A Distributed Bandwidth Sharing Heuristic for Backup LSP Computation

Mohand Yazid SAIDI, Bernard COUSIN
Université de Rennes I – Campus de Beaulieu
IRISA, 35042 Rennes Cedex, France
{msaidi,bcousin}@irisa.fr

Jean-Louis LE ROUX
France Télécom
2, Avenue Pierre Marzin, 22300 Lannion, France
jeanlouis.leroux@orange-ftgroup.com

Abstract— **With the advent of MPLS, the restoration times of communications is decreased down to 50 ms by the use of preconfigured backup LSPs. To ensure there are enough resources after a failure, the backup LSPs must reserve the resources they need beforehand. However and contrarily to the primary LSPs which really use their resources, the backup LSPs do not use them until a failure of the protected component occurs. Hence, to optimize and maximize resource availability in the network, backup LSPs may share their resource reservation. Indeed, under the hypothesis of single failures in the network, some backup paths are not active at the same time since they protect against the failure of different components.**

In this article, we propose an efficient Distributed Bandwidth Sharing (DBS) heuristic capable to protect the primary LSPs against all types of failure risks (link, node and SRLG risks) with the transmission of a very small amount of bandwidth information. Our technique is completely distributed; it balances the computations on the different nodes of the topology and is easy to be deployed.

Simulations show that with the transmission of a small vector of bandwidth information per link, the rate of rejected backup LSPs is low and close to the ideal.

Keywords-- **network, local protection, SRLG, MPLS, bandwidth sharing, backup LSP**

I. INTRODUCTION

Today's applications (IP telephony, video on demand, etc) are very sensitive to the disruption of communications and thus must run on reliable and fault-tolerant networks. Hence, when a network component fails, all the communications using that component must be restored rapidly and before the disruption of services supported by the affected communications. For that, various proactive protection techniques [1, 2] are developed. The principle of these techniques is to precompute (and generally pre-configure) backup paths before a failure happens. In this way, at the detection of a failure nodes switch traffic from the affected primary paths to their backup paths quickly and without performing any computation or configuration.

Proactive protection schemes can be grouped in two classes: global (end-to-end) and local [1]. At failure detection with global schemes, a notification message is sent to the source node (of the primary affected path) which switches traffic from the primary affected path to its backup path. Obviously, this protection scheme increases the restoration time and generates additional notification messages to cope with failures. Such problems are resolved with the use of local protection schemes in which one backup path is established for each node of the primary path. Hence, when a node detects a failure (on its downstream primary link and/or primary node), it treats it locally and rapidly by switching traffic from the affected primary paths to their backup paths without any control plane notification.

To ensure enough resources (especially the bandwidth) after the restoration from a failure, the backup paths must reserve the resources they need. If we consider that each backup path has its own exclusive resources, the network will be overloaded quickly since the available resources decrease rapidly. However, with few realistic assumptions, the available resources in the network can be increased significantly. Typically and under the hypothesis of single failures in the network, all the backup paths protecting against different failure risks (network components which can fail simultaneously) can share their resource allocation on the common links they traverse. Indeed, such backup paths cannot be active at the same time (since at most one network component is falling at any time) and as a result, they will not use their resources simultaneously.

With the advent of MPLS in the last decade [3], the protection and resource optimization functionalities are provided in an efficient manner. Indeed, MPLS offers a great flexibility for choosing paths (called Label Switched Paths or LSPs). For instance, resource optimization can be achieved by choosing paths that increase resource sharing. Moreover, the possibility to pre-configure backup LSPs allows fast restoration (50 ms approximately).

Two types of backup LSPs are defined for MPLS local protection [4]: next hop backup LSP (NHOP LSP) and next next hop backup LSP (NNHOP LSP). A NHOP LSP (resp. NNHOP LSP) is a backup LSP protecting against link failure (resp. node failure); it is setup between a primary LSR called Point of Local Repair (PLR) and one primary LSR downstream to the PLR (resp. to the PLR next-hop) called Merge Point (MP). Such backup LSP bypasses the link downstream (resp. the node downstream) to the PLR on the primary LSP. When a link failure (resp. node failure) is detected by a node, this later activates locally all its NHOP (resp. NNHOP) backup LSPs by switching traffic from the affected primary LSPs to their backup LSPs.

To account for the bandwidth sharing when the backup LSP are computed, some bandwidth information must be transmitted to the backup path computation entities. In distributed environments, the advertisement of such bandwidth information is very costly since it increases significantly the network load. To decrease the quantity and the frequency of advertisements, we propose in this article a

Distributed Bandwidth Sharing (DBS) heuristic for MPLS backup LSP computation. Our heuristic treats all the failure risks types and computes the backup LSPs efficiently by sharing the bandwidth when it is possible. With the advertisement of a small amount of bandwidth information in the network, our heuristic is capable to predict and select the links which can be used to establish a new backup LSP (links verifying the bandwidth constraints). Moreover, the DBS heuristic is scalable, very fast in its computations and balances equitably and efficiently the computation tasks on the nodes of the network. It simplifies also the configurations by the use of the on-line mode which preserves the computed LSPs.

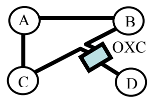
The rest of this article is organized as follow. Section 2 describes the three types of failure risks which gather network components in entities failing simultaneously. By relying on the hypothesis that one risk at most is failing at any time, we give the formulas allowing the computation of the minimal protection bandwidth to be reserved on each arc. In section 3, we review works related to bandwidth sharing. Then, we describe and explain in section 4 the principles of the DBS heuristic. In section 5, we present simulation results and analysis. Finally, section 6 is dedicated to the conclusions.

II. Failure Risks and Bandwidth Sharing

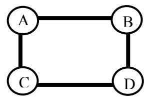
In a real network, the failures of physical components are inevitable. To cope efficiently with such physical failures in a logical layer level (MPLS level), it is judicious to inform the backup path computation entity *BPCE* (entity computing the backup paths) of the logical components which share same physical components. This allows the determination of the logical components which can fail simultaneously under the single physical failure hypothesis (adopted in this article).

In Fig. 1, two topologies corresponding to the same network are depicted. The first one is obtained according to the Data Link neighborhood information (Fig. 1 (a)) while we used only the (IP) Network neighborhood information to deduce the second topology (Fig. 1 (b)). As we see, the optical crossconnect *OXC* in Fig. 1 (a) is not visible by the Network (and MPLS) layer. This crossconnect is an optical component used to connect router *D* to routers *B* and *C*. Thus, the Network link *D-B* (resp. link *D-C*) in Fig. 1 (b) corresponds to the optical path *D-OXC-B* (resp. optical path *D-OXC-C*) in Fig. 1 (a). As a result, we conclude that the two Network links *D-B* and *D-C* fails simultaneously if the crossconnect *OXC* fails. In order to address this shared risk case, links can be grouped together in a Shared Risk Link Group (SRLG), as defined in [5]. For instance, the two logical links *D-B* and *D-C* in Fig. 1 (b) are grouped in one SRLG risk since they go through the same crossconnect *OXC*. These two links are said to be as *SRLG diverse*.

Two others types of failure risks exist: link risk and node risk. The first failure risk corresponds to the risk of a logical link failure due to the breakdown of an exclusive



(a) Physical topology



(b) Logical topology

Figure 1. Topology correspondence

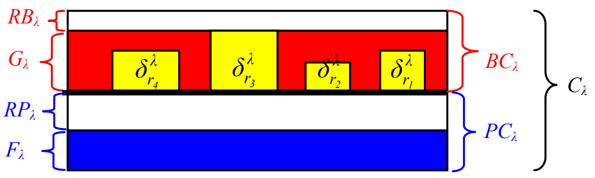


Figure 2. Bandwidth allocation on an arc λ

physical component to the link. The second failure risk corresponds to the risk of a logical node failure.

To determine the minimal protection bandwidth to be reserved on arcs, [6] defines two concepts: *the protection failure risk group (PFRG)* and *the protection cost*. The *PFRG* of a given arc λ, noted *PFRG* (λ), corresponds to a set which includes all the risks protected by the backup LSPs traversing the arc λ. The protection cost of a risk r on an arc λ, noted δ_r^λ, is defined as the cumulative bandwidth of the backup LSPs which will be activated on the arc λ after a failure of the risk r. For a SRLG risk *srlg* composed of links $(l_1, l_2, .., l_n)$, the protection cost on an arc λ is determined as follow:

$$\delta^\lambda_{srlg} = \sum_{0<i\le n} \delta^\lambda_{l_i} \tag{1}$$

To cope with any failure, a minimal quantity of protection bandwidth G_λ must be reserved on the arc λ. Such quantity G_λ is determined as the maximum of the protection costs on the arc λ.

$$G_\lambda = \underset{r\in PFRG(\lambda)}{Max}\ \delta^\lambda_r \tag{2}$$

In order to better control (explicitly specify) the quantity of bandwidth used for protection and to separate the tasks of primary LSP computation from those of backup LSP computation, the bandwidth capacity C_λ on arc λ can be divided in two pools: primary bandwidth pool and protection bandwidth pool (Fig. 2). The primary bandwidth pool on an arc λ has a capacity PC_λ and it is used to allocate bandwidth for primary LSPs. The protection bandwidth pool on an arc λ has a capacity BC_λ and it is used to allocate bandwidth for backup LSPs.

To ensure the respect of bandwidth constraints, the reserved protection bandwidth on each arc λ must verify:

$$G_\lambda \le BC_\lambda \tag{3}$$

To keep inequality (3) valid (invariant) after the setup of a backup LSP b of bandwidth *bw (b)* which protects against the risks in *R (b)*, only the arcs (λ) verifying the following inequality can be selected to be in the LSP b:

$$\underset{r\in R(b)}{Max}\ \delta^\lambda_r \le BC_\lambda - bw(b) \tag{4}$$

Finally, we define the residual protection bandwidth RB_λ as a quantity of bandwidth which is not used on an arc λ. It is determined as follow:

$$RB_\lambda \le BC_\lambda - G_\lambda \tag{5}$$

III. Related Works

In the last years, the domain of bandwidth sharing met a great interest. Hence, many works propose techniques for the on-line computation of the backup LSPs sharing bandwidth. These techniques can be classed in two categories: centralized techniques and distributed techniques.

In a centralized environment, the server can memorize the topology information and the properties of the primary and backup LSPs. With such data, the problem of bandwidth sharing can be formulated using integer linear programming. An example of such formulation for end-to-end protection is described in [7]. In this formulation, the primary path and its backup path are computed so that the additional bandwidth they need is minimal. Even though this technique increases the bandwidth availability, its utilization is limited to small networks. Indeed, the use of a centralized server does not scale and presents some well known disadvantages like the formation of bottlenecks around the server (non scalable) and the sensitivity to the failure or the overload of the server.

To get around the problems encountered in centralized environments, various distributed techniques are developed. In [8], Kini suggests to advertise the topology information, the primary bandwidth, the capacities and all the protection costs of the topology arcs ($\{\delta_r^\lambda\}_{\lambda,r}$) in the network. In this way, each node has a complete knowledge of the information necessary to the backup LSP computation and as a result, it can use a similar model as in the centralized environment to perform the computations. This computation technique allows the optimization of the additional bandwidth but it overloads the network with messages transmitting all the protection costs (the number of protection costs on an arc maybe large and up to the number of failure risks in the network). With a similar technique, [6] proposes to broadcast the structures and properties of backup LSPs instead of the protection costs. The size and the number of messages broadcast in the network are noticeably decreased with the use of the facility backup protection but they remain high and awkward in large networks. In order to decrease the quantity of information distributed in the network, [9] proposes the PCE-based MPLS-TE fast reroute technique. With this last technique, a separate PCE (path computation element) is associated to each failure risk in order to compute backup LSPs which will be activated upon the failure of that risk. This computation technique is effective when SRLGs in the network are disjoint. Otherwise, additional communication between PCEs is necessary to compute the NNHOP LSPs protecting against links appearing in SRLGs. Moreover, the requirement of the technique to manage the non disjoint SRLGs by a same PCE centralizes the computations and introduces identical problems as that encountered in the centralized environments.

To decrease the quantity of information advertised in the network with the distributed techniques, other approaches using heuristics have emerged. In such heuristics, the bandwidth information (protection costs especially) is aggregated before its transmission in the network. In this manner, the frequency and the size of the advertised information are small and easy to be transported in IGP-TE protocols [10, 11]. All the heuristics presented here assume that each node knows the protection costs on its adjacent arcs (its incoming or out-going arcs). These protection costs can be obtained easily without overheads when the backup LSPs are signaled.

The first obvious heuristic sharing bandwidth is based on the residual bandwidth. In [8], Kini proposes to approximate the protection cost δ_r^λ of a risk r on an arc λ by the maximum of protection costs (G_λ) on that arc. In this way, only one value per arc is distributed in the network. The advantage of this technique is that it does not require any modification to the IGP-TE or signaling protocols to be implemented. An arc λ is selected to be in a backup LSP of bandwidth bw and protecting against risk r if $bw \le R_\lambda$. Despite of its simplicity, this heuristic presents a very high blocking probability (ie. the number of backup LSPs that can be built with this heuristic is low). In order to decrease the blocking probability, [8] proposes the Kini's improved heuristic which enhances the estimation of the protection costs. Hence, on each arc λ of the topology, the protection cost δ_r^λ is approximated by the minimum between the backup bandwidth G_λ on the arc λ and the primary bandwidth F_r reserved on the risk r. In practice, this last heuristic has performances comparable to those of the residual bandwidth-based heuristic since the primary bandwidth F_r reserved on the risk r (case of a node) is often higher than the backup bandwidth G_λ on the arc λ.

IV. DISTRIBUTION BANDWIDTH SHARING (DBS) HEURISTIC

To compute the backup LSPs protecting against the three types of failure risks (link, node and SRLG), we developed the DBS heuristic. With this DBS heuristic, one backup path computation entity ($BPCE_n$) is running on each (MPLS) node (n) of the topology. Each entity $BPCE_n$ is responsible of the computation of the backup LSPs protecting against the failure of node n and/or against the failure of the incoming arcs to node n. After each computation of a backup LSP protecting against the failure of an incoming arc m->n, $BPCE_n$ sends the structure of the determined path to node m (which is the PLR) in order to configure it. In this manner, the extremity nodes (n, m) of each link n-m know the structures of all the backup LSPs protecting against the failure of link n-m (ie. against the failures of the arcs n->m and m->n). Moreover, each $BPCE_n$ knows the structures of the backup LSPs protecting against the node n since the computations of such backup LSPs are done by the same entity $BPCE_n$. As a result, each $BPCE_n$ can deduce easily the protection costs of the risks n and its adjacent links (on all the arcs of the topology). As explained in [9], such protection costs allow the computation of backup LSPs protecting against the failure of node n and/or the failure of the adjacent links to n which are not in SRLGs. In fact, $BPCE_n$ can apply inequality (4) to select the arcs which can be used to determine these backup LSPs.

When the link to be protected by $BPCE_n$ appears in several SRLGs, additional information must be communicated to $BPCE_n$. This information must allow the computation of the maximum protection costs of the SRLGs including the protected link on the arcs of the topology (inequality (4)). An easy way to determine such maximums consists either to centralize computations of LSPs protecting against non disjoint SRLGs (as in [9]) or to advertise all the SRLG protection costs in the network. These two solutions can be interesting for small networks where the number of SRLGs is low.

In networks containing a great number of SRLGs however, both centralized computations and advertisement of SRLG protection costs have severe consequences on the network scalability. The use of the DBS heuristic, which reduces and limits the quantity (size and frequency) of information to be advertised, is an interesting solution to

Algorithm 1. Extremity node e_λ of an arc λ

```
parameters: Integer: $Ts_\lambda$ ;
            Integer: x ; {maximal size of the x_vector}
inputs: Const SRLG_id generic_srlg = "-" ;
        Sorted_list <SRLG_id, Integer> costs ; {SRLG protection
        costs, on the arc $\lambda$, sorted by descending order}
variables:
  Sorted_list <SRLG_id, Integer> old_x_vector , new_x_vector ;
begin_algorithm
  old_x_vector = compute_vector (costs, $x_\lambda$, $Ts_\lambda$) ; {procedure below}
  while (true)
    wait ( costs ) ; {wait for a change in the sorted list costs}
    new_x_vector = compute_x_vector (costs, $x_\lambda$, $Ts_\lambda$) ;
    if (old_x_vector $\neq$ new_x_vector) then
      broadcast ($\lambda$, new_x_vector) ;
      old_x_vector = new_x_vector ;
    endif
  endwhile
end_algorithm

procedure compute_x_vector ; {returns the x_vector}
  input: Sorted_list <SRLG_id, Integer> costs ;
         Integer $x_\lambda$, $Ts_\lambda$ ;
  output: Sorted_list <SRLG_id, Integer> x_vector ;
  variables:
    Integer i ;
    <SRLG_id, Integer> element ;
  begin_procedure
    x_vector = costs.elements (1, $x_\lambda$) ; {builds and affects to x_vector a
    new list including the element from 1 to $x_\lambda$ of the sorted list costs}
    i = x_vector.size () ; {i $\leq$ costs.size ()}
    if (i== 0 or costs.element_at (i).cost () $\leq$ $Ts_\lambda$) then
      while (i > 0 and costs.element_at (i).cost () $\leq$ $Ts_\lambda$) do
        x_vector.erase_element_at (i) ;
        i = i – 1 ;
      endwhile ;
      return x_vector ;
    endif
    if (i < costs.size () and costs.element_at (i + 1).cost () > $Ts_\lambda$) then
      while (i > 1 and costs.element_at (i -1).cost () ==
                                  costs.element_at (i).cost ()) do
      {keep only one element containing the $x^{th}$ highest SRLG cost}
        x_vector.erase_element_at (i) ;
        i = i – 1 ;
      endwhile
      x_vector.element_at (i).SRLG () = generic_srlg ;
    endif
    return x_vector ;
  end_procedure
```

Algorithm 2. Each node receiving an *x_vector*

```
inputs: Const SRLG_id generic_srlg = "-" ;
        Array (Arc_set * SRLG_set) $\rightarrow$ Integer costs ; {SRLG costs}
variables:
  Sorted_list <SRLG_id, Integer> x_vector ;
  <SRLG_id, Integer> element ;
  Integer size, i, min_cost ;
  Arc $\lambda$ ;
begin_algorithm
  ($\lambda$, x_vector) = receive () ; {receives a broadcast message and returns
  the values of the arc and the x_vector included in this message}
  size = x_vector.size () ;
  element = x_vector.element_at (size) ;
  if (element.SRLG () $\neq$ generic_srlg) then min_cost = 0 ;
  else min_cost = element.cost () ;
  endif
  for all (SRLG_set a_srlg) do costs [$\lambda$, a_srlg] = min_cost ; endfor
  i = 1 ;
  while (i $\neq$ size)
    element = x_vector.element_at (i) ;
    costs [$\lambda$, element.SRLG ()] = element.cost () ;
    i = i+ 1 ;
  endwhile
  element = x_vector.element_at (size) ;
  if (element.SRLG () $\neq$ generic_srlg) then
    costs [$\lambda$, element.SRLG ()] = element.cost () ;
  endif
end_algorithm
```

resolve the network scalability problem. In this heuristic, we fix a threshold Ts_λ on each arc λ of the topology and we send only, for each arc λ, the x_λ highest SRLG protection costs (called the *x_vector*$_\lambda$) which are larger than the threshold Ts_λ. Obviously, the choice of the two DBS heuristic parameters x_λ and Ts_λ is very important to control the quantity of information to be advertised in the network and the degree of bandwidth sharing. In general, the threshold is a constant depending on the protection bandwidth capacity in each arc. It may be chosen so that the quantity of bandwidth desired in any request of backup LSP computation is always lower or equal than the difference between the protection capacity and the threshold (BC_λ - Ts_λ). Formally:

$$\forall\, bw : bw \leq BC_\lambda - Ts_\lambda \qquad (6)$$

where *bw* is the maximal quantity of bandwidth that a backup LSP can clam.

Concerning the second parameter x_λ, it regulates the degree of bandwidth sharing and the quantity of bandwidth information to be advertised in the network. The higher is its value, the better is the bandwidth sharing and more information may be advertised in the network. For small values of x_λ, the bandwidth sharing is decreased but the quantity of information to be advertised is very small.

In addition to the advantage of decreasing the advertised information, DBS heuristic is simple to be implemented. In fact, the introduction of slight extensions to IGP-TE protocols allows the flooding of *x_vectors*. Typically, we propose to define a new arc (unidirectional link) TLV to advertise the *x_vector*$_\lambda$ of the arc λ. Such TLV will be conveyed in the TE LSA for OSPF-TE [10] and in IS reachability for ISIS-TE [11].

Alg. 1 and Alg. 2 summarize the behavior of the nodes implementing the DBS heuristic. Let us illustrate the operation of these algorithms by an example. Fig. 3 shows the protection bandwidth pool of an arc λ. This arc λ is used in the protection of six SRLG risks and several node and link risks (only the positive SRLG protection costs on the arc λ are shown in the figure). In order to compute efficiently the backup LSPs, the extremity (outgoing) node e_λ of the arc λ floods within the IGP-TE the x_λ *highest* values of the SRLG protection costs (and their corresponding SRLG risks) which are larger than the threshold. Dependently of the value of the $(x_\lambda+1)^{th}$ highest SRLG cost (*x*$_\lambda$*_plus_1_cost*), we distinguish essentially two cases: *doubtful cases* where *x*$_\lambda$*_plus_1_cost* $> Ts_\lambda$ (equivalent to $x_\lambda \leq 2$ in Fig. 3) and *sure cases* where *x*$_\lambda$*_plus_1_cost* $\leq Ts_\lambda$ (equivalent to $x_\lambda > 2$ in Fig. 3).

A. Doubtful case (*x*$_\lambda$*_plus_1_cost* $> Ts_\lambda$ or $x_\lambda \leq 2$)

With $x_\lambda = 2$, the extremity node e_λ transmits (at most) the two highest values of the SRLG protection costs which are larger than the threshold Ts_λ. These two values form the *x_vector*$_\lambda$ and they are flooded each time the *x_vector*$_\lambda$ changes (Alg. 1). For the arc λ in Fig. 3, e_λ advertises the

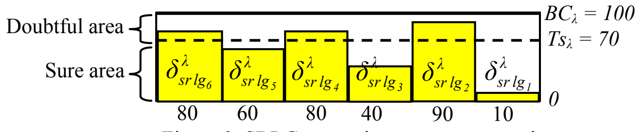


Figure 3. SRLG protection costs on an arc λ

following information: [$(srlg_2, 90), (\text{-}, 80)$]. The character '-' refers to the *generic srlg* which is a fictitious SRLG. It is transmitted within the last couple of the x_vector_λ when the $(x+1)^{th}$ highest SRLG cost is larger than the threshold.

Each node receiving the information transmitted by e_λ updates its SRLG protection cost database and approximates the SRLG protection costs on the arc λ as follows (Alg. 2):

$$(\delta^\lambda_{srlg_2} = 90) \wedge (\forall\, srlg_i \neq srlg_2 : \delta^\lambda_{srlg_i} = 80)$$

Due to the localization of the $(x+1)^{th}$ highest SRLG protection cost (equal to 80) in the doubtful area (protection costs between the threshold and the protection bandwidth), a $BPCE_i$ receiving the x_vector_λ above can reject the arc λ by mistake when it computes a backup LSP (presence of the *generic srlg*). For instance, any computation of a backup LSP protecting against the failure of a link appearing in SRLGs of the set $S_{SRLG} \setminus \{srlg_1, srlg_3, srlg_5\}$ (S_{SRLG} includes all the SRLGs of the topology) excludes by mistake the arc λ if the bandwidth desired is in $]BC_\lambda - 80, BC_\lambda - Ts_\lambda]$ (i.e. in $]20, 30]$). For all the other cases, the different $BPCE_i$ decide without mistake if the arc λ can be selected to be in a new backup LSP.

B. Sure case ($x_\lambda\_plus\_1\_cost \leq Ts_\lambda$ or $x_\lambda > 2$)

With $x_\lambda = 3$, the extremity node e_λ transmits (at most) the three highest values of the SRLG protection costs which are larger than the threshold Ts_λ (Alg. 1). For the arc λ in Fig. 3, e_λ broadcasts the following information: [$(srlg_2, 90), (srlg_4, 80), (srlg_6, 80)$]. The third maximal value (equal to 80) of the SRLG protection costs is not transmitted with the *generic srlg* since the fourth maximal SRLG protection cost (equal to 60) is lower than the threshold.

Each node receiving this x_vector_λ updates its SRLG protection cost database and approximates the SRLG protection costs on the arc λ as follow (Alg. 2):

$$\begin{cases} (\delta^\lambda_{srlg_2} = 90) \wedge (\delta^\lambda_{srlg_4} = 80) \wedge (\delta^\lambda_{srlg_6} = 80) \\ \forall\, srlg_i (srlg_i \neq srlg_2 \wedge srlg_i \neq srlg_4 \wedge srlg_i \neq srlg_6) : \delta^\lambda_{srlg_i} = 0 \end{cases}$$

Due to the localization of the $(x+1)^{th}$ SRLG protection cost (equal to 60) in the sure area (protection costs between zero and the threshold), a $BPCE_i$ receiving the x_vector_λ above deduces without mistake if the arc λ can be used to build a new backup LSP. Indeed, for backup LSPs protecting against a link which appear in an SRLG belonging to $\{srlg_2, srlg_4, srlg_6\}$, $PBCE_i$ knows the maximal value of the SRLG protection cost on the arc λ since it received it (90 if the link appear in $srlg_2$, 80 otherwise). For the backup LSPs which protect against the failure of a link appearing in SRLGs of the set $S_{SRLG} \setminus \{srlg_1, srlg_3, srlg_5\}$ $PBCE_i$ selects always the arc λ (without mistake) when it computes a new backup LSP.

At this point, we see that the transmission of the three highest SRLG costs on the arc λ is sufficient to decide, without mistake, if the arc λ can be used in the establishment of a new backup LSP.

With regards to the potential impact on the scalability of the IGP-TE, we affirm that the frequency and the size of the information flooded are significantly decreased. Indeed, the size is decreased since at most the x_λ highest values of the SRLG protection costs are transmitted in the network, for each arc λ. Moreover, the x_vector_λ does not change at each establishment of a new backup LSP going through the arc λ. This decreases the frequency of advertisements since it is not necessary to flood an x_vector_λ as long as it keeps its value.

Finally, we note that we can decide without mistake if the arc λ can be used in the establishment of a new backup LSP when x_λ is infinite. In such case, the x_vector_λ includes only the SRLGs protection costs (and their corresponding SRLGs) which are higher than the threshold. Moreover, the values of parameters $\{x_\lambda\}$ can be different from an arc to another. For instance, the loaded arcs can use high values of x_λ whereas it is possible and desired to use small values of x_λ for less loaded arcs.

V. Analysis and Simulation Results

A. Simulation model

In order to examine the performances of our bandwidth sharing heuristic, we compare it to the improved Kini heuristic (IKH) [8]. In our simulations, we divide the total bandwidth available on arcs in two pools (protection pool and primary pool) before applying the two compared heuristics (the DBS and IKH). All arcs of the topology have same primary and protection capacities. The primary capacity is assumed infinite whereas the protection capacity is equal to 100 units.

The topology network used for the comparison, with 24 nodes and 53 bidirectional links, is shown in Fig. 4. In this topology, we created 30 SRLGs (ellipses in Fig. 4) so that the protection against any SRLG failure remains possible.

The traffic matrix is generated randomly and consists of LSPs asking for quantities of bandwidth uniformly distributed between 1 and 10. The LSPs are computed with the use of the Dijkstra's shortest path algorithm. Their ingress and egress nodes are chosen randomly among the nodes of the network.

Three variants of the DBS heuristics are used in the simulations. The first one $DBS_{(\infty, 90)}$ uses a threshold ($Ts = 90$) and an infinite x_vector ($x = \infty$) on all the arcs. The second variant $DBS_{(2, 0)}$ uses an x_vector of maximal size equal to 2 ($x = 2$) but it does not employ the threshold ($Ts = 0$). The last variant $DBS_{(2, 90)}$ uses a threshold ($Ts = 90$) and an x_vector of maximal size equal to 2 ($x = 2$).

Two metrics are used for the comparison: ratio of rejected backup LSPs (RRL) and number of protection bandwidth parameter changes per backup LSP establishment (NPC).

The first metric (RRL) measures the ratio of backup LSPs that are rejected because of lack of protection bandwidth. This metric is computed as the ratio between the number of backup LSP requests that are rejected and the total number of backup LSP requests.

The second metric measures the number of changes in the bandwidth protection parameters (at each establishment

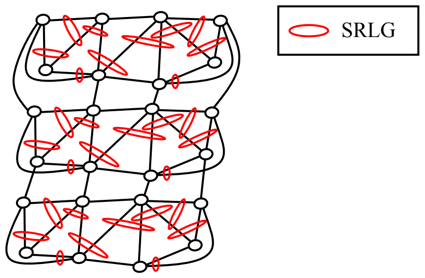


Figure 4. Test topology

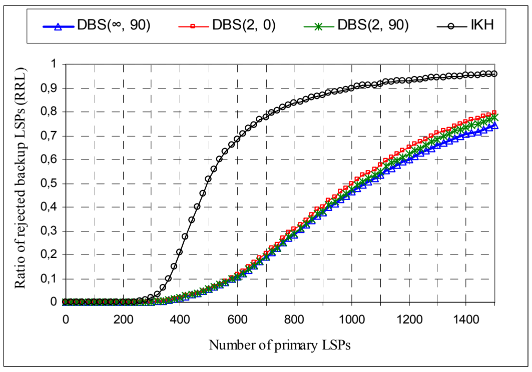


Figure 5. Ratio of rejected backup LSPs (RRL)

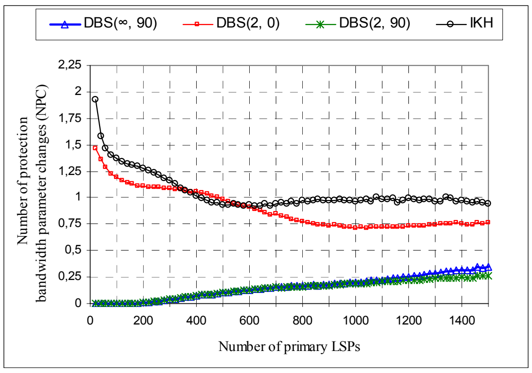


Figure 6. Number of protection bandwidth parameter changes (NPC)

of a new backup LSP) which allow the estimation of the different protection costs. A growth of the NPC's values implies the increase of the size and frequency of messages advertising the protection bandwidth information.

For the DBS heuristic, each change in the *x_vectors* increases the NPC whereas only the changes in the minimal protection bandwidth G_λ allocated on arcs are counted with the IKH.

At each establishment of 20 primary LSPs, the two metrics NPC and RRL are computed for each heuristic.

B. *Results and analysis*

Fig. 5 shows the evolution of RRL as a function of the number of primary LSPs established in the network. As we see, the RRL values of the three DBS heuristics ($DBS_{(\infty, 90)}$, $DBS_{(2, 0)}$ and $DBS_{(2, 90)}$) are lower and better than those of IKH. This is due to the overestimation of the protection costs with the IKH heuristic. Indeed, on each arc of the topology, the IKH approximates the protection costs of all the failure risks by their highest protection cost whereas only some SRLG protection costs are approximated by their x^{th} highest SRLG protection cost with DBS heuristics. Moreover, the quality of the approximation is better with the DBS heuristics since the x^{th} highest SRLG protection cost is always lower or equal to the highest protection cost used in IKH.

With regard to the three variants of DBS heuristics used in simulations, Fig. 5 shows that the RRL values of these variants are very close. The $DBS_{(\infty, 90)}$ which represents the ideal RRL is slightly lower and better than $DBS_{(2, 0)}$ and $DBS_{(2, 90)}$. This is essentially due to the location of the backup LSPs which tend to traverse arcs close to the protected risk. As a result, the number of protection costs which are higher than the threshold on an arc is generally limited. This decreases the *x_vector* sizes allowing a good quality approximation of protection costs. Typically, the advertisement of an *x_vector* of size equal to 2 in our simulations provides an RRL values close to the ideal.

Concerning the second metric NPC, we observe in Fig. 6 that the NPC values of $DBS_{(\infty, 90)}$ and $DBS_{(2, 90)}$ have very low values comparatively to $DBS_{(2, 0)}$ and IKH. This is arisen from the use, in $DBS_{(\infty, 90)}$ and $DBS_{(2, 90)}$, of the threshold which eliminates the flooding of a great number of SRLG protection costs.

When threshold is not used (as in $DBS_{(2, 0)}$), the NPC values of the DBS heuristic increase but remain lower than those of IKH in most cases. Even though we chose a topology with a great number of SRLGs, the two highest SRLG protection costs change less rapidly than the highest protection cost.

Finally, note the great similarity between the NPC values of $DBS_{(\infty, 90)}$ and $DBS_{(2, 90)}$. Such NPC resemblance justifies in great part the similarity between the RRL values of the three variants of DBS.

VI. Conclusion

In this article, we proposed a novel distributed heuristic, called DBS, for backup LSP computation. Our heuristic shares efficiently the bandwidth between backup LSPs and allows the protection against the three types of failure risk (link, node and SRLG). It is also scalable and easy to be implemented. Indeed, with the advertisement of limited quantity of bandwidth information in the network, DBS is capable to estimate efficiently the protection costs necessary for backup LSP computation.

Simulation results show that the heuristic decreases noticeably the number of rejected backup LSPs and the frequency of advertisements when the threshold and the highest sizes of *x_vectors* are well chosen.

VII. REFERENCES

[1] P. Meyer, S. Van Den Bosch, and N. Degrande. "High Availability in MPLS-Based Networks". Alcatel Telecommunication Review, 4th Quarter 2004.

[2] S. Ramamurthy, B. Mukherjee. "Survivable WDM Mesh Networks". Part I - Protection. In: IEEE INFOCOM. Volume 2, pages 744-751, 1999.

[3] E. Rosen, A. Viswanathan, R. Callon. "Multiprotocol Label Switching Architecture". RFC 3031, January 2001.

[4] P. Pan, G. Swallow, A. Atlas. "Fast Reroute Extensions to RSVP-TE for LSP Tunnels". RFC 4090, May 2005.

[5] K. Kompella, Y. Rekhter. "Routing Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)". RFC 4202, October 2005.

[6] J. L. Le Roux, G. Calvignac. "A method for an Optimized Online Placement of MPLS Bypass Tunnels". IETF draft, draft-leroux-mpls-bypass-placement-00.txt, February 2002.

[7] M. Kodialam, T. V. Lakshman. "Dynamic Routing of restorable Bandwidth-Guaranteed Tunnels Using Aggregated Network Resource Usage Information". IEEE/ACM Transactions On Networking, vol. 11, N. 3, June 2003.

[8] S. Kini, M. Kodialam, T. V. Lakshman, S. Sengupta, C. Villamizar. "Shared Backup Label Switched Path Restoration". IETF draft, draft-kini-restoration-shared-backup-01.txt, May 2001.

[9] J. P. Vasseur, A. Charny, F. Le Faucheur, J. Achirica, J. L. Le Roux. "Framework for PCE-based MPLS-TE Fast Reroute Backup Path Computation". IETF draft, draft-leroux-pce-backup-comp-frwk-00.txt, July 2004.

[10] D. Katz, K. Kompella, D. Yeung. "Traffic Engineering Extensions to OSPF". RFC 3630, September 2003.

[11] H. Smit, T. Li, "IS-IS Extensions for Traffic Engineering". RFC 3784, June 2004.