Tree Reconfiguration without Lightpath Interruption in Wavelength Division Multiplexing Optical Networks with Limited Resources

Bernard Cousin[1], Joël Christian Adépo[2], Michel Babri[3] and Souleymane Oumtanaga[3]

[1] IFSIC, Université de Rennes 1, IRISA
[2] Sciences Fondamentales et Appliquées, Université Nangui Abrogoua, LARIT
Abidjan, Côte d'Ivoire
[3] Mathématique et Informatique, INHPB, LARIT
Abidjan, Côte d'Ivoire

Abstract

Today, operators use reconfiguration to improve the performance of connection oriented networks. In our previous work, we studied tree reconfiguration without data flow interruption in wavelength division multiplexing optical networks. In our previous approach, the available resources of the network (wavelengths) are not considered as limited. In this work, we study tree reconfiguration in a network which has a limited number of wavelengths per link. This paper proposes an algorithm called *TRwRC (*tree reconfiguration with resources constraint). *TRwRC* reconfigures a tree without data flow interruption in a network with a limited number of wavelengths.

Keywords*: Path reconfiguration, optical WDM network, multicast tree, reconfiguration sequence, lightpath interruption.*

1. Introduction

Path reconfiguration is used by network operators to improve network performance. It consists of moving from an initial path to a new one. Since networks are disturbed by failures, overloads, and deployments of new network resources or maintenance operations, network reconfiguration becomes very important because it enables the fulfillment of QoS requirements with the available network resources. In our previous work [1], we presented an efficient algorithm which achieves tree reconfiguration in WDM optical networks in which a multicast connection is established. It was the first work on that problem in the literature. The objectives of this work are: first, tree reconfiguration without data flow interruption; second, reduction of the latency between the triggering event and the effective reconfiguration of the nodes on the paths by the tree reconfiguration process; and third, reduction in the resources used during the reconfiguration by the tree reconfiguration process. In our previous work, we proposed a branch-by-branch process called BpBAR_2. In that approach, the number of available wavelengths per link was not considered as a constraint. At each step of the reconfiguration process, the proposed algorithm computed a light-forest to span all the destinations. Since the network resources are supposed to be unlimited, the number of trees per light-forest is also unlimited. In this current work, the number of available wavelengths per link is a constraint. So, the number of trees per light-forest produced at each step should be limited. This constraint first allows the tree with the available network resources to be reconfigured, and second, contributes to optimizing the network resources used. The problem addressed in this paper is tree reconfiguration without data flow interruption in a network with a limited number of available wavelengths per network link. The proposed algorithm must maintain the continuity of the data flow toward all the destination nodes and reduce the cost of resources used and the duration of the reconfiguration process.

Section 2 and 3 describe respectively the multicast connection in WDM optical networks and the tree reconfiguration in WDM optical networks. Section 4 presents the related works. In Section 5, we prove the continuity of data flows during the reconfiguration with the BpBAR_2 process. Section 6 presents the context of tree reconfiguration with limited resources and the proposed algorithms called TRwRC. Section 7 presents the evaluation criteria and the results of the evaluation. This algorithm allows tree reconfiguration in a network which has limited resources and it reduces the resource cost during the reconfiguration. Finally, we conclude this article in the section 8.

2. Multicast Connection in WDM Optical Networks

The path of a connection in WDM optical networks [2] consists of optical channels. An optical channel in a WDM optical network is an optical signal transmitted over one wavelength in one fiber. The ends of an optical channel consist of an output port of a switch called the source port

and an input port of another switch called the sink port. If the source port belongs to a node, then this node is the source node of the optical channel. If the sink port belongs to a node, then this node is the sink node of the optical channel.

A lightpath is an all-optical path on the same wavelength between two nodes: a lightpath is made of a set of successive optical channels. The set of paths used by a multicast connection [3]-[4] in a WDM optical network forms a tree called a light-tree. A light-tree is an all-optical structure on the same wavelength between a source node and a set of destinations nodes. The light-tree is obtained with the use of particular nodes which are capable of duplicating an incoming light signal from one node input port into one or several node output ports. In this work, we assume that all network nodes have this light splitting capability. In [5], the reader may find some interesting considerations when this assumption is not fulfilled, and may find some efficient computation heuristics or protocols in [6] and [7]. A structure called a light-forest is used when a single light-tree is not sufficient to cover all destinations. A light-forest is a set of light-trees originated at the same source node but established with different wavelengths. For example, $T_0^{l_0}$ and $T_0^{l_1}$ are the two light-trees of the light-forest in Fig. 1 between the source node *s* and the destinations $\{d_1, d_2, d_3\}$. $T_0^{l_0}$ is established with a wavelength l_0 between the source node s and the destination $\{d_3\}$. $T_0^{l_1}$ is established with a wavelength l_1 between the source node *s* and the destination $\{d_1, d_2\}$. A branch $Br(T,s,d)$ of the light-forest is a lightpath between the source node *s* and the destination d on the tree *T*.

In a WDM optical network, before transmitting data from a source *s* to a destination *d*, a connection has to be set up first along the path from *s* to *d*. This connection is established by allocating a wavelength on each link along the connection's path and each node is instructed to switch the wavelength transparently through its switch fabric. We called this path with the allocated wavelength a pre-established path. A pre-established path between node *y* and the destination *d* is denoted $p(y,d)$. $p(s,d_3)$ in yellow in Fig. 1 is a pre-established path between *s* and d_3.

If there is no confusion, a tree $T_i^{l_0}$ will be noted T_i and a pre-established $p(y,d)$ will be noted *p*.

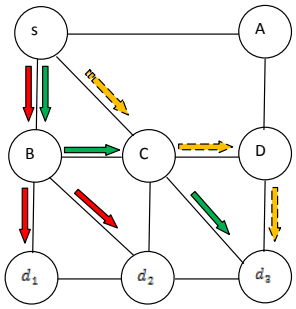

Fig. 1. Light-forest between the source node *s* and the destinations $\{d_1, d_2, d_3\}$ and the pre-established path (in yellow) between *s* and d_3.

The key characteristic of the multicast in optical networks is that an optical channel can be shared to feed multiple output ports and reach many destinations. In Fig. 1, the optical channel between *s* and B is shared to reach the destinations d_1 and d_2. B becomes a branching node. An optical channel which has B as a sink node is an upstream optical channel of B. An optical channel which has B as a source node is a downstream optical channel of B.

The number of upstream optical channels of a node *x* on a tree *T* is noted $\left|\deg_T^+(x)\right|$. The number of downstream optical channels of *x* on the tree *T* is noted $\left|\deg_T^-(x)\right|$. A branching node *x* of a light-tree *T* is such that its number of downstream optical channels is superior or equal to 2 ($\left|\deg_T^-(x)\right| \geq 2$). Any node connected to a multicast tree has a single upstream optical channel (except the source, which has none) and any node may have multiple downstream optical channels (except the destinations, which have none).

Two optical channels of a light-forest are adjacent if one optical channel is an upstream channel of a node on a tree *T* and the other is a downstream channel of the same node on the same tree.

3. Tree Reconfiguration operations

Four reconfiguration operations are applicable on an optical switch. We denote C_j as the configuration (set of lightpaths) obtained at the j^{th} step of the tree reconfiguration process.

1) Addition of a light switching: *Add*

$Add(C_j) \Rightarrow \exists x \in C_j / (+x_i^I, x, l, x_j^O)$. If x_j^I is the sink port of an optical channel *u*, $Add(C_j)$ adds one optical channel *v* with wavelength *l* in the configuration C_j such that:

- x_j^O is the source port of *v*
- *v* is adjacent to *u*

- The signal coming from x_i^I feeds *v*

2) Deletion of a light switching: *Del*

$Del(C_j) \Rightarrow \exists x \in C_j / (-x_i^I, x, l, x_j^O)$. If x_j^O is the source port of an optical channel *u*, $Del(C_j)$ deletes the link *u* from the configuration C_j.

3) Combined operation: *Comb*

$Comb(C_j) \Rightarrow \exists x \in C_j / (x_i^I, x, l, x_j^O \Rightarrow x_k^O)$. If x_j^I is the sink port of an optical channel *u*, and x_j^O the source port of an optical channel v_1, $Comb(C_j)$ simultaneously deletes the link v_1 from the configuration C_j and adds an optical channel v_2 with wavelength *l* such that x_k^O is the source port of v_2, and is adjacent to *u*.

4) Wavelength conversion: *Conv*

$Conv(C_j) \Rightarrow \exists x \in C_j / (x_i^I, x, l \Rightarrow m, x_j^O)$. If x_j^O is the source port of an optical channel *u*, $Conv(C_j)$ converts the wavelength *l* of *u* to wavelength *m*.

4. Related Works

In the literature, reconfiguration policy and path computation have been studied in [8]-[11]. They do not consider the configuration process in itself. The well-known reconfiguration processes *MBB* (Make-Before-Break) and *BBM* (Break-Before-Make) are proposed for path reconfiguration. *MBB* achieves path reconfiguration without path interruption but not regardless of the paths and the cost of many reconfiguration steps. [12] studied fast path reconfiguration without data flow interruption as they are interested in unicast connections.
The connections considered in all these previous works are unicast connections. In [13], the frequency of light-tree reconfiguration is studied. It do not consider the configuration process in itself. In this paper, since the problem considered is tree reconfiguration without data flow interruption, we first describe an *MBB* adaptation to the tree reconfiguration problem. This *MBB-based* algorithm is called *MBB_2*. Then we present the *BpBAR_2* process proposed in our previous work. It was the first time that a tree reconfiguration algorithm which achieves uninterrupted flow was proposed in the literature. We compare our new algorithm to these two algorithms.

4.1 MBB_2

MBB is a well-known path reconfiguration process. The *MBB* reconfiguration process causes an interruption when the initial and final paths make a cyclic dependency. In the adaptation of *MBB* to tree reconfiguration which uses no additional resource, the no interruption requirement is not fulfilled. In tree reconfiguration without additional resources, since the same link may be shared by several branches of the tree, the probability of having cyclic dependency between the branches of the trees increases. Thus, *MBB_2* is an *MBB-based* algorithm which uses one additional wavelength during the reconfiguration process. *MBB_2* consists of two phases. In each phase, it uses the function *TRwMBB2* described in Algorithm 2. In the first phase of *MBB_2*, *TRwMBB2* configures each sub-tree of the new tree with the additional wavelength. In the second phase, it configures each sub-tree of the new tree with the final wavelength. The sub-trees of the initial tree (respectively the final tree) considered in this algorithm are rooted at the son nodes of the tree source *s* in the initial tree (respectively in the final tree). The set of destinations of a sub-tree on the new tree is denoted *Dest*.

Algorithm 1: *MBB_2*

```
MBB2 (T0, Tz)
1. {
2. TRwMBB2 (T0, T)
3. TRwMBB2 (T, Tz)
4. }
```

Algorithm 2: Function used by *MBB_2*

```
TRwMBB2 (T0, Tz)
1. {
2. Configure in one step all the nodes on the new tree Tz to
   build all the pre-established branches with the additional
   wavelength
3. For each sub-tree of the new tree Tz
4.  {
5.  Select a sub-tree of the old tree not configured
6.  Interrupt in one step the data flow on branches (on the
      initial tree T0) toward all the destinations dj not
      configured and such that dj ∉ Dest
7.  Switch wavelength from the sub-tree of the old
    tree to the sub-tree of the new tree at the tree source s to
    feed the new sub-tree
8.  Configure in parallel all the ports of the nodes on the old
      sub-tree to remove its branches
9.  }
10. }
```

4.2 BpBAR_2

BpBAR_2 is a *BpBAR-based* algorithm which consists of two phases. Each phase consists of several stages. Each stage is a series of reconfiguration steps to reconfigure one branch. Figure 3 represents a diagram of operations scheduling in a stage of the *BpBAR* process to ensure the continuity of the lightpath during the reconfiguration. This diagram can be subdivided into five parts according to the

successive operations executed as shown Table 1. *BpBAR_2*, which is a *BpBAR-based* process, uses parallel execution of operations to reduce the reconfiguration delay. *BpBAR_2* executes, in parallel, all the operations belonging to the same part of a *BpBAR* stage as shown Table 2. Figure 4 represents the diagram of a reconfiguration stage with *BpBAR_2*.

In the first phase of a reconfiguration with *BpBAR_2*, each branch is established per stage; a new tree, a copy of the final tree which uses wavelengths different from the initial, is obtained. In the second phase, each branch is established per stage; a tree, a copy of the final tree which uses the initial wavelength, is obtained. We describe below a reconfiguration stage with the *BpBAR* process.

Table 1 describes the different parts of a reconfiguration stage with the *BpBAR* process. The operations are executed sequentially in this diagram. The length of each part (number of steps) is equal to the number of ports to be configured in this part.

After the setup of the pre-established path in part 1 of a stage, the combined operation is executed in part 2 to feed the pre-established path. *Comb* is executed at the end node of the pre-established path. Since it feeds the pre-established path and interrupts data flow on the other path, the appropriate switching node is the one which maintains the data flow continuity toward all the destinations after its configuration. This appropriate switching node depends on the initial tree *T* and the destination of the pre-established path *d*. It is denoted $Sw\_node(T,d)$. For example, in Fig. 2, C is the node $Sw\_node(T,d_1)$. Then, in part 3, the wavelength switching is deleted on all nodes of the exclusive part of the old branch on which flow is interrupted. Parts 4 and 5 occur when the switching node is different to the source node *s*.

Definition 1: The end part of path *p* beginning at path node *x* is denoted $term(p,x)$

Definition 2: $Sw\_node(T,d)$ is the root node of the sub-tree of the tree *T* which will be used during the branch configuration between the source node and *d*. A node *x* is the node $Sw\_node(T,d)$ of the branch $Br(T,s,d)$ if:

(1) *x* is a branching node of tree *T* which uses the initial wavelength

(2) For each node $y \in term(Br(T,s,d),x)$ and $y \notin \{x,d\}$, $\left|\deg_T^-(x)\right| = 1$.

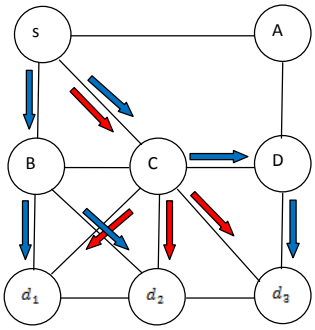

Fig. 2. Example of tree reconfiguration problem. The old tree T_0 is in red and the new tree T_z is in blue.

Table 1: Distinct parts of a stage in *BpBAR* process

Parts	*Description*
1	From C_i to C_{i+k_1}: This part consists of setting up the pre-established path between the destination d and $Sw\_node(T_j,d)$. The operation used is *Add*: $C_{j+1} = Add(C_j)$. The length is equal to the number of ports to be configured.
2	From C_{i+k_1} to C_{i+k_1+1}: This part feeds the pre-established path by executing *Comb* operation at $Sw\_node(T_j,d)$: $C_{j+1} = Comb(C_j)$. It consists of one step.
3	From C_{i+k_1+1} to C_{i+k_2}: This part deletes the pre-established path between the destination d and $Sw\_node(T_j,d)$. The operation used is *Del*: $C_{j+1} = Del(C_j)$. The length is equal to the number of ports to be configured.
4	From C_{i+k_2} to C_{i+k_2+1}: This part consists of executing *Comb* operation at the source *s* to interrupt the flow between $Sw\_node(T_j,d)$ and *s*. It consists of one step.
5	From C_{i+k_2+1} to C_{i+k_3}: $C_{j+1} = Del(C_j)$. This part deletes the pre-established path between the node $Sw\_node(T_j,d)$ and the source node *s*. The length is equal to the number of ports to be configured.

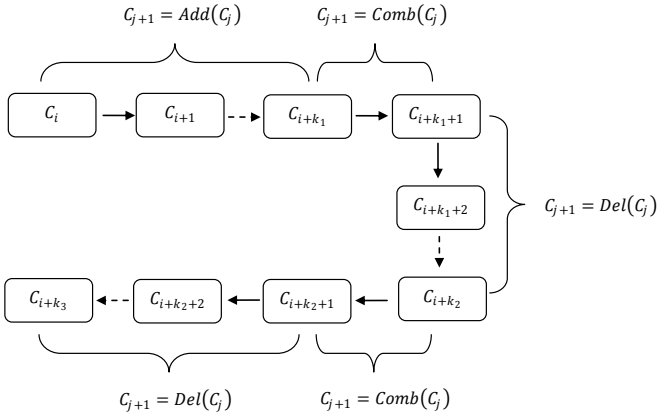

Fig. 3. Diagram of operations scheduling during a branch reconfiguration in a stage with *BpBAR*

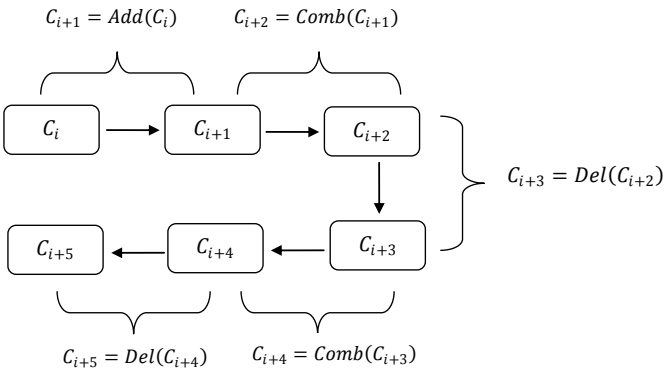

Fig. 4. Diagram of operations scheduling during a branch reconfiguration in a stage with *BpBAR_2*

Table 2: Operations executed in parallel in a stage of *BpBAR* process

Parts of stage	Series of configurations by part in Fig.3	Configurations produced after parallel execution
1	$< Add(C_i)$, $Add(C_{i+1})$,..., $Add(C_{i+k_1-1}) >$	$< Add(C_i) >$
2	$< Comb(C_{i+k_1}) >$	$< Comb(C_{i+1}) >$
3	$< Del(C_{i+k_1+1}), Del(C_{i+k_1+2}),..., Del(C_{i+k_2-1}) >$	$< Del(C_{i+2}) >$
4	$< Comb(C_{i+k_2}) >$	$< Comb(C_{i+3}) >$
5	$< Del(C_{i+k_2+1}), Del(C_{i+k_2+2}),... , Del(C_{i+k_3-1}) >$	$< Del(C_{i+4}) >$

We prove in the following section that *BpBAR_2* maintains lightpath continuity during tree reconfiguration.

5. Proof of Data Flow Continuity Toward All Destinations During Tree Reconfiguration with *BpBAR_2*

BpBAR_2 is a *BpBAR-based* algorithm which uses parallel execution of operations. First, we prove that a reconfiguration stage with *BpBAR* maintains lightpath continuity. Second, we prove that the parallel execution of all the operations belonging to the same part of a stage with *BpBAR* maintaining continuity of the lightpath.

Let us assume that a stage begins with a configuration C_i and is noted $< O(C_i), O(C_{i+1}),..., O(C_{i+k_3}) >$ where C_i is the i^{th} configuration which contains a light-forest spanning all the destinations of a multicast connection <s, {d}> and *O* is a tree reconfiguration operation (described in Section 3). C_i is represented by $C_i = LF_i$ where LF_i is a light-forest spanning all the destinations. We will prove that each part of a reconfiguration stage with a *BpBAR* process maintains the continuity of the lightpath.

(1) Part 1

Let us consider a configuration C_j of part 1 which contains a light-forest spanning all the destinations of a multicast connection, $C_j = LF_j + p_j$ where p_j is a pre-established path. Let V_j denote the set of optical channels of C_j such that LF_j spans all the destinations of the multicast group. $Add(C_j)$ implies add new optical channel *u* in C_j such that *u* is adjacent to the end link of p_j. Thus, the new set of optical channels after $Add(C_j)$ is $V_{j+1} = V_j \cup \{u\}$. The optical channels of LF_j are not modified so LF_j spans all the destinations. Thus, C_{j+1}=$Add(C_j)$ contains a light-forest spanning all the destinations.

In *BpBAR*, the *Add* operation ($(+x_i^I, x, l, x_j^O)$), is carried out such that the ports x_i^I and x_j^O of the switch *x* do not belong to the set of ports of LF_j. Thus, the new pre-established path p_{j+1} is not fed and there is no loop.

(2) Part 2

Comb is executed on the switching node $Sw\_node(T_j, d)$. The conditions to execute *Comb* in a configuration C_j with *BpBAR*, in part 2 are:

- $C_j = LF_j + p_j$ where the end nodes of p_j are $Sw\_node(T_j, d)$ and *d*.
- $Comb(C_j) \Rightarrow (Sw_i^I, Sw, l, Sw_j^O \Rightarrow Sw_k^O)$, where *Sw* is the node $Sw\_node(T_j, d)$. Sw_i^I and Sw_j^O belong to LF_j, Sw_k^O does not belong to LF_j but belongs to p_j.

$Comb(C_j)$ interrupts the wavelength coming from Sw_i^I to Sw_j^O and switches the wavelength coming from Sw_i^I to Sw_k^O. Since Sw_k^O belongs to p_j, the pre-established path is now fed and ensures the continuity of the lightpath toward the destination *d*. Since Sw_j^O belongs to $term(Br(T_j, s, d), Sw)$, $term(Br(T_j, s, d), Sw)$ is interrupted. We know by definition that each node of $term(Br(T_j, s, d), Sw)$ has only one downstream optical channel on T_j and feeds only destination d_j before its interruption. Since the pre-established path ensures the continuity of the lightpath toward *d* after $Comb(C_j)$, a light-forest spanning all the destinations of a multicast connection exists. If we assume that $term(Br(T_z, s, d), Sw)$

is the new pre-established path p_{j+1}, $C_{j+1} = Comb(C_j) = LF_{j+1} + p_{j+1}$.

(3) Part 3

The conditions to execute *Del* in a configuration C_j with *BpBAR*, in part 3 are:

- $C_j = LF_j + p_j$
- $Del(C_j) \Rightarrow (-x_i^I, x, l, x_j^O)$, where x is the end node of p_j, x_j^O does not belong to LF_j but belongs to p_j.

We know by definition that the pre-established path p_j is not used to transmit data (x_j^O does not belong to LF_j but belongs to p_j). $Del(C_j)$ configures the port x_j^O on p_j to delete the wavelength switching. After $Del(C_j)$, the ports of LF_j are not modifying and a light-forest spanning all the destinations of the multicast connection exists. If p_{j+1} is the new pre-established path after the *Del* operation, $C_{j+1} = Del(C_j) = LF_{j+1} + p_{j+1}$.

(4) Part 4

Comb is executed on the source node of the multicast connection *s* in this part. The conditions to execute *Comb* in a configuration C_j with *BpBAR*, in this part are:

- $C_j = LF_j$
- The lightpath between the source node *s* and a destination *d* contains *s* twice: for example (*s, x, …,Sw,.., s, y,…,d*).
- $Comb(C_j) \Rightarrow (-, s, l, s_k^O \Rightarrow s_k^O)$ where s_k^O belongs to LF_j.

This operation is equivalent to $(-, -, s, l, s_k^O)$ followed by $(+, -, s, l, s_k^O)$. The signal coming from any input port of *s* is interrupted. The lightpath between *s* and the destination *d* becomes (**s**, *y,…,d*). The part of path (*Sw,..*, **s**) which is not used to transmit data after the operation becomes the pre-established path p_{j+1}. Then $C_{j+1} = Comb(C_j) = LF_{j+1} + p_{j+1}$ contains a light-forest LF_{j+1} spanning all the destinations of the multicast connection.

(5) Part 5

The conditions to execute *Del* in a configuration C_j with *BpBAR* in this part are:

- $C_j = LF_j + p_j$
- $Del(C_j) \Rightarrow (-x_i^I, x, l, x_j^O)$, where x is the end node of p_j, x_j^O does not belong to LF_j but belongs to p_j.

p_j is not used to transmit data (x_j^O does not belong to LF_j but belongs to p_j). $Del(C_j)$ configures the port x_j^O on p_j to delete the wavelength switching. After $Del(C_j)$, the ports of LF_j are not modifying and a light-forest spanning all the destinations of the multicast connection exists. If p_{j+1} is the new pre-established path after the *Del* operation, $C_{j+1} = Del(C_j) = LF_{j+1} + p_{j+1}$.

We have proved that all the parts of a reconfiguration stage with *BpBAR* maintain the lightpath continuity.

BpBAR_2 is a *BpBAR-based* algorithm which executes operations in parallel. Table 2 presents the operations of BpBAR done in parallel. The path pre-established in part one is not used for transmission during the duration of this part. All the *Add* operations in part one can be execute in parallel, in one step to pre-establish the path. In part three and five, the old path which data flow is interrupted is not used for transmission during the duration of this part. All the *Del* operations in part three can be executed in parallel, in one step to delete this part of old path and All the *Del* operations in part five can be executed in parallel, in one step to delete this part of old path.

6. Reconfiguration with Wavelengths Limit Constraint

6.1 Context and Problem Formulation

We consider a WDM optical network which has a limited number of available wavelengths per link. A multicast connection established in this network on the tree T_0 uses wavelength l_0. We want to move from the tree T_0 to a new tree T_z. The initial tree T_0 and the final tree T_z must use the same wavelength l_0. We assume that the new tree is known. The reconfiguration must be carried out without data flow interruption. The tree configuration process (*BpBAR_2*) proposed in our previous work produces a series of light-forests since it must maintain lightpath continuity. The number of trees per light-forest is not increased with the proposed algorithm. This number is proportional to the number of branches to be established. In a network with a multicast connection established, several branches can share the same link which has very

limited available resources. In this context, *BpBAR_2* may be inapplicable. Some branches which share the same link cannot be established. The problem in this work is to reduce the number of trees per light-forest at each step of the reconfiguration, regardless of the initial and final trees, and the network topology.

6.2 Our Proposition: *TRwRC*

The proposed tree reconfiguration is the *TRwRC* (tree reconfiguration with resources constraint) algorithm. It has two phases. *TRwRC* produces a series of light-forests during the tree configuration process. Let us assume that the destinations of the multicast connection is subdivided into two sets during the reconfiguration process: a set of destinations whose lightpaths are reconfigured (uses paths on the new tree) denoted *Dest_Config* and a set of destinations whose lightpaths are not reconfigured (uses paths on the initial tree) denoted *Dest_NConfig*. The maximal number of trees per light-forest in a configuration produced with *TRwRC* is two. Each tree, at each step, spans the destinations of one set described above.

Algorithm 3 represents *TRwRC*. *TRwRC* uses the function *ReconfBr* described in Algorithm 4 to reconfigure each branch of the initial tree T_0. The reconfiguration of each branch with *ReconfBr* consists of two stages. In the first stage, the branch is established with an additional wavelength l_2. In the second stage, the branch is established with the additional wavelength l_1. The tree obtained after the reconfiguration of all the branches of the initial tree is a copy of the new tree T_z using the additional wavelength l_1 denoted *T'*. In the second phase, *TRwRC* configures simultaneously the ports of the nodes of the tree *T'* obtained after the first phase to convert the additional wavelength l_1 into the initial wavelength. We described below the function *ReconfBr* used to reconfigure a branch.

Let C_0 be the initial configuration and $Br(T_z, s, d)$ a branch we want to establish. C_0 contains a light-tree spanning all the destinations. We note $C_0 = T_0$. *TRwRC* uses the *Add* operation to setup in one step a pre-established path between $Sw\_node(T_0, d)$ and d with an additional wavelength (for example l_2). After this operation, $C_1 = Add(C_0) = T_1 + p_1$ where $T_1 = T_0$ and p_1 is the pre-established path. The combined operation is executed on $Sw\_node(T_0, d)$ to simultaneously feed the pre-established path and interrupt the flow on the old path. The pre-established path now ensures the continuity of the lightpath toward the destination d. Before execute *Comb* operation, $term(Br(T_0, s, d), Sw\_node(T_0, d))$ is shared only by the lightpath toward d. After the switching, $C_2 = Comb(C_1) = T_2 + T_2' + p_2$ where T_2 spans the destinations of *Dest_NConfig*, T_2' spans the destinations of *Dest_Config* ({d}) and, p_2 is a part of the old branch $term(Br(T_0, s, d), Sw\_node(T_0, d))$. p_2 is deleted in one step and $C_3 = Del(C_2) = T_3 + T_3'$ where $T_3 = T_2$ and $T_3' = T_2'$. If the node $Sw\_node(T_0, d)$ is not the source node *s*, *s* is reconfigured to interrupt the data flow between $Sw\_node(T_0, d)$ and *s*, then this part of the branch is deleted. After this stage, the new branch is established with the additional wavelength l_2. The same process is repeated in a new stage to establish the branch with the additional wavelength l_1. The two stages are repeated for each branch. In the first phase of the proposed algorithm, we obtain a tree *T'*, a copy of the new tree T_z, using the additional wavelength l_1. In the second phase, the ports of the nodes of the tree *T'*, obtained after the first phase, are configured simultaneously to convert the additional wavelength l_1 into the initial wavelength.

Algorithm 3: *TRwRC* algorithm

```
TRwRC (T0, Tz)
1. {
2. For each destination d
3.   {
4.   ReconfBr (T0, Tz, d, T)
5.   ReconfBr (T, Tz, d, T')
6.   }
7. Configure simultaneously the ports of the nodes of the
     tree T' to convert the wavelength used into
     the initial wavelength.
8. }
```

Algorithm 4: Function used to established a new branch

```
ReconfBr (T0, Tz , d ,T')
1. {
2. If ( Br(Tz, s, d) is different from Br(T0, s, d) )
3.   {
4.   Determine node Sw_node(T0, d) and configure
     in parallel the ports of the nodes between each
     destination node d and the node Sw_node(T0, d)
5.   Configure the node Sw_node(T0, d) to feed the
     new branche and interrupt the flow of the old branches
6.   Remove, in parallel, the part of the old branche in
     which flow is interrupted
7.   If T0 and Tz have different topologies
8.        {
9.        Configure s to interrupt the part between s
          and Sw_node(T0, d) which is different to s
10.       Configure, simultaneously, the ports of
          the nodes between s and Sw_node(T0, d)
          to remove this part
11.       }
12. }
13. }
```

7. Performance Evaluation

7.1 Criteria and Method

We use NSF networks (see Fig. 5) and COST 239 (see Fig. 6) to evaluate our algorithm. Well-known topologies are used in many similar papers as performance evaluation topologies. The initial and final trees are selected arbitrarily (they are called pairs of trees). One is the minimum spanning tree (MST) computed with Prim's heuristic [14]. This algorithm calculates a good approximation of the Steiner tree for a multicast group. The other is the shortest paths tree (SPT) which uses Dijkstra's algorithm [15] for the multicast group.

We compare our algorithm *TRwRC* to *BpBAR_2* and *MBB_2*. The criteria are: the duration of the process, the additional resources used and the number of interruptions during the tree configuration process. In addition to the above criteria, also used in our previous work, we define a new criterion which is the number of unsolved tree configuration problems. An unsolved problem is a pair of trees which a certain algorithm cannot reconfigure without data flow interruption. The number of unsolved problems depends on the number of available wavelengths per link. If the number of pairs of trees generated is *nb_tirage*:

$$nb\_unsolv\_pb(w) = \sum_{k=1}^{nb\_tirage} \alpha_k^w(T_0, T_z) \quad (1)$$

Where $\alpha_k^w(T_0, T_z) = 1$ if for the k^{th} pair of trees, *w* available wavelengths per link is not enough to reconfigure all the tree branches without flow interruption and $\alpha_k^w(T_0, T_z) = 0$ otherwise. In the evaluation, *nb_tirage* is chosen experimentally and is equal to 5000.

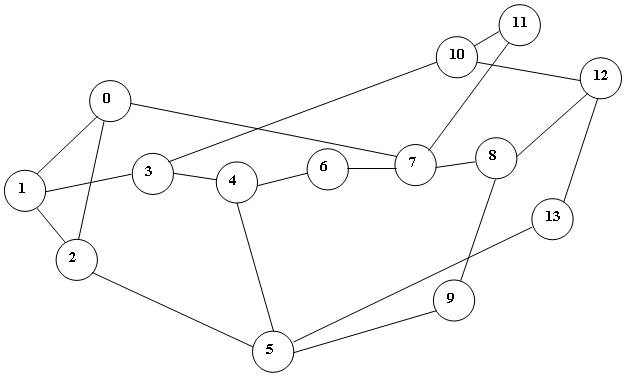

Fig.5. Topology of NSF Network

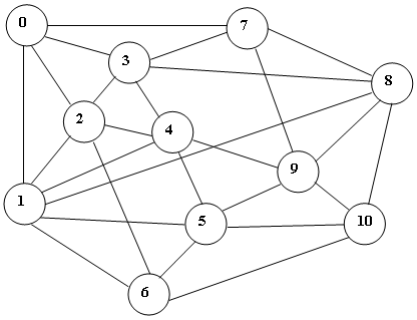

Fig. 6. Topology of COST 239 network

7.2 Results and Analysis

The results of the different evaluations are noted in Tables 3 to 7. Tables 3 to 5 show the performance results when the initial tree is SPT and the final tree MST. Tables 6 and 7 show the results obtained when we swap the algorithms used to compute the trees. The initial tree is now MST and the final tree SPT.

Table 3: Average and standard deviation of the number of steps for each reconfiguration process

	COST 239			*NSF NETWORK*		
CONF. PROCESS	*AVG (STEP)*	*SD (STEP)*	*MIN/MAX (STEP)*	*AVG (STEP)*	*SD (STEP)*	*MIN/MAX (STEP)*
BpBAR2	*18.07*	*12.09*	*6/51*	*22.62*	*17.35*	*6/74*
TRwRC	*18.31*	*11.61*	*7/54*	*22.68*	*16.15*	*7/79*
MBB2	*8.32*	*2.42*	*6/17*	*8.58*	*2.04*	*6/17*

Table 4:Average and standard devisation of additional cost for each reconfiguration process

	COST 239			*NSF NETWORK*		
CONF. PROCESS	*AVG (WL)*	*SD (WL)*	*MIN/MAX (WL)*	*AVG (WL)*	*SD (WL)*	*MIN/MAX (WL)*
BpBAR2	*166.84*	*195.57*	*10/861*	*370.1*	*448.1*	*9/2273*
TRwRC	*107.52*	*96.45*	*13/430*	*227.8*	*220.4*	*13/1104*
MBB2	*85.95*	*30.13*	*18/177*	*122.7*	*40.34*	*18/229*

Table 5: Average and standard deviation of interruption duration for each reconfiguration process

	COST 239			*NSF NETWORK*		
CONF. PROCESS	*AVG (STEP)*	*SD (STEP)*	*MIN/MAX (STEP)*	*AVG (STEP)*	*SD (STEP)*	*MIN/MAX (STEP)*
BpBAR2	*0*	*0*	*0/0*	*0*	*0*	*0/0*
TRwRC	*0*	*0*	*0/0*	*0*	*0*	*0/0*
MBB2	*2.85*	*1.47*	*0/6*	*3.42*	*1.86*	*0/6*

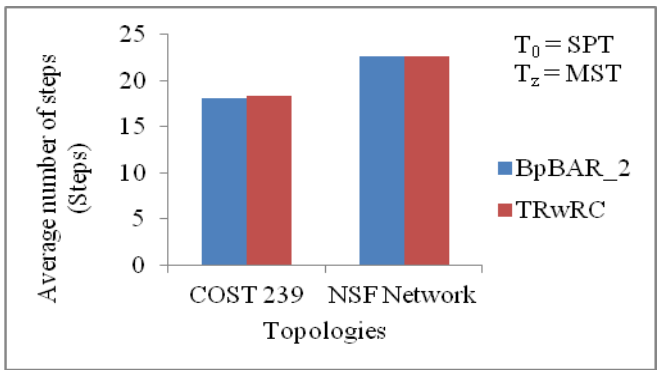

Fig.7. Setup duration of each algorithm in COST 239 and NSF Network

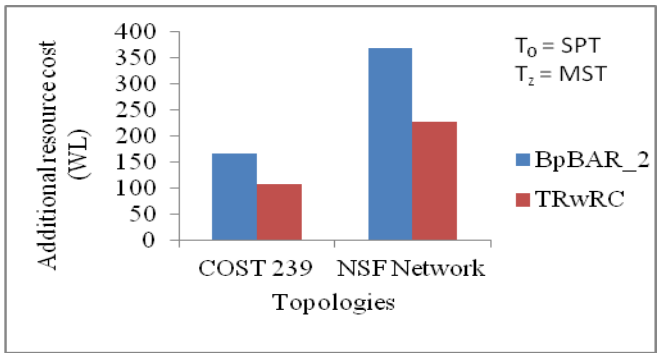

Fig.8. Additional resource cost of each algorithm in COST 239 and NSF Network

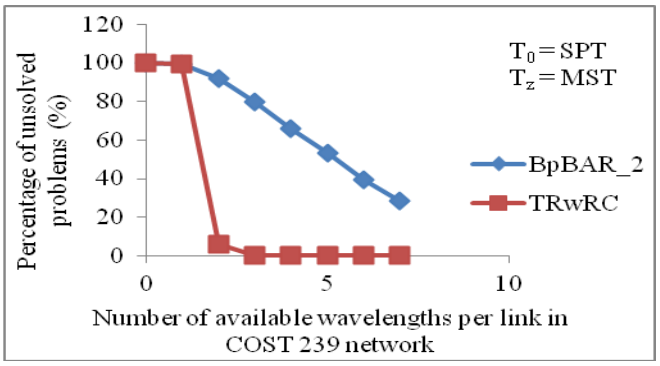

Fig.9: Percentage of unsolved problems in COST 239

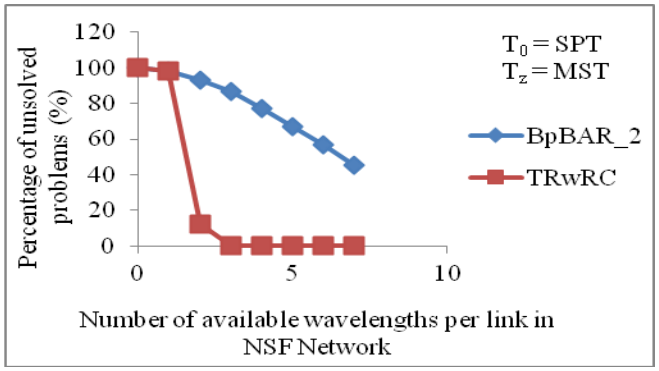

Fig.10. Percentage of unsolved problems in NSF Network

The results of the performance evaluations noted in Table 3 show that the number of steps produced by *TRwRC* is approximately equal to the number of steps produced by *BpBAR_2 (*Fig. 7), but is greater than those produced by *MBB_2*. This is due to the fact that *TRwRC* is a back and forth tree reconfiguration process. Table 4 shows that tree reconfiguration with *TRwRC* uses more resources than reconfiguration with the *MBB_2* process. This is due to the fact that *MBB_2* uses the same initial wavelength during all the process. So, it requires no additional wavelength while *TRwRC* uses additional resources to establish new branches. Table 5 shows that *MBB_2* causes data flow interruption while *TRwRC* maintains continuity of data flow. The continuity of data flow is due to the properties of the *BpBAR* process used by the *TRwRC* algorithm. A good tree reconfiguration algorithm is one which does not interrupt the flow toward any destination during the reconfiguration. Thus, the *TRwRC* algorithm is much better than the *MBB_2* algorithm.

BpBAR_2 and *TRwRC* maintain the continuity of data flow and produce approximately the same number of steps (see Table 5 and Fig. 7). The important improvement is shown in Figs. 8, 9, and 10. Figs. 9 and 10 show the evolution of the percentage of unsolved problems with the *BpBAR_2* and *TRwRC* algorithms as a function of the number of available wavelengths per link. They show that when the number of available wavelengths per link is under or equal to two, generally the *BpBAR-based* algorithms produce tree reconfiguration with data flow interruption. But the performance produced by *TRwRC* is better than the performance produced by *BpBAR*_2. These figures show also that when the number of available wavelengths per link is equal to three, *TRwRC* allows reconfiguration of any tree without data flow interruption regardless of the topology, while *BpBAR_2* requires more available resources. Moreover, Fig. 8 shows that the cost of additional resources produced by *TRwRC* is less than the cost of additional resources produced by *BpBAR_2*. So *TRwRC* does not use more additional resources than *BpBAR_2* during its process. This is due in the fact that each configuration produced by *TRwRC* contains at most two light-trees and one pre-established path, while with *BpBAR_2* the number of trees increases with the number of branches.

When we swap the algorithms used to compute the intial and final trees, Figs. 11 to 13 show that the same performance gains are kept: *TRwRC* reconfigures any pair of trees when the number of available wavelengths per link is equal to or higher than three and does not use more additional resources compared to *BpBAR_2*.

Table 6: Average and standard deviation of reconfiguration duration for each reconfiguration process

	COST 239			NSF NETWORK		
CONF. PROCESS	AVG (STEP)	SD (STEP)	MIN/MAX (STEP)	AVG (STEP)	SD (STEP)	MIN/MAX (STEP)
BpBAR2	19.02	12.81	6/61	22.59	16.43	6/80
TRwRC	20.33	12.80	7/62	22.21	12.92	7/62
MBB2	10.90	3.20	6/17	8.77	2.4	6/17

Table 7: Average and standard deviation of additional cost for each reconfiguration process

	COST 239			NSF NETWORK		
CONF. PROCESS	AVG (STEP)	SD (STEP)	MIN/MAX (STEP)	AVG (STEP)	SD (STEP)	MIN/MAX (STEP)
BpBAR2	92.75	100.65	5/542	183.8	219.6	5/1088
TRwRC	93.15	89.67	8/427	116.4	127.7	9/701
MBB2	62.98	32.24	9/149	10.51	40,01	18/229

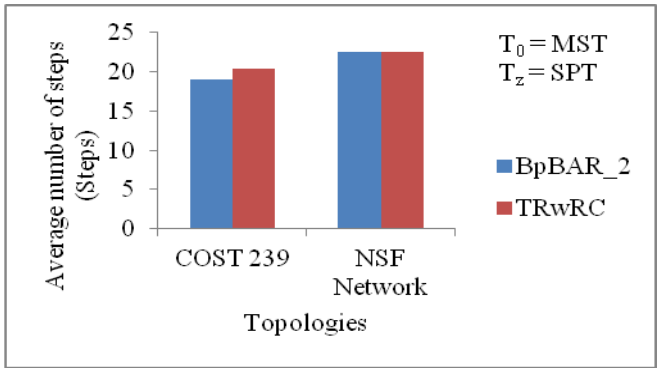

Fig.11. Setup duration of each algorithm in COST 239 and NSF Network

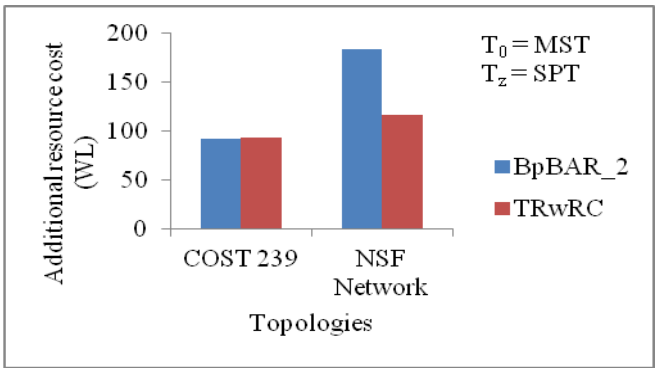

Fig.12. Additional resource cost of each algorithm in COST 239 and NSF Network

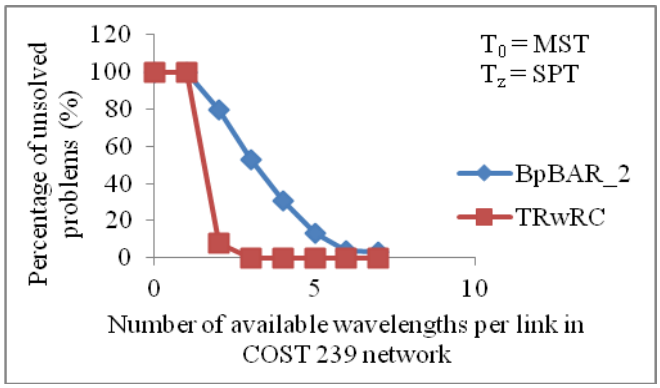

Fig.13. Percentage of unsolved problems in COST 239

8. Conclusion

Tree reconfiguration in WDM optical networks is studied in this paper. In the literature, the well-known path reconfiguration process *MBB* is proved to induce flow interruption. Similarly, adaptation of *MBB* to tree reconfiguration with no additional wavelengths may induce interruption. Even more, the tree branches may increase the probability of cycle outbreak and thus interruption. Adaptation of *MBB* to tree reconfiguration with an additional wavelength is also inefficient because it produces flow interruption. In our previous work, we proposed a branch-by-branch algorithm (*BpBAR_2)* which aims to reconfigure trees without data flow interruption, and reduces the reconfiguration setup duration and the network resources used during the reconfiguration process. The availability of network resources was not considered in this previous work. In this paper, we consider tree reconfiguration problems in a network wherein the number of available wavelengths is limited. In this context, the algorithm *BpBAR_2* becomes inefficient. In this paper, we propose *TRwRC* which allows reconfiguration of trees when each optical link has a limited number of available wavelengths. *TRwRC* maintains the continuity of the flow since it uses the appropriate switching node $Sw\_node(T_0, d)$ and the combined switching operation to feed the new branches. The reconfiguration setup duration produced is also good compared to the previous algorithm. *TRwRC* uses, at most, two trees per light-forest to span all the destinations during the different configurations produced. Thus, it allows tree reconfiguration in all-optical networks with limited resources. It also reduces the cost of used resources during the reconfiguration process. When the number of available wavelengths per link is three (at least), any tree reconfiguration without data flow interruption is possible with *TRwRC.*

References

[1] B. Cousin, J. C. Adépo, S. Oumtanaga, and M. Babri, "Tree Reconfiguration Without Lightpath Interruption in WDM Optical Networks", International Journal of Internet Protocol Technology, Vol. 7 No. 2, 2012, pp. 85–95.
[2] B. Mukherjee, "WDM Optical Communication Network: Progress and Challenges", IEEE Journal on Selected Areas in Communication, Vol. 18 No. 10, 2000, pp. 1810–1824.
[3] A. Ding and G-S. Poo, "A survey of optical multicast over WDM networks", Computer Communications, Vol. 26 No. 2, 2003, pp. 193–200.
[4] R. Malli, X. Zhang, and C. Qiao, "Benefit of Multicasting in All-Optical Networks", Photonics East (ISAM, VVDC, IEMB) in International Society for Optics and Photonics, 1998.
[5] F. Zhou, M. Molnar, and B. Cousin, "Is Light-Tree Structure Optimal for Multicast Routing in Sparse Splitting WDM Networks?", in 18th IEEE International Conference on Computer Communications and Networks (ICCCN 2009).
[6] S. Jawhar, B. Cousin, and S. Lahoud "Effect of Splitting Factor on the Generated Multicast Trees in Optical Networks", in International Conference on Network Communication and Computers (ICNCC 2011).
[7] F. Zhou, M. Molnar, B. Cousin, and C. Qia, "Cost Bounds and Approximation Ratios of Multicast Light-trees in WDM Networks", Journal of Optical Communications and Networking, Vol. 3 No. 4, 2011, pp. 323–334.
[8] I. Baldine and G. N. Rouskas, "Dynamic Reconfiguration Policies for WDM Networks", in 18th Annual Joint Conference of the IEEE Computer and Communications Societies 1999 (INFOCOM), Vol. 1, pp. 313–320.
[9] I. Baldine and G. N. Rouskas, "Traffic Adaptive WDM Networks: A Study of Reconfiguration Issues", *IEEE/OSA* Journal of Light Wave Technology, Vol. 19 No. 4, 2001, pp. 433–455.
[10] T. Kárász, "Consolidation Strategies of Provisioning Oriented Optical Networks", Journal of Optical Networking, Vol. 5 No. 6, 2006, pp. 445–462.
[11] T. Kárász and Z. Pándi, "Optimal Reconfiguration of Provisioning Oriented Optical Networks", in 3rd International Working Conference on Performance Modeling and Evaluation of Heterogeneous Networks (HET-NETs), 2005.
[12] B. Cousin and M. Molnar, "Fast Reconfiguration of Dynamic Networks", International Workshop on Dynamic Networks, 2007.
[13] M. Perényi, P. Soproni, T. Cinkler, and D. Larrabeiti, "Regular Reconfiguration of Light-Trees in Multilayer Optical Networks", in 12th International Conference on Optical Networking Design and Modeling (ONDM), 2008, pp. 1–6.
[14] R. C. Prim, "Shortest Connection Networks and Some Generalizations", Bell System Technical Journal, Vol. 36, 1957, pp.1389–1401.
[15] E.W. Dijkstra, "A Note on Two Problems in Connection with Graphs", NumerischeMathematik, Vol. 1 No. 1, 1959, pp. 269–271.

Bernard Cousin is Professor at the University of Rennes 1, France. He received his PhD in Computer Science in 1987 from the University of Paris 6, and the Habilitation à Diriger les Recherches in 1992 from the University of Bordeaux 1. He is currently a member of IRISA (a CNRS-University Joint Research Laboratory located at Rennes). He is the head of a research group on networking. He has co-authored more than 100 papers published in international journals and conferences. His research interests include high speed networking and distributed multimedia applications.

Joël Christian Adépo was born in Akoupé (Côte d'Ivoire). He received his MS in Computer Science from the University of Nangui Abrogoua (Côte d'Ivoire) in 2011. Currently a PhD student in Computer Science at University of Nangui Abrogoua, Abidjan, Côte d'Ivoire, 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 member of the Laboratoire de Recherche en Informatique et Télécoms (LARIT) since 2006.

Souleymane Oumtanaga received his PhD in Computer Science from University Paul Sabatier of Toulouse, France in 1995. From 1999 to 2000, he was at the Laboratoire de Recherche en Informatique et Mathématiques Appliquées (LARIMA) at INPHB, Côte d'Ivoire. Since 1990, he has also been the Head of the Network Information Center (NIC) of the Côte d'Ivoire. He has been a Professor in Computer Science since 2007 and he currently manages the Laboratoire de Recherche en Informatique et Télécoms (LARIT) at INPHB. His research interests include IP mobility, IP network security, IPv6, wireless networks, and mobile networks.