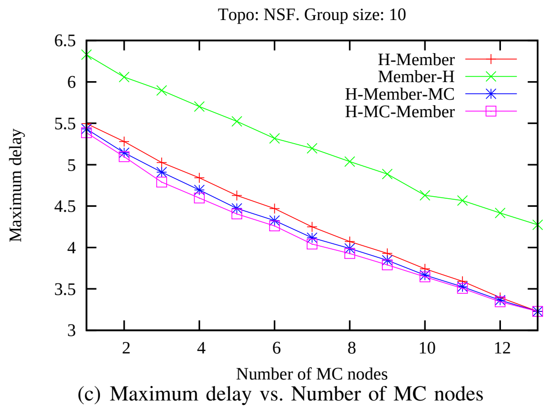
(a) Link Stress vs. Group size



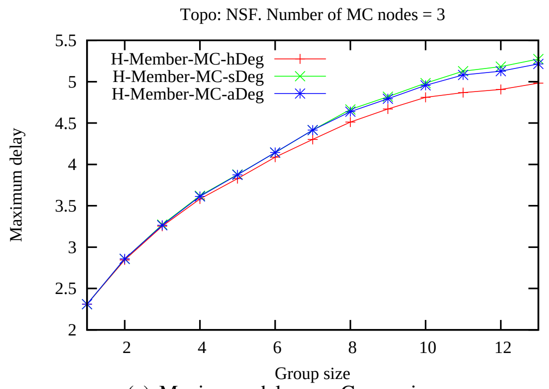
(b) Total Cost vs. Number of MC nodes



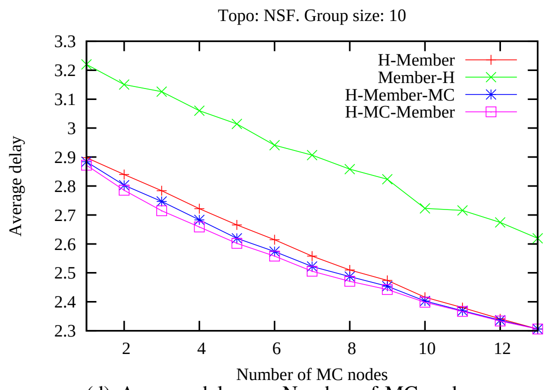
(b) Total Cost vs. Group size



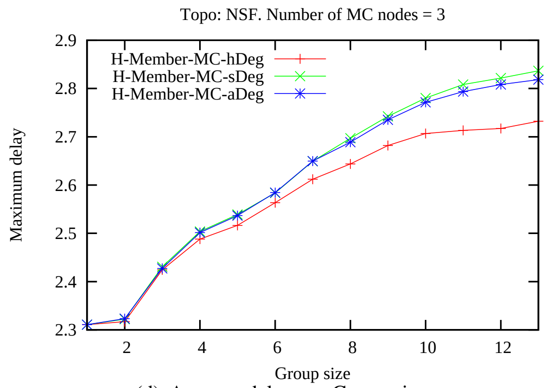
(c) Maximum delay vs. Number of MC nodes



(c) Maximum delay vs. Group size



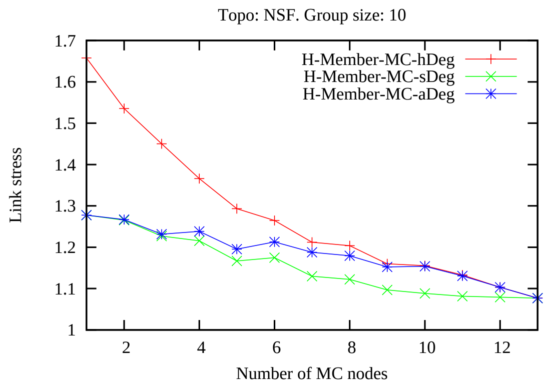
(d) Average delay vs. Number of MC nodes



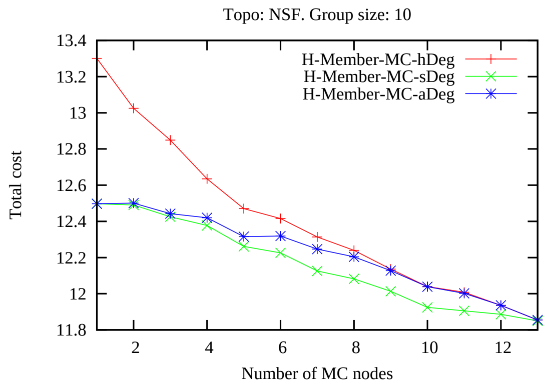
(d) Average delay vs. Group size

Fig. 3. H-Member vs. Member-H and Member-MC vs. MC-Member when Number of MC nodes varies in NSF network with group size of 10

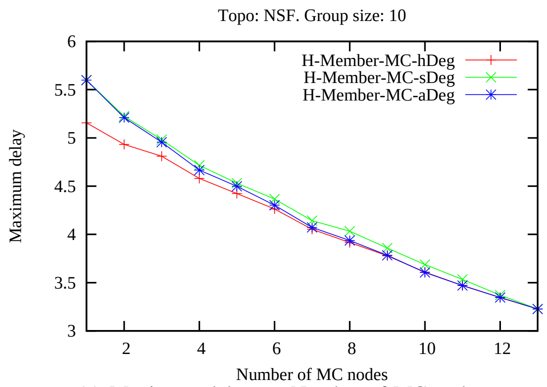
Fig. 4. H-Member-MC-sDeg, H-Member-MC-hDeg and H-Member-MC-aDeg when Group size varies in NSF network with 3 random MC nodes



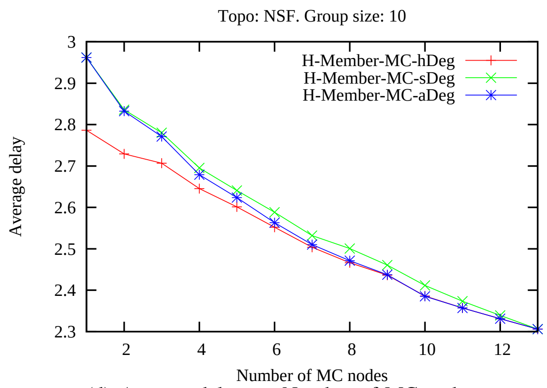
(a) Link Stress vs. Number of MC nodes



(b) Total Cost vs. Number of MC nodes



(c) Maximum delay vs. Number of MC nodes



(d) Average delay vs. Number of MC nodes

Fig. 5. H-Member-MC-sDeg, H-Member-MC-hDeg and H-Member-MC-aDeg when Number of MC nodes varies in NSF network with group size of 10

To give a better visualization, let us consider the network in Figure 6 where the MC nodes are drawn in a circle (including 4, 6, 9), and MI nodes in a square (the other nodes). Suppose that the multicast session is triggered at node 6 to the destination set $\{1, 2, 3, 5, 7, 8, 10, 11\}$ that are shaded in grey.

We can compare the two light-forests constructed by original Member-First (Figure 7) and by Member-First with the new priority definition (Figure 8). When constructing the light-trees according to original Member-First, we assume that when all the factors *(h, member)* are the same for two fringe links, the algorithm choose the link in the order of nodes, the smaller the sooner. After pruning branch (6, 4) (step (6)), in Figure 7, the light-forest created by the original Member-First consists of 2 light-trees (*link stress* = 2), with the total number of branches is 10 (*total cost* = 10), the maximum number of hops from the source to destinations is 3 (*max delay* = 3) and *average delay* = 2.0. In Figure 8, the light-forest created by Member-First with the new priority definition is better with one light-tree (*link stress* = 1), *total cost* = 9, *max delay* = 4, and *average delay* = 2.25.

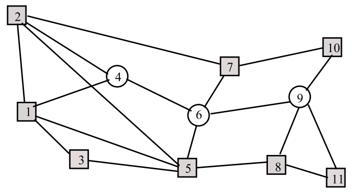


Fig. 6. A network to consider

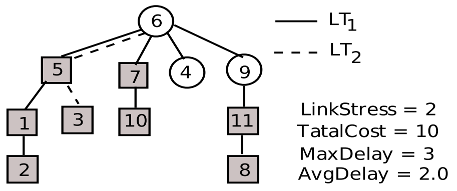


Fig. 7. Member-First light-forest

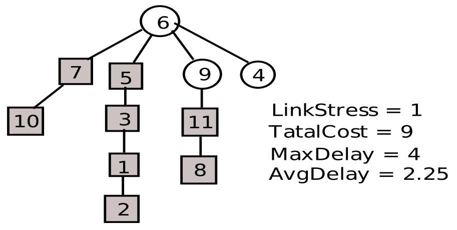


Fig. 8. Member-First with New Priority Definition light-forest

C. Updating Bud-Links

When updating fringe links (step 5) from a node, node v for example, Member-First algorithm adds all the possible adjacent links of v to the fringe link set L regardless to its splitting capacity. It is worth noting that, if v is an MC node, there is no problem. However, if it is an MI node, it can

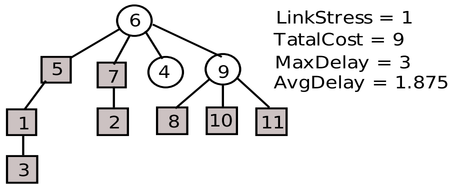


Fig. 9. Member-First with New Bud-Link Concept light-forest

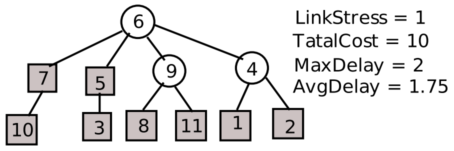


Fig. 10. Member-Splitter First light-forest

support only one child, if all the possible adjacent links are added, there is no chance for other nodes (if any) connecting to these links. Let us use the topology in Figure 1 and employ Member-First to explain this point in detail. Suppose that at first Member-First select link (6,5), there are four possible adjacent fringe links from node 5: (5,1), (5,2), (5,3) and (5,8). If all of these links are added to fringe link set L, then when updating L from node 4, there is no chance for node 4 to connect 1 and 2, because (4,1) and (4,2) have the same priority to (5,1) and (5,2), and hence they cannot be updated (according to the procedure *UpdateFL* in [10]). This is one of the key reasons making Member-First inefficient.

Consequently, it is more reasonable to add only one link from an MI node, that selected link must have the highest priority. By this way, when selecting the highest priority link from L in order to add to the multicast tree, we just go straight-forward without any branch-cutting or link-removing. Moreover, there are chances for other adjacent nodes of v to be end-nodes of other fringe links.

To do it, we define a new concept corresponding to fringe links and call them *bud-links*. A bud-link is similar to a fringe link except that from an MI node there is only one possible bud-link (whereas there may be more than one in term of fringe links). Formally, given a network (G, S), a session multicast (s, D) and a tree T, a *bud-link* is an edge $\{x, y\}$ such that $x \in V_T$ and $y \notin V(T)$ and if $x \notin S$ (i.e. x is MI) then the degree of x in T is 1. Besides, at any time, a bud-link $\{x, y\}$ is the link that has the highest priority among those possible links leading to node y. Thus, when updating a bud-link $\{x, y\}$, if there are links $\{x_i, y\}$ already in the bud-link set (BLs), the selected link is the one that has highest priority, the other must be removed from the set. In addition, from an MI node (x), the only selected bud-link is the link with highest priority among possible adjacent links ($\{x, y_i\}$) of it.

Using the new concept of bud-links and applying the way they are updated as mentioned above to the network in Figure 6, the resultant light-forest including a single light-tree with the total cost of 9, the maximum delay of 3 and the average delay of 1.875 is created as shown in Figure 9.

However, like all the greedy heuristics, among many possible choices, only one that meets a specific condition can be chosen. Thus, this heuristic can lead to useless branches in the computed light-tree from MI nodes. These unnecessary branches need to be resolved by a special mechanism that is detailed in the next subsection.

D. Pruning Useless Branches

As mentioned above, with the defined priority of the bud-links, in many cases our heuristic could direct the tree following the same route for different loops for constructing light-tree, even though the algorithm can go to an unlimited loop with the useless route. To be clearer, let us consider the example in the topology in Figure 11. For the topology shown in Figure 11(a), according to the new definition priority described in section IV-B, the algorithm in turn selects bud-links (0,1), (0,2), (1,4), (2,5), (4,8), (4,9) for the first tree as shown in Figure 11(b). If nothing is done, the algorithm will stops and the first tree after pruning will be $\{0, 1, 4, 8\}$, and it is the only tree can be computed although there are two more destinations yet to be spanned. Because according to Member-First, the graph remains unchanged after each tree has been computed, just change the membership of destinations that have been spanned. Thus, in many cases, for the next tree formation, the tree is still grown in the same route as before, because there is no mechanism to change the route if it is still better under the control of priority.

Realizing the situation, we equip an efficient technique to re-direct the tree in such cases: when the tree cannot grow more, all the useless branches (branches that do not lead to any member) in the current tree as well as in the graph must be pruned. When affected branches cannot be pruned (e.i. when it reaches to a destination, or a MC branching node in the current tree, or a MI node that can lead to other nodes in the graph), we re-update the BLs from MI nodes (if any) at which these branches cannot be pruned.

Return to the above example, suppose that the algorithm chooses bud-link (1,4) before (2,5). After the algorithm selects bud-link (1,4), from node 4, links (4,8) and (4,9) can be added to the BLs, whereas from node 5, there is no more bud-link can be updated (the link (5,9) cannot be updated because its priority is not higher than bud-link (4,9) that is already in BLs). Besides, node 5 is not a destination, so node 5 and its adjacent links must be pruned from the tree as well as the graph. After pruning node 5, node 2 is taken into consideration, but from node 2 there is an other possible bud-link (2,6), so node 2 cannot be pruned. Besides, because node 2 is MI, the algorithm re-updates the BLs from it and hence bud-link (2,6) is added. The algorithm then adds bud-links (2,6), and (6,10) to the tree. It then chooses bud-links (4,8) and then (4,9) to add to the tree. Similarly, node 9 and its adjacent links are pruned from the tree and the graph. After all, the algorithm stops with the first tree as shown in Figure 11(c).

In order to compute the next tree, after constructing a tree, the algorithm must prune some parts of the graph that cannot grow any tree. For example, the sub-graph corresponding to the first tree in Figure 11(c) must be pruned. This can be described as follows: first, remove from the graph the nodes that are leaves in the current tree; after that, while there exists a leaf in the current tree, and it is also a pendent node of the graph, then remove it from the graph and the tree. Applying this to the example, nodes 8, 10, 4, 6, 2 (and their adjacent links) are in turn removed from the graph. The remaining part of the graph is shown in Figure 11(d). Thus, the second tree is then created as shown in Figure 11(e). This point is implemented as procedure *PruneGraph* as shown in the next section.

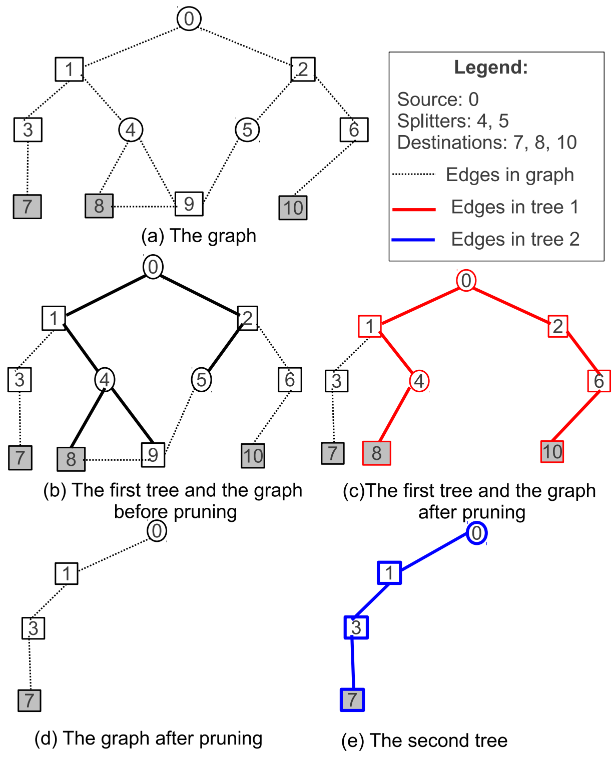


Fig. 11. Demonstration of pruning useless branches

E. *Member-Splitter First algorithm*

Combination of the above improvements results in the new algorithm, and we call it Member-Splitter First, or MSF in short. The resultant forest obtained when applying MSF to the network in Figure 6 is shown in Figure 10. As it is shown, this light-forest includes a single light-tree with the total cost of 10, the maximum delay of 2 and the average delay of 1.75. It is the best trade-off in comparison to the previous light-forests.

Our proposal can be described as below:

MEMBER-SPLITTER FIRST ALGORITHM

Input: A network (G, S) and a multicast session (s, D)
Output: A light forest F satisfying (s, D) in (G, S)

```
1:  $F \leftarrow \emptyset$; $G' \leftarrow G$; $D' \leftarrow D$; {$D'$ is set of destinations yet
    to be included}
2:  **while** $D' \neq \emptyset$ **do**
3:      $T \leftarrow (\{s\}, \emptyset)$
4:      **while** a bud-link exists **do**
5:          Choose a bud-link $e = \{u, v\}$ with highest priority
6:          Add $\{u, v\}$ and $v$ to $T$
7:          **if** $u \in D'$ **then** $D' \leftarrow D' - \{u\}$
8:          **while** there exists a dead-vertex[2] $v_d$ in $T$ **do**
9:              remove $v_d$ and its adjacent edges from $T$ and $G$
10:         **end while**
11:     **end while**
12:     add $T$ to $F$
13:     $T' \leftarrow T$
14:     PruneGraph($G', T'$);
15: **end while**
```

```
**Procedure** PruneGraph($G', T'$) {prune graph $G'$}
    delete from $T'$ and $G'$ the vertices that are leaves of $T'$
    **while** $\exists l$ such that $l$ is a leaf of $T'$ and $l$ is also a pendant
    vertex of $G'$ **do**
        delete $l$ from $T'$ and from $G'$
    **end while**
```

In the loop of line 4 the tree T grows as much as possible in the graph G' by choosing a highest priority link among all possible edges (the bud-links). The procedure *UpdateBL* for updating bud-links is described right away. During this time, the loop of line 8 prunes unnecessary branches of the tree that cannot grow more. Finally, when the tree is constructed and added to the forest, the procedure *PruneGraph* is invoked to delete vertices and edges from G' (and T') that are no longer necessary to continue the computation of the forest.

The correctness of the algorithm

The algorithm is correct and deterministic. Indeed, during the execution of the algorithm, the graph G' remains connected, since each deleted vertex is either a leave of a maximal branches of T' or a pendant vertex of G'. This ensures the correctness of the light-tree creation in each turn of the main loop. Moreover, every tree built has at least two vertices (s and a destination). Thus, at least one vertex of G' is deleted by the call of procedure *PruneGraph* in line 14. Since $D' \subset V'_G$ and that $|V'_G|$ strictly decrease, it is eventually empty and the algorithm stops.

Updating bud-links

The bud-link set is computed for each addition or removal of a vertex of the tree. The procedure *UpdateBL* can be described as follows. It is invoked whenever a new vertex is added to the tree, or from a vertex that cannot be removed when pruning useless branches.

[2]*dead-vertex* is a leaf of T that is not a destination nor end-node of any possible bud-links.

```
Procedure UpdateBL($v, L$){update bud-link set $L$ from $v$}
  for every adjacent link $e = (v, u)$ do
    if $\exists u \notin T$ and $\nexists e' = (v', u) \in L$
    or $\exists e' \in L$ has lower priority then
      if $v$ is MC then
      $L \leftarrow L \cup \{e\}$;
      if $\exists e' \in L$ has lower priority then $L \leftarrow L - \{e'\}$;
      else find $e^*$ with highest priority among $e$;
    end if
  end for
  if $v$ is an MI node then
    $L \leftarrow L \cup \{e^*\}$;
    if $\exists e' \in L$ has lower priority then $L \leftarrow L - \{e'\}$;
  end if
```

An example and explanation of the execution of the algorithm is shown in Figure 12.

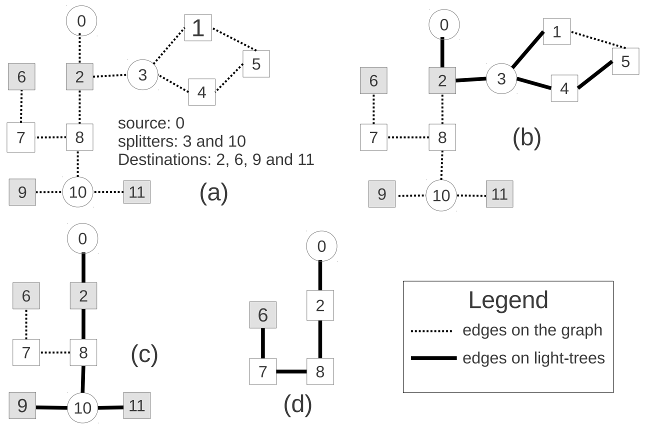


Fig. 12. Fig(a) represents a network and a multicast session. Fig(b) shows the first tree in which vertices 3, 1, 4 and 5 are not destinations. These vertices have been removed by the loop of line 8. Then the procedure *UpdateBL* is invoked to re-update the bud-link from MI node 2. The light-tree continue to cover 8, 9, 10 and 11 as shown in Fig(c). It is the actual first light-tree. After that, vertices 9, 11 and then 10 are deleted by procedure *PruneGraph*. Finally, a second tree including 0, 2, 6, 7 and 8 is created as shown in Fig(d).

V. SIMULATION RESULTS

To evaluate the proposed algorithm in comparison to the other classical ones, we carry out series of simulations with well-known networks: USA NSF network (Figure 1), USA Longhaul network (Figure 13) and European Cost-239 network (Figure 14). The fact that these networks are the testbeds of many studies ([1], [9], [11]) is the reason for our selection.

In our simulation, each node of the network is in turn selected as the source for a multicast session. Because the source can inject data traffic using multiple transmitters, hence it can transmit to as many of its out-ports on the same wavelength even if it is an MI node. Similarly, the source can also transmit traffic to its children on different wavelengths even if it has no wavelength conversion. Consequently, we will consider the source to be capable of both splitting and wavelength conversion [10]. Besides, although the source can be considered as a multicast member, it is not a destination.

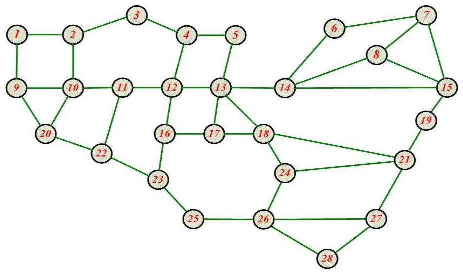


Fig. 13. USA Longhaul network topology

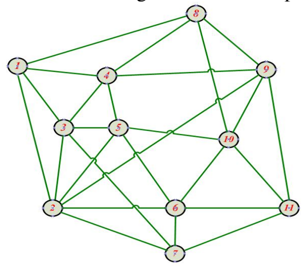


Fig. 14. European Cost-239 network topology

For these reasons, in our simulations, the source will not be counted in the group size nor in the fraction of MC nodes. In other words, given an N-node network, the group size and MC nodes are selected from N-1 remaining nodes. For a given source, a given multicast group size, and a given fraction of MC nodes, 100 random multicast sessions are generated. (The destinations and MC nodes are distributed randomly through the network.) Hence, the result of each point in the simulation figures is the average of $100 \times |V|$ computations on the four metrics mentioned above: *link stress*, *total cost*, *maximum delay* and *average delay*.

Effect of Group Size (the fraction of destinations)

Firstly, we study the performance of the proposed algorithm versus multicast group size: the number of MC nodes is set fixed while the group size varies. For the sparse splitting capacity of network, a few MC nodes can be set.

For 14-node NSF network, the number of MC nodes is set at 3 nodes (˜23%) and the group size is varied from 1 to 13 nodes. MC nodes and destinations are distributed randomly in the network. The performance metrics are calculated and plotted in Figure 15. When the group size varies, the link stress (Figure 15(a)) achieved by MSF remains flat around the value of 1. It is much better compared to MF, especially when the group size becomes large. As shown in the Figure 15(a), when the group size is 100%, the difference between the two is maximal at about 50%. In Figure 15(b), the total cost produced by all the algorithms increases when the group size increases. In particular, MSF provides a better total cost than MF. When group size becomes larger, MSF is close to the optimal resulted from MO. In Figure 15(c) and Figure 15(d), the end-to-end delay (maximum deday and average delay) resulted from MSF is also lower than that from MF (as well as Re2A and MO).

In 28-node USA Longhaul network, the fraction of MC nodes is set as 30% and the group size is varied from 10% to

100%. As shown in Figures 16, the results achieved with this network topology is quite the same that with NSF network. In particular, the link stress is minimal (the same MO's), the total cost is slightly better then MF, the maximum delay and the average delay are better than MF when the group size is high.

In 11-node European Cost-239 network, it is very dense when all the nodal degrees are higher or equal to 4. Like other network topologies, we set a few nodes as MC nodes (2 nodes, or 20%) and set the group size changing from 1˜10, or 10%˜100%. In Figure 19, the link stress resulted from MSF is minimal, similar to that from MO and Re2A, the maximum delay and average delay is close to the optimal delay of Re2S and much better than MF while the total cost is the same as MF.

In short, with the sparse splitting capacity, when the group size varies, MSF algorithm outperforms MF. Among all the algorithms, it produces the lowest link stress, a low total cost and a low end-to-end delay. Especially, its performance is better when the group size is large. Moreover, MSF is more advantageous with dense networks (like European Cost-239 network).

Effect of Splitting Capacity (the fraction of MC nodes)

We also study the performance of the proposed algorithm versus the number of MC nodes. Previously, we see that MSF works better with the large group size. Thus, the group size is set at high value. In NSF network, the group size is set at 10 nodes (˜77%) while the number of MC nodes varies from 1 to 13 nodes. The simulation results with NSF network are shown in Figure 17.

According to Figure 17(a), while the link stress of the other algorithms converge to 1 in skew lines from left to right, MSF and MO keep the link stress flat at the value around 1, in which MSF is better than MO when the number of MC nodes is very sparse. For the total cost and the end-to-end delay, (Figure 17(b), 17(c)), while MO achieves the lowest total cost but the highest delay and Re2S, in contrary, produces the lowest delay but the highest total cost, MSF provides a good trade-off between the two metrics when always ranking second in those metrics. Compared to MF, it is always better, and the difference between the two is clearer when the number of MC nodes is sparse.

In Figure 18, almost the same result above can be seen with USA Longhaul network, except that the end-to-end delay of Re2A is better than MSF when the fraction of MC nodes is very small. It is because there are 7 nodes in this network with degree of 2. These nodes are useless in term of splitting capacity even though they are set as MC nodes, but the setting probably reduces the fraction of MC nodes reserved for the other nodes. The worst situation can be happened when all of these 2-degree nodes are set as MC nodes in very sparse splitting settings (less than or equal to 7 MC nodes or 25%).

In European Cost-239 network, the result is better as shown in Figure 20. As we can see, with the large group size (8 nodes or 80%), MSF appears the best one when achieving the minimal link stress (the same as MO and Re2A, Figure 20(a), slightly better total cost than MF (Figure 20(b), nearly the optimal end-to-end delay as Re2S (Figure 20 c,d).

In short, when the group size is set at high value, MSF achieves the best link stress, a good total cost and end-to-end delay. Especially, MSF works best in dense networks (e.g. European Cost-239 network).

VI. Conclusion

In this paper the multicast routing problem in sparse splitting WDM networks are investigated and well-known algorithms for light-tress construction and their performance are also reviewed. The comparison results show that the algorithms offer various performance on the important metrics as cost, delay and generated link stress. To ensure a best trade-off among these metrics, we proposed a new algorithm called Member-Splitter First (MSF in short). It is based on the framework of the known Member-First algorithm. Our proposal exploited the two very important elements in MF: the priority definition of network links and the way of constructing light-tree. MSF associates higher priority to MC nodes and avoids MI nodes with high degree. To compare and evaluate the performance of our proposed algorithm, number of simulations have been carried out with all the algorithms. Simulation results show that MSF outperforms MF in all the performance metrics. Particularly, MSF achieves the lowest link stress (among all the algorithms), a low total cost (ranks second below the optimal produced by MO) and a low end-to-end delay (also ranks second below the optimal resulted from R2S). In general, MSF provides a good trade-off among performance metrics in sparse splitting WDM networks and with the large number of destinations. Besides, MSF works better in dense networks (like European Cost-239 network).

References

[1] M. Ali and J. S Deogun. Allocation of splitting nodes in all-optical wavelength-routed networks. *Photonic Network Communications*, 2:247265, 2000.

[2] M. Ali and J. S Deogun. Cost-effective implementation of multicasting in wavelength-routed networks. *EEE/OSA Journal of Lightwave Technology*, 18:16281638, 2000.

[3] J. Elmirghani and Mouftah. All-optical wavelength conversion: technologies and applications in DWDM networks. *IEEE Communications Magazine*, 38:8692, 2000.

[4] A Hamad. *All optical multicasting in wavelength routing mesh networks with power considerations: design and operation.* PhD thesis, Iowa State University, 2008.

[5] Xiao-Hua Jia, Ding-Zhu Du, and Xiao-Dong Hu. Integrated algorithms for delay bounded multicast routing and wavelength assignment in all optical networks. *Computer Communications*, 24:1390–1399, 2001.

[6] Hwa-Chun Lin and Sheng-Wei Wang. Splitter placement in all-optical wdm networks. In *GLOBECOM'05*, pages 306–310, 2005.

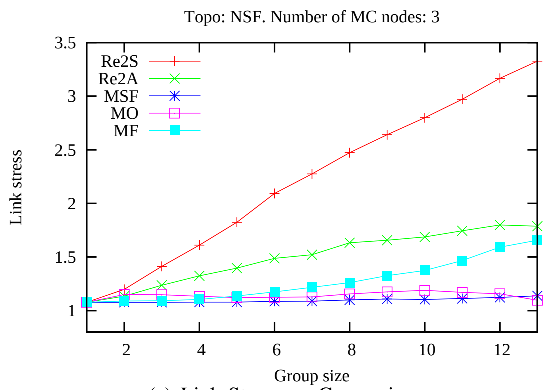
[7] Biswanath Mukherjee. *Optical WDM Networks*. Springer, 2006.

[8] D. Nesset, T. Kelly, and D Marcenac. All-optical wavelength conversion using SOA nonlinearities. *IEEE Communications Magazine*, 36:5661, 1998.

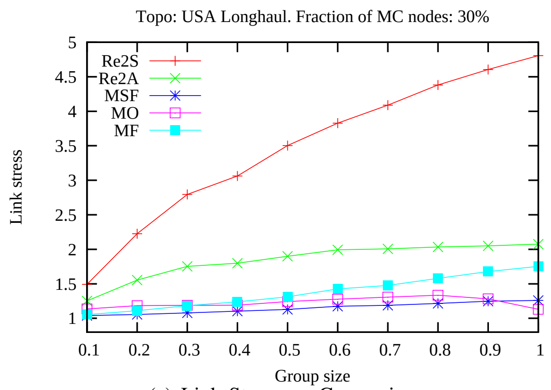
[9] Xiong Wang, Sheng Wang, and Lemin Li. A novel efficient multicast routing algorithm in sparse splitting optical networks. *Photon Netw Commun*, 14:287295, 2007.

[10] Xijun Zhang, John Wei, and Chunming Qiao. Constrained multicast routing in WDM networks with sparse light splitting. *IEEE/OSA Journal of Lightwave Technology*, 18:1917–1927, 2000.

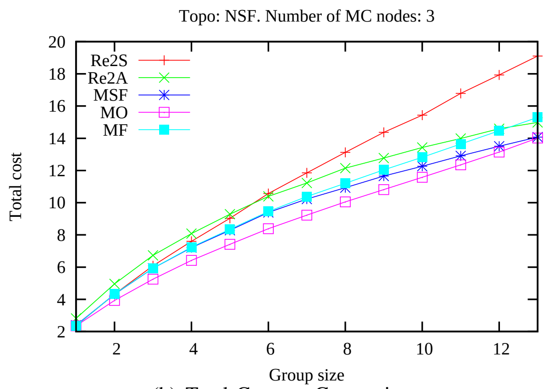
[11] Fen Zhou. *Routage multicast tout optique dans les rseaux WDM*. PhD thesis, INSA-UR1, 2010.



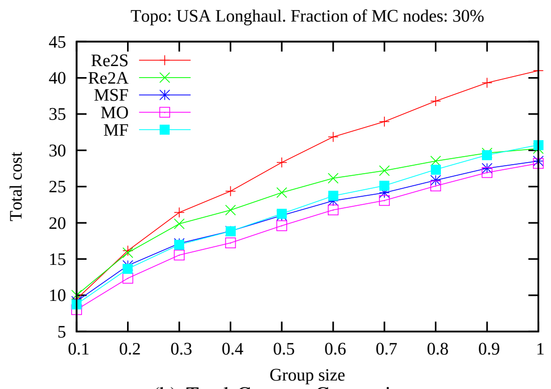
(a) Link Stress vs. Group size



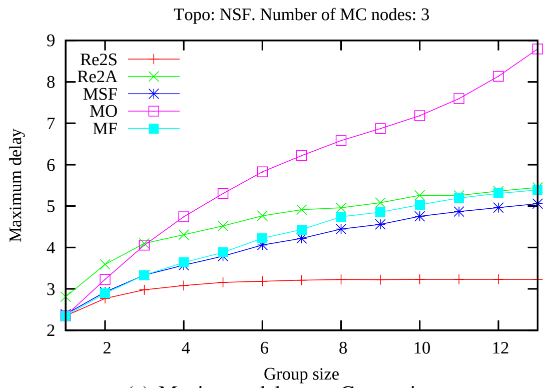
(a) Link Stress vs. Group size



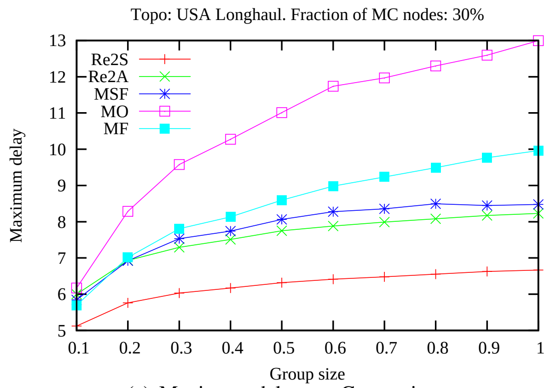
(b) Total Cost vs. Group size



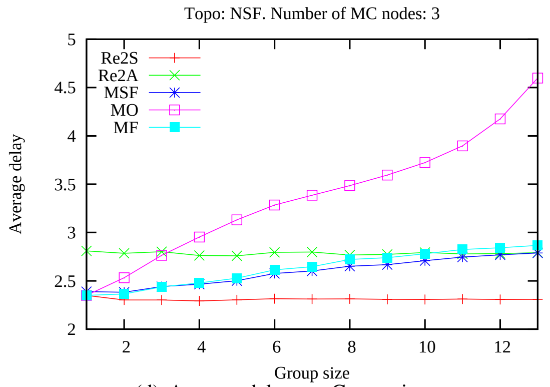
(b) Total Cost vs. Group size



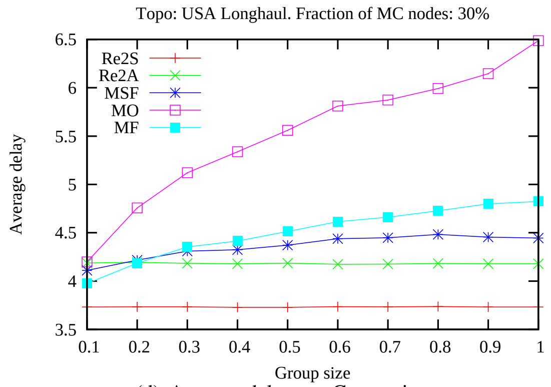
(c) Maximum delay vs. Group size



(c) Maximum delay vs. Group size



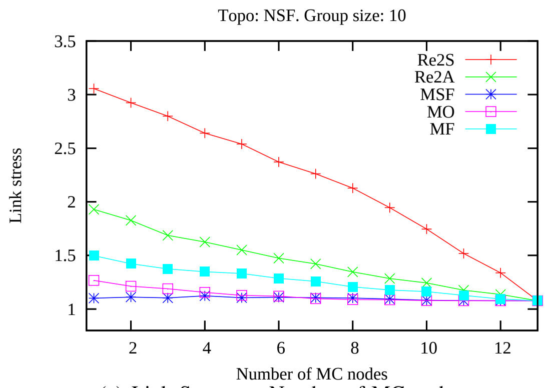
(d) Average delay vs. Group size



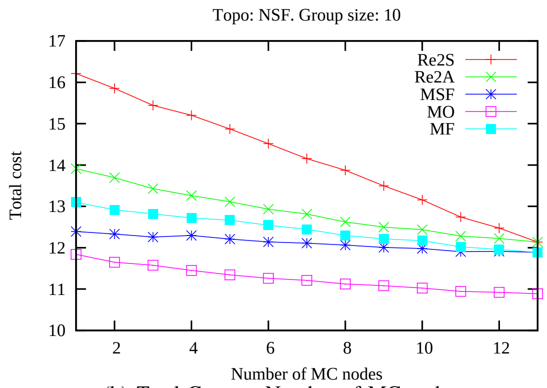
(d) Average delay vs. Group size

Fig. 15. Performance of algorithms vs. Group size in NSF network with 3 random MC nodes

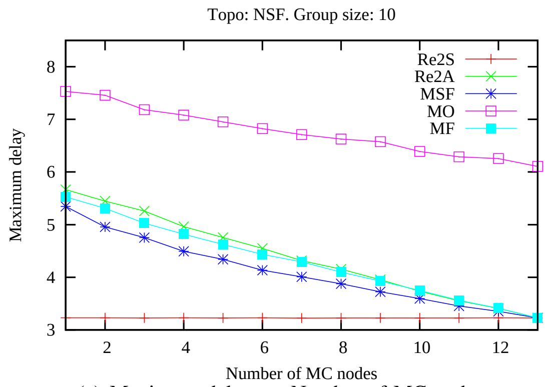
Fig. 16. Performance of algorithms vs. Group size in USA Longhaul network with 30% random MC nodes



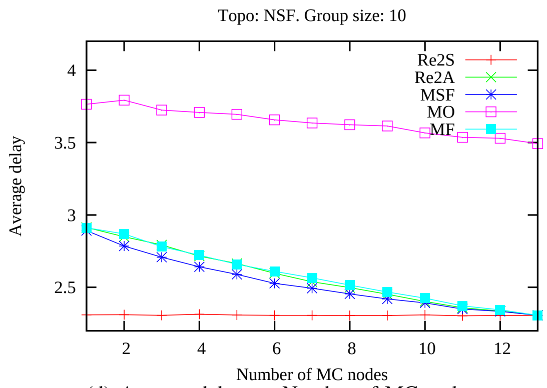
(a) Link Stress vs. Number of MC nodes



(b) Total Cost vs. Number of MC nodes

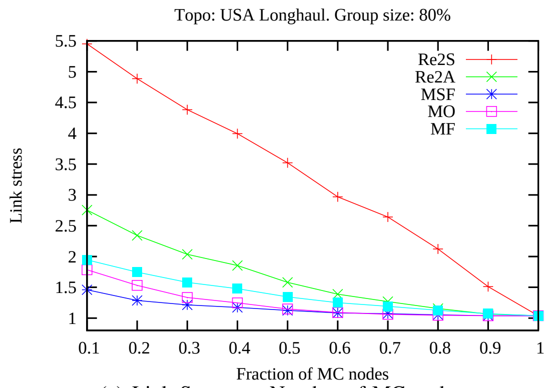


(c) Maximum delay vs. Number of MC nodes

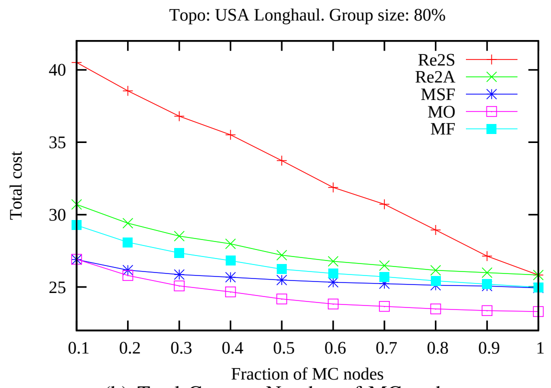


(d) Average delay vs. Number of MC nodes

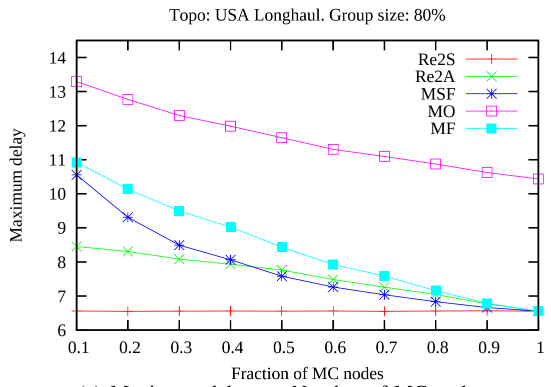
Fig. 17. Performance of algorithms vs. Number of MC nodes in NSF network with group size of 10



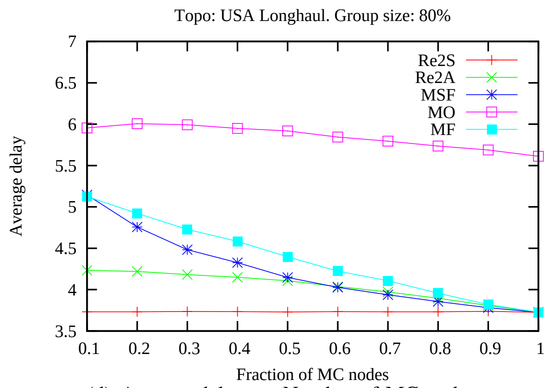
(a) Link Stress vs. Number of MC nodes



(b) Total Cost vs. Number of MC nodes

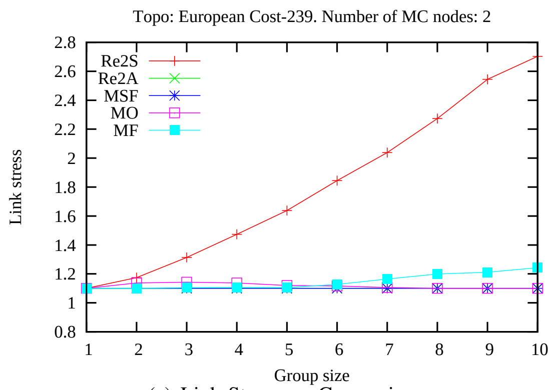


(c) Maximum delay vs. Number of MC nodes

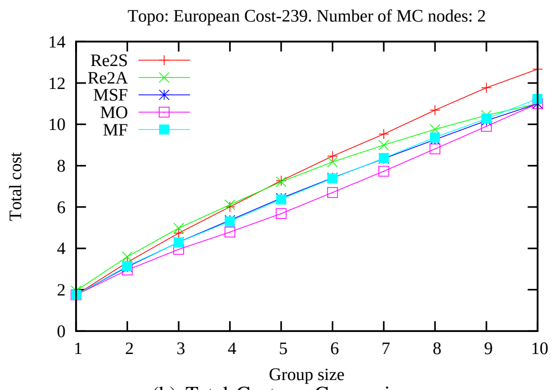


(d) Average delay vs. Number of MC nodes

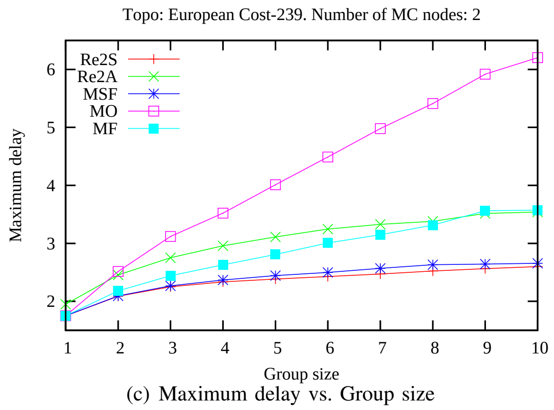
Fig. 18. Performance of algorithms vs. Number of MC nodes in USA Longhaul network with group size of 80%



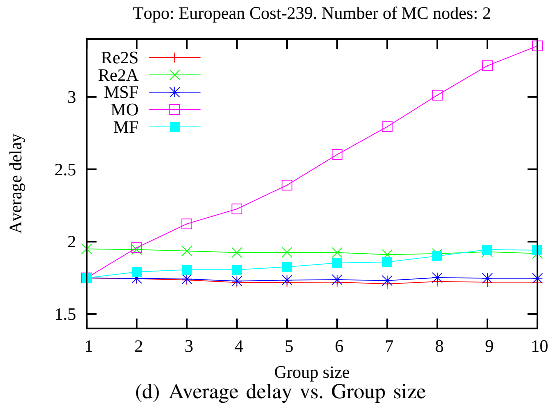
(a) Link Stress vs. Group size



(b) Total Cost vs. Group size

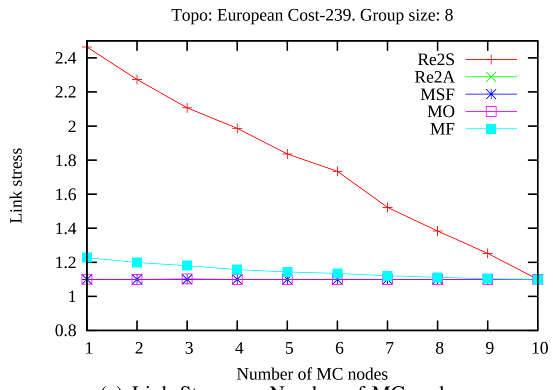


(c) Maximum delay vs. Group size

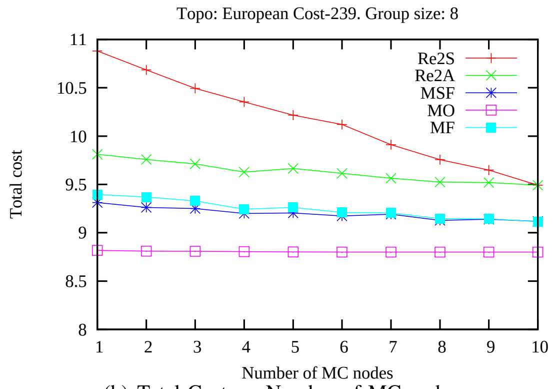


(d) Average delay vs. Group size

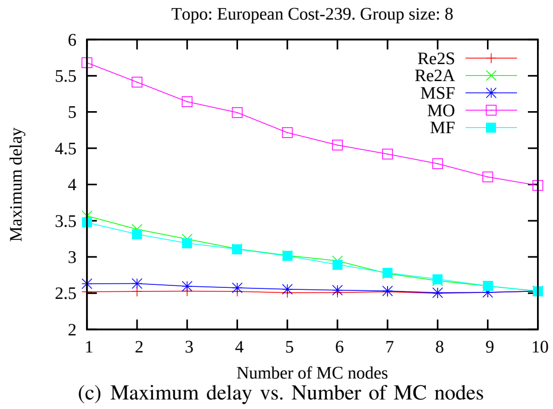
Fig. 19. Performance of algorithms vs. Group size in European Cost-239 network with 2 random MC nodes



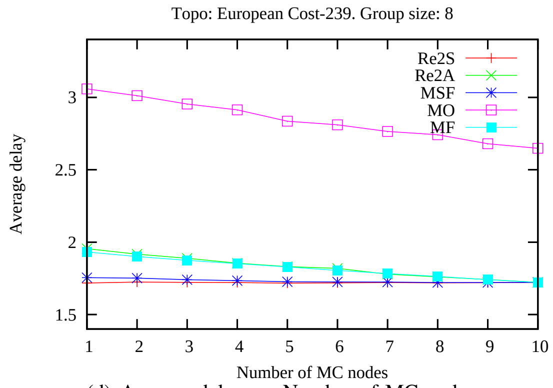
(a) Link Stress vs. Number of MC nodes



(b) Total Cost vs. Number of MC nodes



(c) Maximum delay vs. Number of MC nodes



(d) Average delay vs. Number of MC nodes

Fig. 20. Performance of algorithms vs. Number of MC nodes in European Cost-239 network with group size of 8