International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2021/209185 Volume 8, Issue 3, May – June (2021)

RESEARCH ARTICLE

Protected Light-Tree Reconfiguration without Flow Interruption in Elastic Optical Networks

N’takpe N’guessan Christian
Computer and Telecommunications Research Laboratory, National Institut Félix Houphouët Boigny, Yamoussoukro, Ivory Coast.
nguessan.ntakpe18@inphb.ci

Adepo Joel Christian
Digital Research and Development Unit, Virtual University of Ivory Coast, Abidjan, Ivory Coast.
joel.adepo@uvci.edu.ci

Babri Michel
National Institut Félix Houphouët Boigny, Yamoussoukro, Ivory Coast.
michel.babri@gmail.com

Received: 11 March 2021 / Revised: 27 April 2021 / Accepted: 09 May 2021 / Published: 28 June 2021

Abstract – Reconfiguration is one of the most important capabilities of optical networks. This task is performed by operators to re-optimize network resource utilization. Multicast applications such as videoconference online learning, etc., emerge rapidly and cause an increase in bandwidth demand. A multicast connection in an optical network is provided by a light-tree. Due to physical link failures, light-trees are protected by backup paths. Thus, our study focuses on the reconfiguration of a protected light-tree. The reconfiguration problem studied here is to migrate rapidly an optical flow from an old light-tree to a new light-tree without service disruption. To solve this problem, we propose a mechanism based on dependencies graphs. This method establishes simultaneously groups of the final light-tree branches by using shared backup paths spectrum resources. The numerical results show that our approach avoids flow interruptions and reduce significantly the number of steps required to achieve the reconfiguration process more than existing methods.

Index Terms – Optical Network, Reconfiguration, Multicast, Protection, Survivable, Light-Tree.

1. INTRODUCTION

1.1. Context

The ever-increasing demand for bandwidth-intensive multicast applications such as UHD-TV (Ultra-high-definition television), distance learning, online gaming, makes multicast services are provided more on the optical layer of networks than on the IP layer. One of the main solutions to meet the new traffic characteristics is the elastic optical network (EON). Compared to its predecessor, Wavelength Division Multiplexing (WDM), EON has finer spectrum channel spacing called “slot”. This feature allows bandwidth (spectrum) to be allocated according to the data rate required by a connection, without having to waste a portion of a large channel spectrum, as in the WDM standard [1, 2]. A unicast connection in an optical network is provided by a "lightpath"[3]. In the case of a multicast connection, the path followed by an optical connection from a source S to a set of destinations D = {d_1,...,d_n} is called "light-tree" [4]. A light-tree can be considered as a tree rooted at the source node where the leaf nodes of the tree are the destinations. In this context, a lightpath from the source to one of the destinations is called a branch. Since a single link failure may disrupt several downstream destinations and cause huge data loss, backup paths mechanisms have been implemented to protected light-tree against link failure in the network. The most advantageous of the protection scheme in terms of spectrum resource use is shared backup path protection comparing to the dedicated backup path protection approach [5]. In the shared backup paths mechanism, any primary lightpaths (or branches) can share the same backup capacity (slots) in a physical link as long as they do not traverse the same physical link. Due to traffic load, fragmentation, and network condition changes, the current virtual topology on the network needs to be reconfiguring in order to maximize the total network throughput [6]. This task is called reconfiguration. The optical network reconfiguration process involves establishing new lightpaths (or light-trees) and deleting existing lightpaths (or light-trees), where new lightpaths (or light-trees) and old lightpaths (or light-trees) have the same sources and the same destinations [7]. Usually, the news paths (or trees) are pre-calculated such that the targeted topology is known. Since spectrum resources are not always available to establish new lightpaths, reconfiguration should be conducted in a hitless way, i.e., the existing service would not be disrupted due to existing lightpath teardown, and the reconfiguration process should be as short as possible to reduce the impact at the end-users. For performing properly the reconfiguration process should meet the below-mentioned requirement:

- Minimize flow interruption: when the number of disrupted lightpaths is reduced, generally fewer services are disrupted, fewer end users are affected.
- Minimal steps of process: Minimal steps of process: when the time of the reconfiguration process is reduced, the lost user traffic carried by lightpaths is reduced.
- Minimal additional spectrum resources: to allow the network to accept new connections during the process.

1.2. Objectives

The purpose of this paper is to provide a mechanism to reconfigure an initial light-tree use to transmit a data flow to a final pre-computed light-tree quickly by using shared backup path resources without flow disruption. For the sake of concise, the network resource considered here is the spectrum resource (slots). The specific objectives of this research work are: (i) to model light-tree reconfiguration problem, (ii) to avoid flow interruption during the reconfiguration process, (iii) to reduce the reconfiguration process steps that impact the process duration, (iv) to reduce the network resources used.

1.3. Motivation

This paper focuses on the reconfiguration of protected light-tree. Migrate from an initial light-tree to a final light-tree involve establishing the new branches on the final light-tree and deleted the old branches one the initial light-tree. This process should be conducted without flow disruption and as short as possible. This objective is only achievable if all the spectrum resources needed to establish the entire new branches on the final light-tree are available on the network. Unfortunately, most of the time, the setup of a new branch may require spectrum resources of a link already used by an old branch in the initial light-tree. The reconfiguration of such a branch requires spectrum resources of the old branch to be released. Due to these spectrum resource conflicts, solving reconfiguration problems is a challenge in the literature. To solve this problem, some works propose to use backup paths in an intermediate step. In this step, the flow of some old branches is shifted to their backup paths in order to free their spectrum resources. However, these works focus on unicast connections, while multicast connections are an alternative for deploying applications for multiple users. Using shared protection backup paths optimally for reconfiguration is complex and NP-hard [8]. Because protection resources are permanently shared by several different primary connections, we cannot shift all primary connections to their backup paths, delete old connections and establish new ones simultaneously without disrupting the flow of most of them. The complexity of using shared protection backup paths increases with multicast connections since paths in the primary tree often share the same links. The deletion of a branch, therefore, results in flow interruption in all branches with which this share at least one link [9]. Thus, several steps are required to judiciously use shared backup protection paths to reconfigure a light tree while reducing the number of process steps without interrupting the flow.

1.4. Organization of the Paper

The current section of the paper has briefed about reconfiguration, along with problem statement, objective, motivation, and contribution. Section 2 provides a review of some research work on reconfiguration techniques. Section 3 presents a formal definition of the problem of reconfiguring multicast connections with shared protection resources. LSP_Reconf (Light-tree reconfiguration algorithm with Shared backup Paths) algorithm reconfiguration is proposed in section 4. In Section 5, the provided algorithm is tested and the results are analyzed and discussed. Finally, Section 6 concludes.

2. RELATED WORK

Reconfiguration techniques can be grouped into two main approaches: Make Before Break (MBB) and Break Before Make (BBM). Service disruptions are significant when an old optical path (or branch) is first removed before the new one is established. This is the case in the "Break Before Make (BBM)" approach, where the reduction of flow interruptions is limited by the intrinsic capabilities of the network [10]. Compared to the BBM method, the "Make Before Break (MBB)" approach significantly reduces service disruption. Network resources are requisitioned and allocated to the new path (or branch) [11]. Thus, the new path (or branch) can be established before the old one is deleted. As can be seen, such reconfiguration significantly reduces service disruptions, but at the cost of increased use of network resources. Nevertheless, the resources to set up the news lightpaths (or branches) are not always available since most of the time some resources to be used to establish the new branches (link, slot) are already allocated to the old branches. In this case, the flow of the old branch must be interrupted, before the new one is established which causes service disruption. To modelling, the resource dependencies between new lightpaths and old lightpaths a directed resources dependency graph are used in [12].

In this graph, a vertex is a lightpath and an arc between vertex A and vertex B means that the new lightpath corresponding to vertex A needs resources from the old lightpath corresponding to vertex B for been reconfigured. It showed that if the directed dependency graph is acyclic (DAG), an order exists to reconfigure the set of lightpath without interrupt one of them and without use spare resources. If a cycle exists on the directed dependency graph, the lightpath interruption is necessary to complete the reconfiguration process. In this case minimization algorithms are used to reduce lightpath interruption or to minimize spare [13] resources used. However, this work is done in unprotected unicast connections, while additional dependencies are to be considered in multicast connections. In [14] the authors have presented an algorithm of reconfiguration without flow interruption using the shared protection resources. This algorithm based on a graph colouring algorithm minimizes the number of steps of the reconfiguration process. However, it concerns unicast connections while the instances of our problem only concern light-trees. Most multicast connection reconfigurations are performed in unprotected networks. This has an impact on the use of spectral resources as well as on the number of interrupted connections.

In [15] and [16], the authors showed that MBB and BBM are not suitable for light-tree reconfiguration and proposed a multicast reconfiguration scheme without flow interruption. These two methods, BpBAR_1 [15] and BpBAR_2 [16] consist of migrating from a current light-tree to a new light-tree by reconfiguring pairs of branches using nodes wavelength conversion capabilities. One branch on the current light-tree and the other on the final light-tree. Although it allows reconfiguration without lightpaths interruption, BpBAR_1 generates a long reconfiguration time which BpBAR_2 improves by performing a series of operations in parallel. These two methods [15] and [16] perform a reconfiguration without flow interruption. However, they require that all nodes in the network have the wavelength conversion capability. In addition, the two trees (current and final trees) are not protected by backup paths while, our problem concerns reconfiguration of protected, in an optical network where no node has the wavelength conversion capability. These two prerequisites mean that the algorithms proposed in [15] and [16] are unsuitable for our problem.

The authors in [17] propose a reconfiguration of a set of multicast connections in a WDM optical network having a single wavelength. In this approach, the dependencies between the multicast connections (current light-trees and new light-trees) are modeled through a weighted dependency graph, and then an MFVSA (Minimum Feedback Vertex Set Algorithm) is computed to minimize the number of flows interrupted. This approach, in addition to producing interruption, concerns a set of unprotected multicast connections. However, our goal is to reconfigure a single protected light-tree using the protection resources. Therefore, the approach proposed in [17] cannot be the solution to our problem. Usually, reconfiguration techniques are not suitable for multicast connections. However, the existing methods use intrinsic network resources, which increase network congestion and, the reconfiguration process duration. The use of protection resources can be a solution to save intrinsic network resources and reduce process duration if they are used wisely. Our goal is to migrate from an initial light-tree to a final light-tree, by using existing protection resources to avoid flow interruption and reduce the duration.

3. PROBLEM MODELLING

3.1. Network Model

Let a physical network topology represented by a graph G = (V, E) where V represents the set of nodes and E the set of links. A node in the graph is a variable bandwidth optical switch (BV-WXC) in the physical network and a link in the graph represents a fiber in the physical network. Let $T_0 = (V_0, E_0)$ and $T_z = (V_z, E_z)$ two trees. T_0 is the tree carrying the current flow (initial tree) and T_z is the new tree to which to migrate (final tree). The two trees T_0 and T_z have the same source and the same set of destinations nodes included in G. L_0 the set of shared backup paths from T_0 given by the network. We assume that each link in T_0, T_z, and L_0 has the same continuous and contiguous slots on each of their links. An optical path from the source to one of the destinations of a tree (initial or final) is called a branch (resp. initial or final). Backup paths of each branch of T_0 can share backup spectrum resources (slots) as long as the branches they protect are links disjoint according to the SBPP principle [18]. In addition:

- As in [19] all nodes in the network are multicast capabilities (MC) but without frequency conversion capabilities
- We assume failure-free during the reconfiguration process, i.e. all shared backup resources are available for the reconfiguration purpose. The resource considered here is slots (spectrum resources).

3.2. Problem Formulation

The problem is formulated as follows:

Given:

- A current light-tree (T_0, S, D, N, L_0), {S: source, D: set of destination nodes, N: slots required, L_0: set of backup paths for T_0} where $b_0(S, d_k)$ is a current branch from source S and destination node d_k); (T_z, S, D, N) a final light-tree with the same source and the same set of destinations nodes than T_0, where $b_z(S, d_k)$ is the new branch from source S and destination node d_k corresponding to $b_0(S,d_k)$.

Goal:

- Migrate from T_0 to T_z quickly and without flow interruption using shared backup paths if necessary.

Constraints:

- Backup paths can share protection spectrum resources (slots) if the branches they protect are links disjoints.
- If a branch is deleted or interrupted, it causes flow interruption in all branches that share at least one link with it.
- If a new branch (or backup path) is established, then the resources to establish new branches (or backup paths) that shares at least one link with it become unavailable. It is a deadlock.

An interruption here is a destination that is not served from the source at each step of the reconfiguration process. A step here is one of the following reconfiguration operations: pre-establishing, establishing or deleting a lightpath. Three topologies are to be considered in our problem T_0, T_z, and L_0. Therefore, the flow from source S to a destination d_k is interrupted if there is no lightpath from S to d_k in T_0 or in L_0 or in T_z at each step. In this case, we say that the branch $b_0(S, d_k)$ is interrupted either:

$$INT^i\_(d_k) = \begin{cases} \mathbf{1}\ if\ a\ lightpath\ from\ S\ to\ d_k\ does\ not\ exist \\ in\ the\ i^{ith}\ step\ either\ in\ both\ T_0\ or\ L_0\ or\ T_z \\ \mathbf{0}\ otherwise \end{cases} \quad (1)$$

Num_INT^i, the total number of branches interrupted at the ***i^ith^*** step:

$$Num\_INT^i = \sum_{d_k \in D} INT^i\_(d_k) \quad (2)$$

Suppose that the reconfiguration process is performed in ***m*** steps, therefore the total number of interruption during the process, Tot_Num_INT is equal to:

$$Tot\_Num\_INT = \sum_{i=0}^{m} Num\_INT^i \quad (3)$$

Since our goal is to avoid any interruptions during the reconfiguration process, we need to make sure that at each step we have:

$$Num\_INT^i = 0 \quad (4)$$

Let a group $\mathbf{g_i}$ of branches to be reconfigured in parallel.

If the backup paths are not used during the reconfiguration process (i.e. resources for setup all branches of the group are available), the operations performed are:

- Simultaneously pre-establishing the new branches of $\mathbf{g_i}$ from the source,
- Simultaneously establishing them,
- And simultaneously deleting the old branches of group $\mathbf{g_i}$

The number of steps Num_STEP according to the process operations is:

$$Num\_STEP = 3 \quad (5)$$

If the backup paths are used during the reconfiguration process (i.e. resources for setup some new groups are already used by old branches), the operations performed are:

Simultaneously pre-establishing the dependencies backup paths,

- Simultaneously establishing the dependencies backup paths,
- Simultaneously deleting the old branches of dependencies,
- Simultaneously pre-establishing the new branches of $\mathbf{g_i}$ from the source,
- Simultaneously establishing them,
- Simultaneously deleting the old branches of group $\mathbf{g_i}$ and the backup paths used.

In this case, the number of steps Num_STEP according to the process operations is:

$$Num\_STEP = 7 \quad (6)$$

In conclusion, the number of steps for a given reconfiguration instance is related to resource dependencies. If there are resource dependencies then we get 7 steps. If there are no resource dependencies then 3 steps are necessary to reconfigure the light-tree.

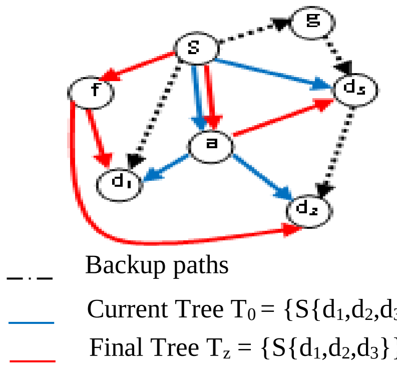
S
g
f
a
d₁
d₂
d₃
Backup paths
Current Tree T₀ = {S{d₁,d₂,d₃}}
Final Tree T_z = {S{d₁,d₂,d₃}}

Figure 1. An Instance of the Reconfiguration Problem. The Currents Branches (Lightpaths) on T_0 are {{S-a-d_1},{S-a-d_2},{S-d_3}}, and the News Branches on T_z {{S-f-d_1},{S-f-d_2}, {S-a-d_3}}. d_2 and d_3 Shared the Same Backup Resources(Link, Slots) on Links S-g and g-d_3

$$\begin{tikzcd} d_1 & & d_2 \\ & d_3 \arrow[ul] \arrow[ur] & \end{tikzcd}$$

Figure 2 Directed Dependency Acyclic Graph of the Instance of Light-Tree Reconfiguration Problem in Figure 1.

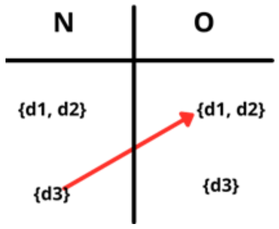

Figure 3 Bipartite Graph G_d=(N,O,E) Shows Resources Dependencies Between New Groups of Branches and Old Groups of Branches

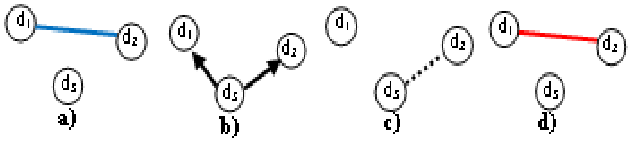

Figure 4 Dependencies Graph of the Instance of Light-Tree Reconfiguration Problem in Figure 1 (a) Initial Tree Dependency, Graph (b) Resources Dependency Graph, (c) Backup Paths Dependency Graph (d) Final Tree Dependency Graph

3.3. Problem Description

The reconfiguration of a lightpath (i.e. a branch) consists of first pre-establishing the new branch, then establishing the new branch, and finally deleting the old one.

According to [20] if the dependency graph is a DAG (Directed Acyclic Graph) reconfiguration schedule exists allowing reconfiguring each lightpath without spare resources. However, considering the resource dependency graph in figure 2 of the reconfiguration instance of figure 1 no order allows reconfiguring the branches of the initial tree T_0 without flow interruption. Indeed, the reconfiguration process cannot be started from the branch {S-d_3} since the resources to set up the new branch S-a-d_3 are already used by the old branches {S-a-d_1} and {S-a-d_2}, on the link S-a. Similarly, if the process starts with branch {S-a-d_2}, although the resources are available on the new branch {S-f-d_2}, the deletion of the old path {S-a-d_2} induces flow interruption on old branch {S-a-d_2}, since {S-a-d_1} and {S-ad_2} share the same link (cf. S-a). Finally, if the reconfiguration process is initiated with {S-a-d_1}, the deletion of the old path {S-a-d_1} induces flow interruption on old branch {S-a-d_2}. In additional the setup of it new branch {S-f-d_1} create a deadlock for the new branch {S-f-d_2} since {S-f-d_1} and {S-f-d_2} share the link S-f. It is obvious that a resource dependency-oriented graph alone cannot correctly model our problem because other dependencies have to be considered. When backup paths are used as transient paths to handle the dependencies modelled in Figure 2, the flows of {S-a-d_1} and {S-a-d_2}, must be switched to their backup paths before established the new branch {S-a-d_3}. Such an operation also causes interruptions. Indeed, if {S-a-d_1} is first toggle on her backup path {S-d_1}, the deletion of the primary path {S-a-d_1} interrupts the flow to {S-a-d_2} ({S-a-d_1} and {S-a-d_2} share the link S-a). If {S-a-d_2} is the first to be shifted on her backup path {S-g-d_3-d_2}, the deletion of its primary path {S-a-d_2} interrupts the flow to {S-a-d_1} (({S-a-d_1} and {S-a-d_2} share the link S-a). Therefore, {S-a-d_1} and {S-a-d_2}, must be simultaneously shifted on their backup paths {S-d_1} and {S-g-d_3-d_2} and simultaneously deleted in the initial tree T_0 to prevent any interruption of the flow.

However, if the branches are divided into a group we can find a sequence to migrate from T_0 to T_z without any flow interruption. Instead of considering resource dependencies between branches of the final tree and the initial tree that cause interruptions, we consider resource dependencies between groups of branches of the final tree and groups of branches of the initial tree. Thus two branches of the final tree belong to the same group if they share at least one link, and two branches of the initial tree belong to the same group if they share at least one link. According to the instance of figure 1, the new branches {S-f-d_1} and {S-f-d_2} belong to the same group and the new branch {S-a-d_3} belongs to another group. The old branches {S-a-d_1} and {S-a-d_2} are in the same group and {S-d_3} in another group. Now we can construct a bipartite resources dependency graph G_d = (N,O,E) where N contain all the new groups and O all the old groups and there is an arc between two groups (a group of new branches and a group of old branches) if the first needs resources already used par by the second for being established. In addition, the backup path of a group is the set of backup paths of all primary branches belonging to this group.

If we designate a branch by its destination node we obtain two groups in the final tree: g_1^{new} ={d_1 ,d_2} and g_2^{new} = {d_3} and two groups in the initial tree: g_1^{old} ={d_1,d_2} and g_2^{old} = {d_3}. Graph G_d of the instance of figure 1 is represented in figure 3. In this case, the group g_2^{new} needs the resources from the group g_1^{old} for being reconfigured. Therefore the group g_2^{new} is shifted from the source S on his backup paths (cf {S-g-d_3} and {S-g-d_3-d_2}}, then g_1^{new} and g_2^{new} can be established in parallel before backup paths {S-g-d_3},{S-g-d_3-d_2} and the old branch {S-d_3} are deleted in parallel.

So, all dependencies relationships between branches and between backup paths induced by a shared protected light-tree reconfiguration instance must be highlighted in order to make the best use of the backup resources and for reconfiguring a light-tree with shared backup paths quickly without flows interruption. To solve this problem while minimizing the interruption, the dependency relationships will be modelled as a graph (cf. Figure 4), and an algorithm is provided for reconfiguring a protected light-tree without flow interruption, which reduces the processing time.

4. PROPOSED SCHEME

The previous section showed that a resource dependency graph alone is not sufficient to model all the dependencies of our problem. This is why an algorithm based on four dependency graphs is proposed in this section in order to take into account all the dependencies of the problem, which will allow us to avoid flow interruptions and reduce the time of the reconfiguration process.

4.1. Dependency Modeling and Notation

4.1.1. Initial Tree Dependency

A dependency between two branches of the initial tree implies that these two branches share at least one link. So the deletion of one leads to the interruption of flow in the other. We model the whole of this sort of dependency through a digraph which we call "initial tree dependency graph". This graph is denoted by $G_{T_0}(V_d, E_{T_0})$ where each vertex in V_d represents a branch in the initial tree and there is a link between two nodes of the graph if their corresponding branches share at least one link. In this graph, two related vertexes belong to the same group. It allows to determinate all the groups of old branches (cf. Figure 4-a).

4.1.2. Resources Dependency

A dependency of resources between two branches means that the first needs the resources (link, slots) of the second to be established. We model this relationship by a directed graph that we call the "resource dependency graph". This graph is denoted by $G_R(V_d,E_R)$ thus, a vertex in the graph represents a branch in the final tree and there is an arc in the graph between two vertices A and B if the new branch corresponding to A needs the resources of the old branch corresponding to B to be established. An example of this graph is found in figure 2 where the new branch {S-a-d_3} representing by the vertex d_3 needs the resources of the old branches {S-a-d_2} and {S-a-d_1} representing respectively by the vertex d_1 and d_2 for being established. It allows determining all the resource dependencies of a reconfiguration instance and constructs the bipartite graph G_d.

4.1.3. Backup Paths Dependency

A backup path dependency between two branches of the initial tree implies that they share at least one link. So, the establishment of one backup path before the other creates a deadlock for the second backup path since they share the same resources. We model such a relationship through a digraph that we call the "backup path dependency graph". This graph is denoted by $G_{L_0}(V_d,E_{L_0})$ and, a vertex in the graph represents a branch in the initial tree and two vertices are related in the graph if their respective backup paths share at least one link. An example is showed in figure 4-c. The vertex d_2 and d_3 are related because their respective backup paths {S-g-d_3} and {S-g-d_3-d_2} share at least on link. This graph allows finding the backup paths of each old group.

4.1.4. Final Tree Dependency

A dependency of the final tree between two branches of the final tree implies that these two branches share at least one link. So the establishment of one before the other creates a deadlock. We model this type of relationship by a digraph that we call the "final tree dependency graph". This graph is denoted by $G_{T_z}(V_d,E_{T_z})$ where each vertex in V_d represents a branch in the final tree and there is a link between two vertices of the graph if their corresponding branches share at least one link. This graph allows to determinate all the groups of new branches (cf. Figure 4-c) need to be reconfigured in parallel. So, in this graph two related vertex means that their corresponding groups belong to the same group.

4.2. Proposed Algorithm

In this section of our paper, an algorithm to reconfigure a light-tree with shared protection (LSP_Reconf) Algorithm 1 is proposed. In, this algorithm resource dependency is not considered between two branches, but between two groups of branches (cf. between new branches group and old branches groups), then some of the old groups are shifted together on their backup paths to process the dependencies of the resources. Finally, all the new groups are established in parallel. The algorithm takes as input:

- An initial tree T_0, a final tree T_z, and the set of
- Backup paths L_0 of T_0, given by the network.

And as output:

- A series of steps $S = (s_0,....,s_k)$ allowing to migrate from T_0 to T_z quickly and without flow interruption

The LSP_Reconf algorithm is described as follows: (a) First construct the 4 graphs of section 4.1 i.e. the "initial tree dependency graph", the "resources dependency graph", the "final tree dependency graph", the "backup path dependency graph". (b) According to the final tree dependency graph, divide the new branches into groups such that branches in the same group share at least one link. Let $G_{new} = \{g_0^{new},\ldots,g_n^{new}\}$ a set of group where g_i^{new} is a group of branches of the final tree sharing at least one link. (cf. line 4 Algorithm 1) (c) According to the initial tree dependency graph divide the old branches into groups such that branches in the same group share at least one link. Let $G_{old} = \{g_0^{old},\ldots,g_n^{old}\}$ where g_i^{old} is a group of branches of the initial tree sharing at least one link (cf. line 4 Algorithm 1). (d) Build according to the resource dependency graph a bipartite graph G_d= (N, O, E) where a vertex in N represents a group element of G_{new}, a vertex in O represents an element of G_{old} and there is an arc between a vertex x_i of N and a

RESEARCH ARTICLE

vertex y_j of O if the group g_i^{new} corresponding to x_i needs the resources of at least one branch of the group g_j^{old} corresponding to y_j for being reconfigured. (cf. line 5 Algorithm 1). The rest of the algorithm is based on the bipartite graph G_d and allows reconfiguring in parallel the groups of branches without interruption of flow, without additional resources and quickly (line 5 to line 27). (e) To start with, we remove from the set of old branch groups O of the graph G_d all the groups whose resources are not needed for the reconfiguration process. (line 6 to line 13 Algorithm 1). (f) Also, remove from the groups of new branch N all new groups that do not need to be reconfigured (line 6 to line 13). (g) An update of the graph G_d is performed (line 14). (h) If set O is empty after the update then no backup paths will be used to migrate from T_0 to T_z. In this case, pre-establish in parallel all the new groups of set N, establish in parallel all the new groups of set N, and delete their old branches in parallel (line 15 to 19). (i) If, on the other hand, the set O is not empty, then the set O contains the old groups whose resources are needed for the reconfiguration process. In this case, shift all the groups of set O simultaneously to their backup path to free the resources (line 21 to line 23). Then establish in parallel all the groups of set N and simultaneously delete their old branch and the set of backup paths established in 23 (line 24 to 26). The algorithm ends with the construction of the new rescue paths of the final tree T_z.

Algorithm: light-tree reconfiguration algorithm with shared backup paths LSP_Reconf

```
1:Begin
2:    If $T_0 = T_z$ Go to line 27 // new light-tree and old light-
      tree are equal
3:        If ($T_0 = T_z$ )
4:                Determine the set of groups $\boldsymbol{G_{new}}$
          and $\boldsymbol{G_{old}}$
5:                Construct $G_d$= (N, O, E)
6:                Remove all vertices have degree equal to 0
          from O
// Their corresponding groups resources are not necessary for
the //reconfiguration process
7:                For each $x_i$ from the set N and $y_j$ from the
                  set O
8:                        If a group $x_i = y_j$ and vertex $y_j$ has
a                         degree equal to 1
9:                                Remove $x_i$ from $\boldsymbol{G_{new}}$
10:                               Remove $y_j$ from $\boldsymbol{G_{old}}$
//Remove their corresponding groups from
11:                       End If
12:               End For
13                Update $\boldsymbol{G_{old}}, \boldsymbol{G_{new}}$
14:              If the set $\boldsymbol{G_{old}}$ is empty
// No dependencies exist
15:                               Pre-establish simultaneously all the
                                  groups of $\boldsymbol{G_{new}}$
16:                               Establish simultaneously all the
                  groups of $\boldsymbol{G_{new}}$
17                                Delete simultaneously all the old
                  branches of all the groups of $\boldsymbol{G_{new}}$
18:               End If
19:               If the set $\boldsymbol{G_{old}}$ is no empty
// Resources dependencies exist
20:                               Pre-establish simultaneously all the
                                  backup paths of all the groups
                        $\boldsymbol{G_{old}}$
21:                               Establish simultaneously all the
                  backup paths of all the groups
          of $\boldsymbol{G_{old}}$
22:                               Delete all the branches of all the
                  groups of $\boldsymbol{G_{old}}$
23:                               Pre-establish all the groups
                  of $\boldsymbol{G_{new}}$
24:                               Establish all the groups of $\boldsymbol{G_{new}}$
25:                               Delete the old branches of $\boldsymbol{G_{new}}$
          and the backup paths Establish in 22
26:               End If
27: Building the new backup paths of $T_z$
28: End
```

Algorithm 1 Light-Tree Shared Protection Reconfiguration Algorithm

5. RESULTS AND DISCUSSIONS

5.1. Simulation Setting and Performance Metrics

To assess the performance of LSP_Reconf in elastic optical networks, simulation experiments were employed using two existing topologies (US-backbone and the pan-European COST-236) and the FlexGridSim [21] simulator. The metrics used here for comparison are:

- The total number of flow interruption to the destination nodes of a light-tree (cf. equation 3),
- The number of the reconfiguration process steps that impact the process time (cf. equation 5 and 6)
- The number of additional resources assigned to the reconfiguration process.

An instance of our light-tree reconfiguration problem takes as parameters: an initial light-tree $T_0(S,D)$, a final light-tree $T_z(S,D)$ with the same source and destination set as T_0, and a set of shared backup paths L_0 of initial light-tree T_0. For each topology (cf. US-backbone and the pan-European COST-236), two experiments are conducted. In the first experiments, 2500 instances are generated where the initial light-tree is an approximation of the Minimum Steiner Tree (MST) and the final light-tree is the Shortest Paths Tree. In the second experiment, another 2500 instances are generated where the initial light-tree is the Shortest Path Tree (SPT) and the final tree is an approximation of the Minimum Steiner Tree (MST), from source to the destinations. The proposed algorithm (i.e LSP_Reconf) is run for each instance. Finally, an average of the different metrics is computed over the 2500 processed instances on each topology used. An instance of the problem is generated as follows:

- The initial light-tree: For each multicast connection, the source, destinations, and the number of destinations are randomly selected from the network nodes. The light-tree is obtained by using SPT or MST algorithm according to experiments conducted and used 5 slots in each link.
- The final light-tree: For each multicast connection, the source, the destinations, and the number of destinations are the same as the initial light-tree. The light-tree is obtained by using SPT or MST algorithm according to experiments conducted and used 5 slots in each link.
- Shared Backup paths: For each branch of the initial light-tree a shared backup path is constructed using Dijkstra algorithm and SBPP algorithm [22]. Each shared backup path used 5 slots in each link.

The proposed algorithm (LSP_Reconf) is compared with Bp_Bar2 and MBB1, the other light-tree reconfiguration scheme with the same parameters.

5.2. Performance Evaluation

5.2.1. Number of Flow Interruption Analysis

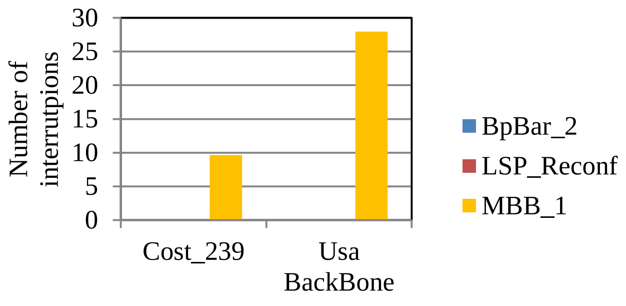

Figure 5 Comparison of the Average Number of Flow Interruptions

Figure 5 clearly shows that our LSP_Reconf algorithm and the Bp_Bar2 algorithm do not produce any flow interruption during the reconfiguration process like Bp_Bar2, unlike the MBB1 algorithm. This is due that the MMB1 algorithm does not use any other spectrum resources apart from the ones available in the network in its light-tree reconfiguration approach. The corresponding value of Figure 5 is provided in Table 1.

Topologies	COST 239		USA BACK.	
Reconf. Process	AVG	Min/Max	AVG	Min/Max
BpBar_2	0	0/0	0	0/0
LSP_Reconf	0	0/0	0	0/0
MBB_1	9,63	0/18	27,97	0/315

Table 1 Average Number of Interruption of the Reconfiguration Process

5.2.2. Additional Resources Analysis

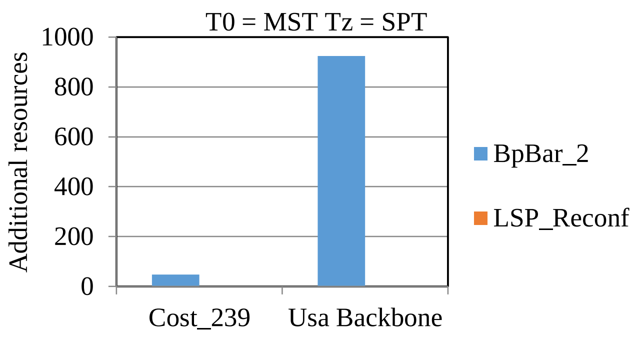

Figure 6 The Average Additional Resources (T_0=MST, T_z= SPT)

Topologies	COST 239		USA BACK.	
Reconf. Process	AVG	Min/Max	AVG	Min/Max
BpBar_2	47,96	2/319	924,54	2/4358
LSP_Reconf	0	0/0	0	0/0

Table 2 Additional Cost of the Reconfiguration Process T_0 =MST and T_z =SPT

Figure 6, and figure 7 show the average number of additional resources used in the reconfiguration process. In figure 6, the initial light-tree is a MST and final light-tree is a SPT. In figure 7 the initial light-tree is a SPT and final light-tree is a MST. An additional resource is a spectrum resource (slots) intended to establish incoming connections that are diverted to meet the lack of spectrum resources for the reconfiguration process. From figure 6, and figure 7 it can be identified that LSP_Reconf required no additional resources. Backup paths spectrum resources used during LSP_Reconf cannot be considered as additional resources because, under normal operation, the backup resources do not carry any traffic. On the other hand, Bpbar_2 generates additional resources because it uses wavelength converters to overcome the lack of resources in its approach. So LSP_Reconf is more hitless Bpbar_2 considering network resources. The corresponding values of figures 6 and 7 are provided respectively in Table 2 and 3.

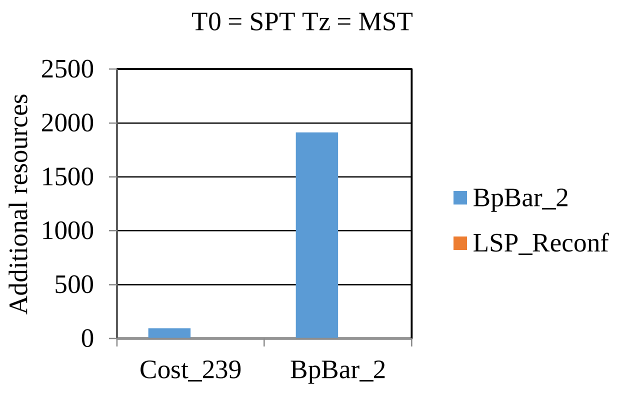

Figure 7 The Average Additional Resources (T_0=SPT, T_z= MST)

Topologies	COST 239		USA BACK.	
Reconf. process	AVG	Min/Max	AVG	Min/Max
BpBar_2	94,04	4/640	1911,71	6/10793
LSP_Reconf	0	0/0	0	0/0

Table 3 Additional Resources of the Reconfiguration Process T_0 =SPT and T_z =MST

5.2.3. Number of Steps Analysis

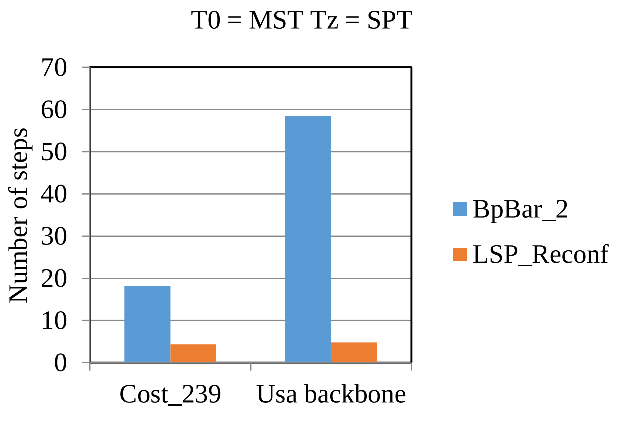

Figure 8 Average of the Number of Steps (T_0 =MST and T_z=SPT)

Topologies	COST 239		USA BACK.	
Reconf. Process	AVG	Min/Max	AVG	Min/Max
BpBar_2	18,21	6/61	58,48	6/153
LSP_Reconf	4,32	0/6	4,77	0/6

Table 4 Duration of the Reconfiguration Process (Number of Steps) T_0 =MST and T_z=SPT

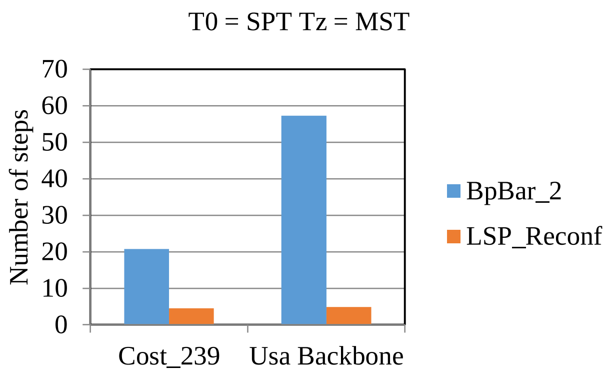

Figure 9 Average Number of Steps (T_0 =SPT and T_z=MST)

Topologies	COST 239		USA BACK.	
Reconf. process	AVG	Min/Max	AVG	Min/Max
BpBar_2	20,75	6/43	57,24	6/130
LSP_Reconf	4,5	0/6	4,9	0/6

Table 5 Duration of the Reconfiguration Process (Number of Steps) T_0 =MST and T_z=SPT

The reconfiguration process should be as short as possible in order to quickly re-optimize all network resources and reduce the impact on the end-user. So speed is an essential metric to assess the reconfiguration scheme. Here the process duration is the number of steps necessary to perform the process. Figures 8 and 9 performed a comparison of the average of the number of steps. In figure 8, the initial light-tree is an MST and final light-tree is a SPT. In figure 9 the initial light-tree is a SPT and final light-tree is a MST. It shows that LSP_Reconf takes between 70% and 90 % fewer steps than Bpbar_2. BpBar_2 does not group the branches and performs several sequential operations on the nodes. This makes Bpbar_2 need more steps than LSP_Reconf. In another hand LSP_Reconf group the branches to be reconfigured and setup them simultaneously, which reduces significantly the number of steps. The corresponding values of figures 6 and 7 are provided respectively in Tables 4 and 5.

Topologies	Cost 239	USA Back.
T_0 = SPT , T_z = MST	68%	87%
T_0 = MST , T_z = SPT	88%	91%

Table 6 Ratio of Backup Resources Used by LSP_Reconf

6. CONCLUSION

This paper presents a novel light-tree reconfiguration scheme that use shared backup paths spectrum resources. A modified bipartite dependency graph was constructed to modelled the light-tree reconfiguration problem. The spectrum resources dependencies between new branches and old branches are extended to dependency between groups of new branches and groups of old branches. This scheme reconfigures in parallel the groups of branches in the final light-tree. The metrics considered here are first the number of flows interruption, then the number of additional spectrum resources used, and finally the number of steps. These three metrics were evaluated through a reconfiguration algorithm propose and compare with MBB_1 and BpBar_2. The results show that our proposed algorithm (i.e LSP_Reconf) avoids flow interruptions and considerably reduces the number of steps than MBB_1 and BpBar_2. Nevertheless, even if no additional spectrum resources are mobilized in the reconfiguration process, the shared backup paths spectrum resources used to conduct the process still huge compared to all existing resources spectrum (between 70% and 90% cf. table 6). This last point makes the algorithm energy-intensive and unable to effectively handle link failure during the reconfiguration process. A lower protection resource ratio could be a solution to this problem.

REFERENCES

[1] Baojia Li, Wei Lu, and Zuqing Zhu, "Deep-NFVOrch: leveraging deep reinforcement learning to achieve adaptive vNF service chaining in DCI-EONs," J. Opt. Commun. Netw. 12, A18-A27 (2020).

[2] J. Velinska, I. Mishkovski and M. Mirchev, "Routing, Modulation and Spectrum Allocation in Elastic Optical Networks," 2018 26th Telecommunications Forum (TELFOR), 2018, pp. 1-4, doi: 10.1109/TELFOR.2018.8611929.

[3] I. Chlamtac, A. Ganz and G. Karmi, "Lightpath communications: an approach to high bandwidth optical WAN's," in IEEE Transactions on Communications, vol. 40, no. 7, pp. 1171-1182, July 1992, doi: 10.1109/26.153361.

[4] C. Xue et al., "Light-tree based multicast flow aggregation scheme in elastic optical datacenter networks," 2017 16th International Conference on Optical Communications and Networks (ICOCN), 2017, pp. 1-3, doi: 10.1109/ICOCN.2017.8121379

[5] Shen, G., Guo, H. & Bose, S.K. Survivable elastic optical networks: survey and perspective (invited). Photon Netw Commun 31, 71–87 (2016). https://doi.org/10.1007/s11107-015-0532-0.

[6] Y. Lee, B. Mukherjee, Traffic engineering in next-generation optical networks, IEEE Communications Surveys and Tutorials 6 (3) (2004) 16–33. Third Quarter.

[7] Golab, W., & Boutaba, R. (2004). Policy-driven automated reconfiguration for performance management in WDM optical networks. IEEE Communications Magazine, 42(1), 44–51.

[8] Y. Chiu and D. Din, "Survivable Virtual Topology Reconfiguration Problem on WDM Networks with Reconfiguration Constraint," in 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA), Chengdu, 2009 pp. 211-218.

[9] JC.Adépo "Multicast routing reconfiguration in WDM opticalnetworks", Thesis,"Université Nangui Abrogoua", 2016, p42.

[10] H. Duong, B. Jaumard, D. Coudert and R. Armolavicius, "Efficient Make Before Break Capacity Defragmentation," 2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR), Bucharest, Romania, 2018, pp. 1-6, doi: 10.1109/HPSR.2018.8850754.

[11] F. Solano, "Analyzing Two Conflicting Objectives of the WDM Lightpath Reconfiguration Problem," GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference, Honolulu, HI, USA, 2009, pp. 1-7.

[12] F. Balmas, 'Displaying dependence graphs: a hierarchical approach', in Proceedings Eighth Working Conference on Reverse Engineering, Stuttgart, Germany, 2001, pp. 261– 270, doi: 10.1109/WCRE.2001.957830.

[13] Cohen, N., Coudert, D., Mazauric, D., Nepomuceno, N., & Nisse, N. (2011). Tradeoffs in process strategy games with application in theWDMreconfiguration problem. Theoretical ComputerScience, 412(35), 4675–46.

[14] Xin, Yufeng et al. "Reconfiguration of survivable IP over WDM networks." Opt. Switch. Netw. 21 (2016): 93-100.

[15] B. Cousin, J. C. Adépo, S. Oumtanaga, and M. Babri, 'Tree reconfiguration without lightpath interruption in WDM optical networks', Int. J. Internet Protoc. Technol., vol. 7, no. 2, pp. 85–95, 2012.

[16] J. C. Adépo, B. Aka, and M. Babri, 'Tree Reconfiguration with Network Resources Constraint', Int. J. Comput. Sci. Telecommun., vol. 7, no. 1, pp. 1–4, Jan. 2016.

[17] Amanvon Ferdinand Atta, Joël Christian Adépo, Bernard Cousin, " Minimize Flow Interruptions during Reconfiguration of a set of Light-trees in All-optical WDM Network ",International Journal of Computer Science and Network Security, VOL.20No.7,July2020.

[18] C. -F. Hsu, H. -C. Hu, H. -F. Fu, J. -J. Zheng and S. -X. Chen, "Spectrum Usage Minimization for Shared Backup Path Protection in Elastic Optical Networks," 2019 International Conference on Computing, Networking and Communications (ICNC), 2019, pp. 602-606, doi: 10.1109/ICCNC.2019.8685656

[19] K. Walkowiak, R. Goścień, M. Klinkowski and M. Woźniak, "Optimization of Multicast Traffic in Elastic Optical Networks With Distance-Adaptive Transmission," in IEEE Communications Letters, vol. 18, no. 12, pp. 2117-2120, Dec. 2014, doi: 10.1109/LCOMM.2014.2367511.

[20] N. Jose and A. Somani, "Connection rerouting/network reconfiguration," in IEEE Design of Reliable Communication Networks (DRCN), Banff, Canada, Oct. 2003, pp. 23–30.

[21] H. M. N. S. Oliveira and N. L. S. Da Fonseca, "Protection, Routing, Modulation, Core, and Spectrum Allocation in SDM Elastic Optical Networks," in IEEE Communications Letters, vol. 22, no. 9, pp. 1806-1809, Sept. 2018, doi: 10.1109/LCOMM.2018.2850346.

[22] H. M. N. S. Oliveira and N. L. S. da Fonseca, "Algorithm for shared path for protection of space division multiplexing elastic optical networks," 2017 IEEE International Conference on Communications (ICC), 2017, pp. 1-6, doi: 10.1109/ICC.2017.7997378.

RESEARCH ARTICLE

Authors

N'takpe N'guessan Christian - received his Master deegre in Computer Science at université Nangui Abrogoua (Côte d'Ivoire) in 2015. Currently, PhD Student in Computer Science in Laboratoire de Recherche en Informatique et telecommunication, Ecole Doctorale Polytechnique (EDP) in Yamoussoukro (INP-HB, Côte d'Ivoire). His research interests include Reconfiguration and Survivability in Optical Networks.

Joël Christian Adépo - received his PhD in Computer Science at Université Nangui Abrogoua (Côte d'Ivoire) in 2016. Currently, he is an assistant professor in Computer Science at Université virtuelle (Côte d'Ivoire) and also member of the Laboratoire de Recherche en Informatique et Télécoms (LARIT) since 2010 his research interests include routing reconfiguration.

Michel BABRI - received his PhD in Computer Science from University Clermont-Ferrand. Since 1995, he has been a researcher at INPHB of Yamoussoukro in Cote d'Ivoire and also a Director of the Laboratoire de Recherche en Informatique et Télécoms (LARIT) since 2012.

How to cite this article:

N'takpe N'guessan Christian, Adepo Joel Christian, Babri Michel, "Protected Light-Tree Reconfiguration without Flow Interruption in Elastic Optical Networks", International Journal of Computer Networks and Applications (IJCNA), 8(3), PP: 140-150, 2021, DOI: 10.22247/ijcna/2021/209185.