Routing and admission control issues for LSPs in MPLS networks

Afef Kotti† and Rached Hamza††,

Techtra Research Unit, Sup'Com, Tunisia.

Summary

Increasing demand for multimedia and distributed applications in recent years has drawn renewed attention to Quality of Services (QoS) routing in IP/MPLS network. This paper presents several new algorithms to be implemented for that purpose. We look, in a first part of the problem at organizing the mapping of Label Switching Paths (LSPs) throughout the network such that we can compromise between several Traffic Engineering (TE) objectives: load balancing, avoiding network bottlenecks, reducing routing cost and minimizing path hop count. In the second part, to bring more and more QoS guarantees to high speed multimedia applications, we have introduced Differentiated Services. In this issue, we propose a new admission control mechanisms based on bandwidth resources. Preemption has been recognized as an important paradigm in our research and it has been conducted into two dimensions: preemption inter Class Type (CT) and preemption across a CT. To achieve significant performance improvement for preemption treatment, we propose two different algorithms: a bandwidth preemption algorithm which selects amount of bandwidth to preempt and an LSPs preemption algorithm which selects the most appropriate LSPs to preempt on the basis on several optimization criteria to avoid rerouting explosion. Simulations studies have been carried out to compare the performance of our approach against existing ones.

Key words:

Traffic Engineering, MPLS, Constrained based routing, Diffserv, Preemption.

1. Introduction

Due to the rapid growth of the Internet and the requirements for QoS of high speed multimedia application, Internet Service Providers need to better engineer their traffic. Traffic Engineering (TE) is defined in [1] as mapping traffic flows onto an existing physical network topology in the most effective way to accomplish desired operational objectives.

Plain IP routing, based on the best effort service is not enough to forward packets along a specific path. The need of Constrained based routing has been imposed. Multi-Protocol Label Switching (MPLS) was been developed to overcome the limitation of conventional routing protocols. MPLS allows the specification of explicit routes through the network, so-called Label Switched Paths (LSPs). From this arises the first motivation of our problem. In fact, we have looked in the first part of our work on the problem of organizing the mapping of paths in an optimal way throughout the network so as to improve backbone efficiency.

Diversity of multimedia and distributed applications has drawn renewed attention to face heterogeneity in networks. Without referring to different classes of services, existing TE researches may be not optimal in a differentiated service environment. To address this issue, the IETF proposed the Diffserv aware MPLS TE (DS-TE) scheme, which performs TE at a per class level. In this issue, traffic flows toward a given destination can be transported on separate LSPs on the basis of service classes and may follow different paths. The TE class is introduced in DS-TE as a pair of a Class Type (CT) and a preemption priority allowed for That CT. In DS-TE, the IETF enforces different Bandwidth Constraints (BCs) on different classes. It specifies three Bandwidth Constraints Models (BCMs) in use for DS-TE: Maximum Allocation Model (MAM) [2], Russian Doll Model (RDM) [3], and Max Allocation with Reservation bandwidth constraints model (MAR) [4]. A BCM provides the rules to support the allocation of bandwidth to individual Class Types (CTs). The use of a given BCM has significant impact on the capability of a network to provide protection for different classes of traffic, particularly under high load conditions.

Bandwidth preemption was recognized as an important piece of DS-TE bandwidth management, but no preemption strategy was proposed in the IETF models. In this context, we propose a new preemption approach on the basis of two algorithms. The first one is a bandwidth preemption algorithm in which preemption is conducted in two dimensions: CT and preemption level. This algorithm would return the amount of bandwidth to preempt from CT *ct* and priority *p*. Having this information, the second algorithm, named LSPs preemption algorithm, selects the most appropriate LSPs to be preempted from the set of LSPs of Class Type *ct* and priority *p*.

The remainder of this paper is organized as follows: Section 2 describes the general problem statements and presents a brief overview of this paper's contributions. Section 3 is dedicated to the description of the Bandwidth Constrained Routing Algorithm (called BCRA) that we

Manuscript received May 5, 2011
Manuscript revised May 20, 2011

use for route computation. Section 4 presents the admission control mechanism proposed for MPLS networks. Section 5 has been dedicated to remind the main rules of the BCM model we use. Section 6 and section 7 analyze respectively the pseudo-code of the bandwidth preemption algorithm and the LSP preemption algorithm. In section 8, simulations studies are carried out to evaluate the performance of our preemption approach. Finally, the work is concluded in section 9.

2. General problem statements and main contributions

The paper presents a bandwidth management framework supporting DS-TE in MPLS networks. Our bandwidth management framework contains two major functions: Route computation and bandwidth management. Although many studies have been conducted on TE, most of them focused on the route selection algorithms [6], [5], [7], [8], [11] and have been put a little effort into DS-TE bandwidth management techniques.

In DS-TE solution, traffic flows can be grouped into classes, and different bandwidth constraints can be applied to each class. By mapping a traffic trunk in a given class on separate LSPs, DS-TE allows the traffic trunk to utilize resources available to that class on both shortest and non shortest paths, and follow path that meet the specific constraints. Obviously, when TE is performed in a Diffserv scenario, a bandwidth manager is indispensable for every network node to enforce different bandwidth constraints for each class, and to flood the network with bandwidth availability information on a per class level. In this context, admission control is imposed. Remind that admission control is the set of actions taken by a network during the service establishment phase to check whether an LSP service request is to be admitted or rejected. A new LSP request is admitted when the desired QoS for the new service can be satisfied, without causing any QoS violation to the already established LSP having higher priority. An additional role of admission control is to optimize the use of network resources.

Admission control should integrate preemption mechanisms to improve efficiency under any offered traffic conditions. In [9], admission control mechanisms are highlighted but no preemption mechanism has been proposed. For this reason, our proposed approach can be considered as the most complete solution in TE. In our present paper, a new preemption policy is proposed. Preemption in our study is conducted in two dimensions: CT and preemption level so that an LSP of CT ct and priority p can not be preempted weather other LSP of weaker precedence remains in the network. The preemption policy we propose is both simple and robust, combining the three main optimization criteria:

- Precedence level: preempt the connection that has the least priority. The QoS of high priority traffic would be better satisfied.
- Number of LSPs to preempt: the number of LSPs that need to be rerouted would be lower to avoid rerouting explosion.
- Amount of bandwidth to preempt: preempt the least amount of bandwidth that still satisfies the request. Resource utilization would be better.

To show how our approach proceeds in details, let us define the network model and operations. We consider an MPLS network where DS-TE is applied. The network is modeled as a directed graph *G(N,L)* where *N* is a set of nodes (routers), and *L* is a set of links between nodes. Each link $i \in L$ is characterized by its bandwidth capacity *C(i)* and its residual bandwidth *Resd_bw(i)*. Furthermore, each LSP is classified into one CT and assigned with one holding priority and one setup priority. Besides the source and destination nodes, an LSP establishment request contains four parameters *(Bw,ct,hp,sp)*, indicating that *Bw* amount of bandwidth is requested for establishment an LSP of CT *ct* at holding priority *hp* and setup priority *sp*. Note that LSPs in the same CT can have different priorities so that they have different priorities to access and retain the resources. We note *Nr(lsp)* the number of components routers of the LSP (noted *lsp*) generated by the route computation algorithm.

Figure 1 resumes the principle steps of our proposed approach. Firstly, when the source node receives an LSP request, it computes a route to the destination based on the network topology, the requested LSP parameters (like the requested bandwidth *Bw*), and some links parameters (like the link residual bandwidth). More details on the route computation technique will be presented in the next section. Secondly, the source sends a request, with parameter the LSP returned by the first step, to all the routers along the path. Thirdly, each router on the path exercises admission control and sends a positive reply to the next router if its outgoing link *i* have enough free bandwidth available to the new connection (*Resd_bw(i)*≥*Bw*). Fourth, if there is not free bandwidth, then the router would activate bandwidth preemption and return positive reply if preemption is successful; otherwise, it would return negative reply. If all routers along the path return positive replies, then the LSP setup is successful without exercising LSPs preemption, and they would reserve the requested bandwidth on the output links. If one or more routers of the returned path exercise bandwidth preemption, *PreemptLSP* variable is set at *true* and in this case of study, LSPs preemption should be activated before establishing the new LSP.

We present in next sections the principle algorithms we have proposed to form a performing LSPs infrastructure that supports differentiated services paradigm.

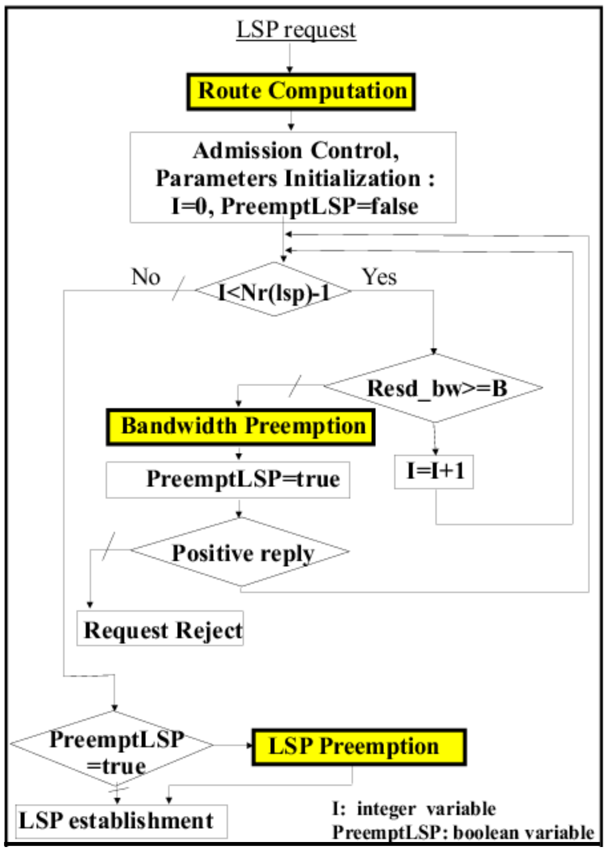

Fig. 1 Flowchart of the admission control procedure.

3. Route computation

3.1 General routing problem

In this section, we discuss routing algorithms for traffic requiring bandwidth guarantees. The goal of the routing algorithm is to find a feasible path if one exists, and to select one that achieves efficient resource utilization if more than one path is available. The most commonly used algorithm for routing LSPs is the shortest path routing. In the shortest path routing, the path with the least number of links between ingress and egress nodes is chosen. The routing algorithm keeps track of the current residual capacity for each link and only those links that have sufficient residual capacity for the new flows are considered. The shortest path algorithm is very simple, but it can also create bottleneck for future flows and lead to severe network under-utilization. In the other hand, load balancing algorithms generate too long paths to avoid network bottlenecks. This fact also affect badly on the network infrastructure since it sometimes augments resource consumption. As we can see, it is difficult to compromise between different TE objectives since optimality criteria sometimes conflict. The motivation for our routing problem arises from the needs of service providers of an approach that can compromise between several TE objectives for the usage efficiency of the network infrastructure. In our approach, we have considered the following objectives: Distributing network load, minimizing path length and reducing path cost. The algorithm has been presented with more details in [11] and it has been compared with the most known existing one. We present briefly, in next paragraph, its basic rules.

3.2 Route computation algorithm

We present the basis ideas of an algorithm for dynamic routing of bandwidth guaranteed flows that we had proposed in [11]. Traditional routing does not take advantage of any knowledge about the traffic distribution or ingress-egress pairs, and therefore can often lead to severe network under-utilization. Algorithms of load balancing, if they exist, they select paths with high number of hops count to avoid network congestion. For this reason, compromising between network load balancing and reducing route hops count will be the most efficient solution for the network infrastructure.

- Distributing network load: for distributing network load, we define a link load parameter *l_load(e)* for the link *e* as follow:

$$l\_load(e) = \frac{reserved\ bandwidth\ on\ \ e}{total\ \ reservable\ bandwidth\ on\ \ e}. \qquad (1)$$

To conserve load balancing we suppose that the TE metric on each link *e* belonging to the graph *G* is defined as:

$$TEML\ \ (e) = f_e(l\ \_\ \mathrm{load(e)}\ \ ). \qquad (2)$$

Where f_e is a positive objective function that we will formulate it later.

The TE metric for a path *i*, called *TEMP(i)*, reflects the cost of path *i* and it is defined as follow:

$$TEMP\ \ (i) = \sum_{e \in E \cap i} TEML\ \ (e). \qquad (3$$

- Reducing routing cost: we assume that network usage cost is solely dependent on the cost of the paths being used. Thus, the TE objective of minimizing network cost is directly translated to the problem of minimizing path cost. Path cost is a static metric and depends directly on its component links costs. Static metrics help in maintaining network stability under high load condition. We denote *Cost(e)* the cost of link *e*. we define *Cost(e)* inversely proportional to its bandwidth capacity.

$$Cost\ (e) = \frac{1}{l\_capacity\ (e)} \cdot \tag{4}$$

Where *l_capacity (e)* the bandwidth capacity of the link *e*.

We formulate now the function f_e as:

$$f_e = l\_load\ (e) \times Cost\ (e)\ \ \forall e \in E \cdot \tag{5}$$

Therefore, *TEML(e)* is reformulated as follow:

$$TEML\ (e) = l\_load\ (e) \times Cost\ (e)\ \ \forall e \in E \cdot \tag{6}$$

- Minimizing path length: to minimize path length, CSPFHopCount assign a weight equal to 1 for each link. We will exploit this fact to formulate our objective function f_e.

$$f_e = l\_load\ (e) \times Cost\ (e) + 1 \forall e \in E \cdot \tag{7}$$

3.3 Pseudo code of the route computation algorithm

We present in this section the pseudo code of the route computation algorithm. The algorithm will return the path taken by an LSP from the source to the destination minimizing the objective function. So, we use the well-known Dijkstra scheme and adapt its formulation to our need. Our route computation algorithm is detailed in Figure 2.

```
Route-computation-algorithm procedure(G(N,L),r(B,ct,hp,sp))
{
Compute the link weight, TEML(e), for all e in L according to equation 7.
Use Dijkstra algorithm to compute the shortest path in the network using TEML(e) as weight of link.
Send the LSP establishment request to the admission Control procedure as shown in figure 1.
}
```

Fig. 2 Pseudo-code of the route computation algorithm.

4. Admission control procedure

Admission control procedure is activated by a router when an LSP establishment request *(Bw,ct,hp,sp)* is received. The admission control decision must be made on the basis of the incoming connection's requested bandwidth, class type *ct*, priorities, and the available bandwidth of the CT. The algorithm returns the decision of either accepting or rejecting the new request. Admission control procedure, as shown in Figure **3**, is activated by all routers along the path. The *while* loop will cross all these routers. For each one, the requested bandwidth *Bw* is compared with the available bandwidth of the routing outgoing link (line 6). If $Bw \leq Resd\_bw$, then there is adequate free bandwidth available to accommodate the new demand, and the request is accepted for the considered router which allow to pass on the following router (I++, line 6). Otherwise, the free bandwidth is inadequate, the preemption is needed and the bandwidth preemption algorithm is activated in line 9. If the latter algorithm is activated at least by one router, the Boolean variable, *preemptLSP* is set at true in line 10.

If one router along the path returns negative reply when bandwidth preemption is activated, the admission control procedure rejects the request. Otherwise, if all the routers along the path return positive reply (*reply*=true) and if bandwidth preemption algorithm is activated at least for one time (*preemptLSP*=true), then the LSP preemption algorithm should be activated to select the most appropriate LSPs for preemption.

```
1   Admission-control procedure(G(N,L),LSP(B,ct,hp,sp))
2   {
3       Integer I=0; reply=true;
4       while((I<Nr(LSP)-1) and (reply==true))
5       {
6           if(Resd_bw≥Bw)  I++;
7           else
8           {
9             Call the Bandwidth Preemption algorithm(reply);
10            preemptLSP=true;
11            if (reply==true)  I++;
12          }
13      }
14      if(reply==false)  Reject the request;
15      elseif(preemptLSP==true)
16      Call LSP preemption algorithm;
17  }
```

Fig. 3 Pseudo-code of the admission control procedure.

We are interested now in presenting preemption algorithms. Before doing this, we have to describe briefly the BCM model we have use to be able to enforce different bandwidths constraints for different classes of traffic.

5. Bandwidth Constraints Model

5.1 Definitions of some notations

Some variables need to be introduced to present the rules defined by the BCM model. Firstly, we define *maxCT* such that (*maxCT+1*) is the number of CTs. Secondly, for a given CT *ct*, we define *maxpriority(ct)* such that (*maxpriority(ct)+1*) is the number of preemption levels in the CT *ct*. For a given CT *ct* ($0 \leq ct \leq maxCT$), let us define a vector *R_ct[ ]* such that its element *R_ct[i]* being the bandwidth reservation on CT *ct* at setup priority *i* ($0 \leq i \leq maxpriority(ct)$). A vector *R[ ]* is also used such that its element *R[ct]* used to record the bandwidth reserved by all the established LSPs of CT *ct*. Finally, a Bandwidth Constraint vector *BC[ ]* is also defined so that the amount of reserved bandwidth by all LSPs of CT *ct* should not exceed the Bandwidth Constraint *BC[ct]*. We note *C(i)* the capacity of the link *i* in term of bandwidth.

5.2 Bandwidth Allocation

We choose to use the MAM model [2] to enforce a maximum allocation constraint for each CT. Briefly, MAM has the following simple rules:

- The bandwidth reserved by all connections of CT *ct* should not exceed *BC[ct]*, the Bandwidth Constraint of CT *ct*.

$$\sum_{i=0}^{maxpriority(ct)} R\_ct[i] \leq BC[ct]. \qquad (8)$$

- The total reserved bandwidth should not exceed the link capacity so that:

$$\sum_{i=0}^{\max\_CT} R[i] \leq C. \qquad (9)$$

- For improving bandwidth efficiency, the sum of the bandwidth constraints is allowed to exceed the link capacity.

$$\sum_{i=0}^{\max\_CT} BC\,[i] \geq C. \qquad (10)$$

The available bandwidth on a link *l* for the TE class is computed as the following, where a TE class is associated with a CT *ct*, and preemption priority *sp*.

$$\text{Resd\_bw(l)} = \text{Min}\left\{\begin{array}{l} \text{BC[ct]} - \sum_{i=0}^{sp} \text{R\_ct[i]}\ , \\ \text{C(l)} - \sum_{i=0}^{maxCT} \text{R[i]} \end{array}\right\}. \qquad (11)$$

6. Bandwidth Preemption

The *Bandwidth Preemption Algorithm* is activated by a router when the requested bandwidth *Bw* in an incoming request (*Bw,ct,hp,sp*) is larger than the unreserved bandwidth of its outgoing link *Resd_bw* (Eq. 11). The Bandwidth Preemption approach offers a new preemption proceeding for the Maximal Allocation Model. The resulting algorithm is called *Bandwidth Preemption Algorithm*.
Let us now concentrate on how the algorithm proceeds. The *Bandwidth Preemption Algorithm* returns a boolean variable *canpreempt*, and three vectors having the same size *nb*. The first one, *vectpreempt_bw[ ]* is used to record the amount of bandwidth to be preempted, and both vectors *vectpreempt_ct[ ]* and *vectpreempt_pp[ ]* record respectively the CTs and the priority levels, from which *vectpreempt_bw* of bandwidth is to be preempted. If bandwidth preemption can occur, the Boolean parameter *canpreempt* is set at *true*, otherwise, it is set at *false*. Figure 4 describes the pseudo code of the *Bandwidth Preemption Algorithm*. In Figure 4, line 6, we begin by searching the possibility of preemption from the amount of bandwith reserved by LSPs of CT *ct* (which is the same CT of the requested LSP). So, we calculate in line 7 the amount of bandwidth we need to preempt (*preemptbw*) from CT *ct*. The *while* loop (lines 8-15) gathers bandwidth for preemption from vector *R_ct[ ]* beginning with the lowest priority (variable *n* initially equal to *maxpriority(ct)*) until a *preemptbw* is located or we arrive at the setup priority *sp* of the requested LSP (*n=sp*). We update *R_ct[ ]* vector, *vectpreempt_bw[ ]*, *vectpreempt_ct[ ]*, and *vectpreempt_pp[ ]* where *vectpreempt_bw[nb]* is the amount of bandwidth to be preempted from the LSPs of CT *vectpreempt_ct[nb]* at priority level *vectpreempt_pp[nb]*.

Line 17 verifies that *Bw* amount of bandwidth granted do not exceed the bandwidth constraint *BC[ct]*. If so, we should withdraw preemption within the CT *ct* and retry it inter CTs. In some cases, we can be in front of the following situations: preemption within the CT *ct* is not possible, or after bandwidth preemptions within the CT *ct*, *preemptbw* is still strictly positive. Facing to these two cases, we should retry preemption across CTs having evidently lower priority than *ct* (line 19).
In Figure 4, line 20, we compute the *Resd_bw* to identify whether granting *Bw* to the new connection would cause the total reservation exceeding the link capacity *C*. If so, *preemptbw* amount of bandwidth needs to be preempted, this is computed in line 20. The *while* loop in line 22 searches bandwidth for preemption beginning from the lowest priority CT (*i=maxCT* in line 21) and the lowest preemption level for each CT (*n=maxpriority(i)* in line 23). In this way, lowest priority connections are preempted before the high priority ones. At last, if *preemptbw* is still strictly positive, the algorithm returns an acknowledgment of preemption failure in lines 36-37 (the Boolean variable *canpreempt* is set at *false*).

```
1Bandwidth preemption algorithm(Boolean canpreempt, int nb,
2float vectpreempt_bw[ ], int vectpreempt_ct[ ],
vectpreempt_pp[ 3]){
4  nb=0; preemptbw=1;R_ct'=R_ct;
5  Resd_bw=BC[ct]-$\sum_{i=0}^{sp} R\_ct[i]$ ;
6  if (Bw>Resd_bw) then
7    n=maxpriority_ct ;preemptbw=Bw-Resd_bw ;
8    while((n>sp) and (preemptbw>0))
9      if (R_ct[n]>0) then
10         vectpreempt_ct[nb]=ct; vectpreempt_pp[nb]=pp;
11         vectpreemp_bw[nb]=Min{preemptbw, R_ct'};
12         preemptbw-=vectpreempt_bw[nb];
13         R_ct'-=vectpreempt_bw[nb]; nb++; n--;
14     endif
15   endwhile
16 endif
17 if ((Bw+$\sum_{j=0}^{maxpriority(ct)} R\_ct'$ [j] )>BC[ct]) then
18     R'=vect_preempt[ct]=null; nb=0;R_ct'=R_ct;
19   elseif (preemptbw>0) then
20   R'= $\sum_{i=0}^{maxCT} R[i]$; Resd_bw=C-R';preemptbw=Max(0,Bw-
21         Resd_bw);            i=maxCT;
22   While((i>ct) and (preemptbw>0))
23     n=Maxpriority(i) ;//the lowest priority of CT i
24     $R_{in}$=Reserved Bandwidth on CT i at preemption level n;
25     while((n>0) and preemptbw>0))
26       if ($R_{in}\neq 0$) then
27         vectpreempt_ct[nb]=i; vectpreempt_pp[nb]=n;
28         vectpreemp_bw[nb]=Min{preemptbw, $R_{in}$};
29         preemptbw-=vectpreempt_bw[nb]; nb++;
30       endif
31       n--;
32     endwhile
33     i--;
34   endwhile
35 endif
36 if (preemptbw>0) then
37   canpreempt=false; plistvar=null;
38 endif
39}
```

Fig. 4 Pseudo-code of the Bandwidth preemption algorithm.

If preemption is accepted (*canpreempt=true*), the *Bandwidth Preemption Algorithm* returns an LSP preemption request with parameters *canpreempt*, *nb*, *vectpreempt_ct*, *vectpreempt_pp*, and *vectpreempt_bw*. This request is sent to *LSPs Preemption Algorithm* to select LSPs for preemption.

7. LSP Preemption

Selection of preempted LSPs is tackled elegantly in this paper to prevent the network from the aggressivity of rerouting process. Figure **5** describes the pseudo code of the *LSPs preemption algorithm*. In this algorithm, the preempted LSPs are chosen on the basis of three criteria: the number of LSPs to preempt, the excess of bandwidth to gain, and the most important aspect of precedence level. Remember that *R_ct[ ]* is a vector of link state summarizing the amount of reserved bandwidth per preemption level in the CT *ct*. For each link, we define two vectors *tab_lsp[ ]*, and *tab_bw[ ]* gathering respectively all LSPs belonging to the same preemption level and the same CT, and their corresponding bandwidths. These LSPs must be kept ordered by decreasing their bandwidth reservations such that *tab_lsp[0]* is the greediest LSP. The *while* loop, (in figure 5, lines 7-27), searches LSPs for preemption by crossing *vectpreempt_ct*, *vectpreempt_pp*, and *vectpreempt_bw* vectors. We define in line 8 a new variable *needbw*. If *needbw* is larger than the amount of bandwidth reserved by all LSPs of CT *vectpreempt_ct* and priority *vectpreempt_pp*, we go to add all these LSPs into the list of LSPs to be cast out, *plistvar*. Otherwise (line 12), we compute the index (called *place*) would have the LSP if its bandwidth reservation were inserted in the vector tab_bw[ ]. Variable *place* is set at zero if the amount of bandwidth to be preempted (*needbw*) is larger than the bandwidth of the greediest LSP. In such case, we preempt LSPs beginning from the greediest one until *needbw* becomes null. If *needbw* is not larger than the bandwidth of the greediest LSP (lines 17-25), we will preempt in this case only one LSP defined by its index *place* in *tab_lsp[ ]*. Clearly, let us show how the LSPs preemption algorithm proceeds using a small example. A new LSP requires preempting *28 Mb* of bandwidth among reserved bandwidth by LSPs belonging to the set of LSPs of CT *2* and priority *2* which are ordered in *tab_bw[ ]* as follows : *tab_bw[0]=50*, *tab_bw[1]=40*, *tab_bw[2]=30*, *tab_bw[3]=25 Mb*. Our preemption scheme, like it was presented above, will select one LSP to preempt which is the third one (*place=2*) whereas other preemption strategies preempt more than one LSP. For example, the preemption procedure implemented in Totem Toolbox [10] selects the tow lowest priority LSPs (having respectively *25*, and *30 Mb* of bandwidth). So, it is clear that with our scheme we gain not only on the number of LSPs to cast out from the network but also the amount of bandwidth to preempt.

8. Performance Analysis

8.1 Simulation context

Simulations studies have been carried out under a TOolbox of Traffic Engineering Methods (TOTEM). In order to address both traffic and resource performance objectives, we consider in our comparison several performance criteria. Firstly, we use the maximum link utilization (*u_max*). This parameter should be minimized since it gives an idea on maximum link load and then on network bottleneck. Secondly, we analyze the mean link utilization (*u_mean*), and the standard deviation of link utilization (*u_stdv*). These two last parameters give an idea on network load balancing. We compare the performance of our algorithm with the Constrained Shortest Path First (CSPF) routing algorithms. CSPF algorithms have the ability to compute shortest paths according to IGP (Interior Gateway Protocol) link metric. The routing algorithm keeps track of the current residual capacity for each link, and only those links that have sufficient residual capacity for the new LSP are considered. When link metric is equal to 1 for each link belonging to the network topology, the CSPF algorithm is called CSPFHopCount. In this case, the path with the least number of links between an ingress and egress router is chosen. Although simple and efficient, CSPF algorithms can create bottlenecks for future LSPs, and may lead to network under-utilization.

```
LSP preemption algorithm{
Call Bandwidth preemption algorithm(canpreempt, nb,
vectpreempt_bw[ ], vectpreempt_ct[ ], vectpreempt_pp[ ]);
if (canpreempt==false then plistvar=null;
else
    i=0;
    while(i<nb)
        needbw=vect_preempt_bw[i];
        if(need_bw≥ $R\_ct$[vectpreempt_pp[i] then
            plistvar.add(all LSPs of priority vectpreempt_pp[i] and
            CT vectpreempt_ct[i]);
        elseif(needbw>tab_bw[0] then
            m=0;place=0;ni=number of LSPs in tab_lsp;
            while((m<ni) and (needbw>0))
                plistvar.add((tab_lsp[m];needbw-=tab_bw[m];m++;
            endwhile
        else
            place=1;
            while(place<(ni-1))
                if(tab_bw[place]≥needbw)
                    place++;
                endif
            endwhile
            place- -;plistvar.add(tab_lsp[place];
        endif
        i++;
    endwhile
endif
return plistvar;
}
```

Fig. 5 Pseudo-code of the LSP preemption algorithm.

We consider a simple network topology composed by 8 nodes as shown in Figure **6**. All links are bidirectional. In this topology, two ingress egress pairs are considered which are (1,8) and (2,8).

In DS-TE approach, up to 8 CTs can be supported. However, to simplify our simulations, we assume a scenario where four CTs can exist simultaneously. The four classes of traffic are denoted by *CT0* (highest priority), *CT1*, *CT2* and *CT3* (lowest priority). Each CT has two possible priorities. Priority *zero* LSP has higher priority and can preempt the priority *one* and priority *two* LSP. Link capacity is considered equal to *200Mb/s* and *BC[0]*, *BC[1]*, *BC[2]* and *BC[3]* are set respectively at *100%*, *75%*, *50%* and 25% of the link capacity.

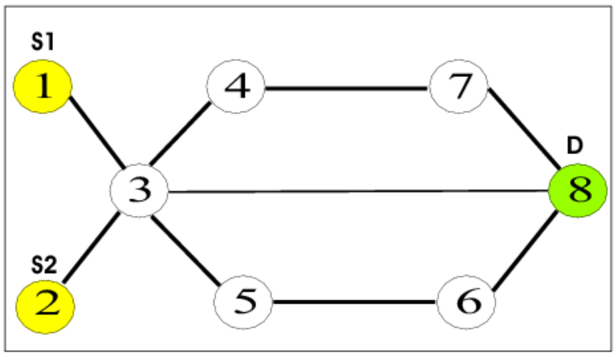

Fig. 6 Simulation network topology.

8.2 Performance under normal load (without DS-TE considerations)

Blocking and preemption probabilities for LSPs setup requests under normal load conditions achieve low level. For this reason, under normal load, we look only at the performance of the routing algorithm without considering DS-TE issues.

In a first time, we look at the performance of our approach, called BCRA-DSTE, under normal load conditions. Hence, we are interested essentially to evaluate the algorithm in term of the maximal link utilization and the standard deviation of link utilization. Each performance criteria is shown separately for each CT.

Figures 7, 8, 9, and 10 show respectively the maximum link utilization of CT0, CT1, CT2, and CT3 LSPs. As we can see, in normal load conditions, the maximal link utilization increases with the increasing of the number of LSPs requests. This result is available for both algorithms BCRA and CSPF. However, as illustrated in these figures BCRA-DSTE achieves the lowest values in maximal loaded link. This result indicates that, BCRA routing algorithm is more performing than CSPF in eliminating network bottlenecks and then increasing the network's ability to honor increasing demands.

CSPF routing forwards packets along the shortest path. Such an approach is sufficient for best effort traffic but makes inefficient use of network resources as it forwards packets along already congested shortest path while longer uncongested path may never be utilized.

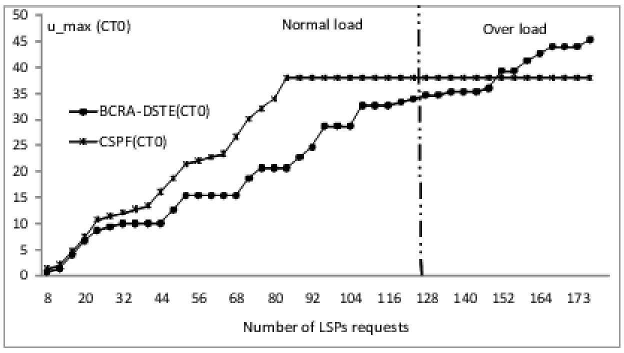

Fig. 7 Maximal link utilization of LSPs of CT0 as function of the number of LSPs requests.

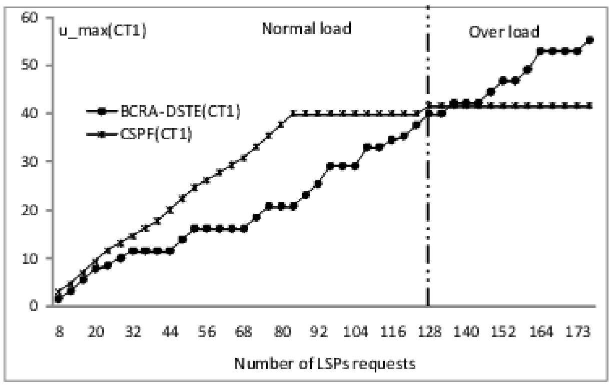

Fig. 8 Maximal link utilization of LSPs of CT1 as function of the number of LSPs requests.

Distribution network load is a difficult objective to meet since it is a dynamic objective and depends on some dynamic path metrics. However, we will see later the efficiency of BCRA-DSTE toward this criterion.

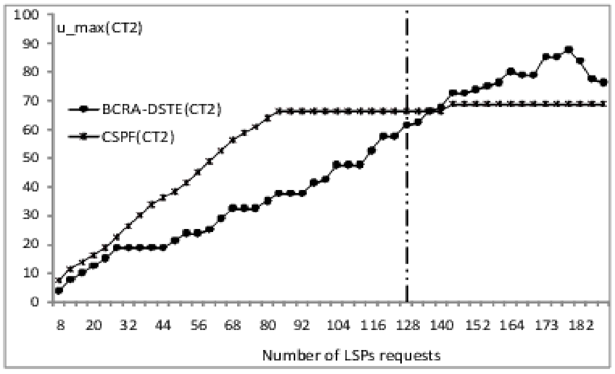

Fig. 9 Maximal link utilization of LSPs of CT2 as function of the number of LSPs requests.

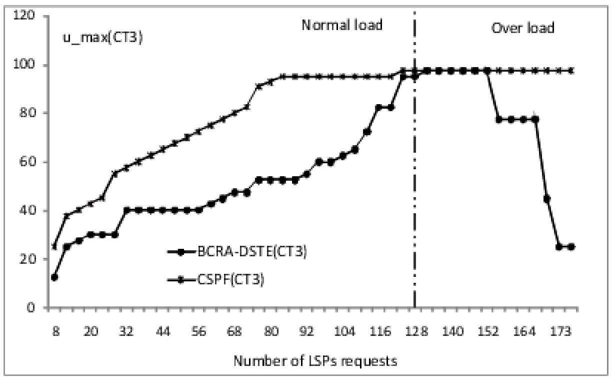

Fig. 10 Maximal link utilization of LSPs of CT3 as function of the number of LSPs requests.

Figures 11, 12, 13, and 14 present the comparison results of *stdv* values for respectively LSPs of CT0, CT1, CT2, and CT3. Requests are generated randomly for each (source, destination) pairs and each algorithm tries to route these requests through the network. We note that, in normal load, BCRA minimizes the *stdv* values even when the number of LSPs requests increases. So, BCRA is better than CSPF by maintaining network load balancing.

8.3 Performance under normal load

- More than one CT is considered: Overload occurs when the traffic on a system is greater than the traffic capacity of the system. Figure 15 plots the number of blocked LSPs as function of the number of LSPs requests. Blocked and preempted LSPs are added together to yield a combined blocked/preempted LSPs. As we can see in such figure, CSPF does not find a route for the 128th request. However, BCRA-DSTE preempts two LSPs to establish the 148th LSP.

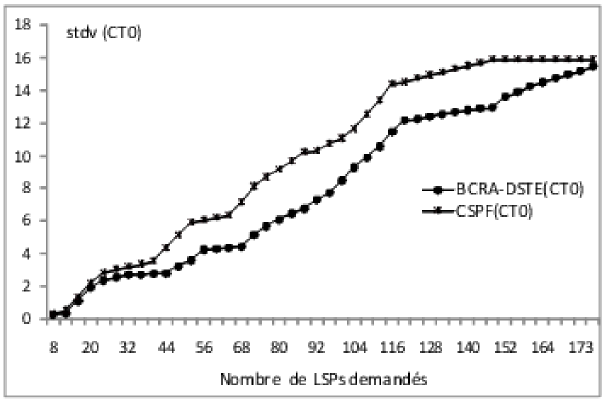

Fig. 11 Standard deviation of link utilization of LSPs of CT0 as function of the number of LSPs requests.

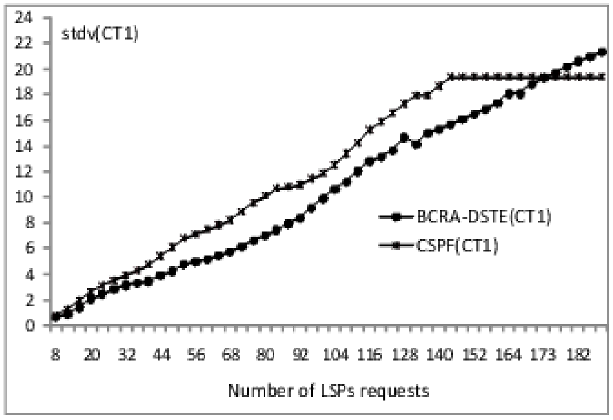

Fig. 12 Standard deviation of link utilization of LSPs of CT1 as function of the number of LSPs requests.

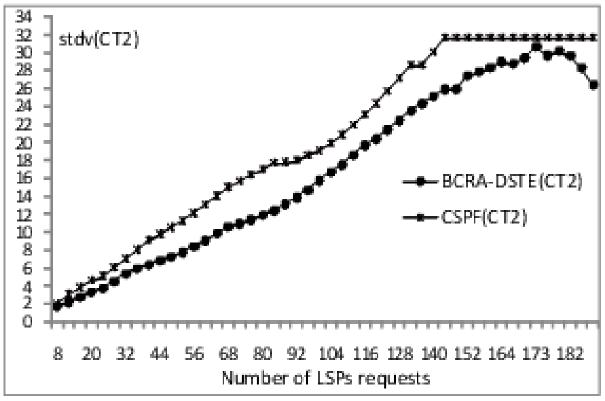

Fig. 13 Standard deviation of link utilization of LSPs of CT2 as function of the number of LSPs requests.

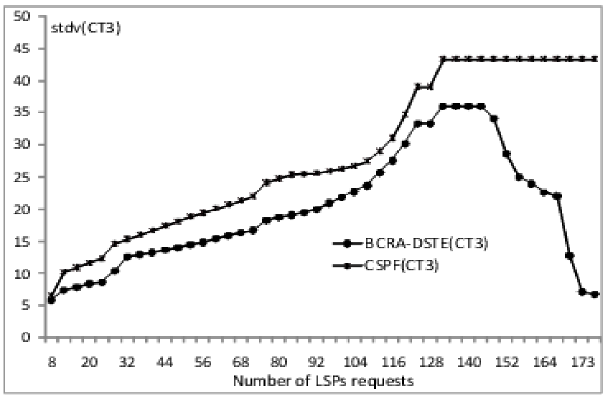

Fig. 14 Standard deviation of link utilization of LSPs of CT3 as function of the number of LSPs requests.

This confirms that BCRA-DSTE accepts more LSP request than CSPF which is due to the important issue of load balancing. The later maximizes the network resource utilization and minimizes the number of requests that would be denied access due to insufficient resource utilization.

Let's now show the importance of our bandwidth preemption algorithm in admission control proceeding. This algorithm brings an important contribution. It is for a full support of DS-TE mechanisms since it supports preemption not only within a CT but also inter CTS. Remember that a TE mechanism, like CSPF, supports only preemption within a same CT. No per class treatment is allowed. Figures 16, 17, 18, and 19 show the main drawbacks of the CSPF approach. The number of blocked/preempted LSPs is varied separately for each CT in these figures. Both figures 16 and 17 show that BCRA-DSTE does not preempt CT0 LSPs requests and CT1 ones. This justifies the increasing values of *u_max* in both figures 7 and 8. Clearly, the *u_max* values given by BCRA-DSTE for LSPs of CT0 and CT1 are increasing with the increasing of CT0 and CT1 LSPs requests. However, in the same conditions, CSPF blocks LSPs without taking into account their CTs levels. As an example, when generating 184 LSPs requests such that the number of CT0 LSPs requests is equal to CT1 LSPs requests, which is also equal to CT2 LSPs requests, CSPF generate 11 blocked/preempted LSPs of CT0, and only 9 LSPs of CT3. In the other hand, BCRA-DSTE accepts all LSPs requests of CT0 by preempting 36 LSPs of CT3 and only one LSP of CT2. This justify the decreasing of CT2 *stdv* values and CT2 *u_max* values in figures 9 and 13 when more than 179 LSPs requests are addressed to BCRA-DSTE approach.

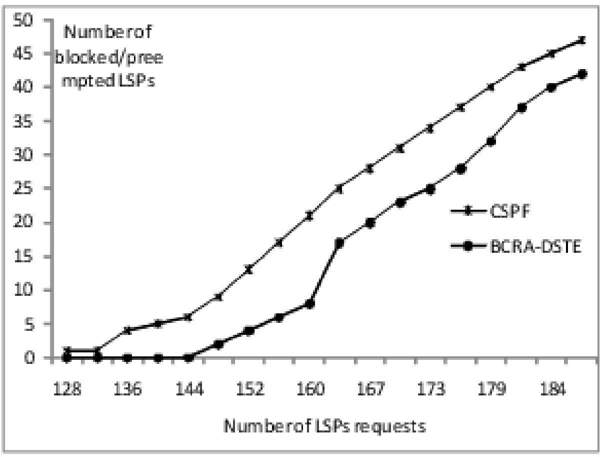

Fig. 15 Number of blocked LSPs as function of the number of LSPs requests.

In conclusion, our admission control mechanism supporting DS-TE functionalities consists of two major functions: route computation and bandwidth management. The route computation algorithm that we have proposed maximizes the efficient use of the network infrastructure.

In fact, simulations results prove that our algorithm performs well in reducing network bottlenecks and distributing network load balancing. The bandwidth management is conducted in two dimensions: CT and priority level. Simulation shows in a first time that CSPF does not provide the preemption fairness among CTs so that CT0 LSPs requests can be rejected while LSPs of weaker CTs remains in the network. Besides, without considering the CT together with the priority, CSPF is not feasible for a full support of DS-TE. In a second time, simulation shows that using preemption across CT let's BCRA-DSTE able to give some degree of immunity to higher priority traffic. However, the danger of preemption aggressivity by rerouting lower precedence LSPs is not ignored. To reduce preemption inter CTs, we have choose to use the Maximal Allocation Model (MAM). Unlike RDM, authors in [9] proved that MAM gives a trade-off between bandwidth sharing to achieve greater efficiency under normal conditions, and to achieve robust class protection/isolation under overload.

Now, an interesting question arises: Does our approach protect the network against rerouting explosion when preemption occurs within CTs?
Bandwidth preemption algorithm, proposed in this paper, is complemented by an LSP preemption algorithm that aims to minimize rerouting by reducing both the number of preempted LSPs and the number of preempted bandwidth. To simplify our experiments, we consider other scenario where only one CT is considered.

- Only one CT is considered: under overload conditions, we are interesting here to compare the performance of our algorithm under overload conditions where only one CT is considered. In the considered scenario, overload occurs with CSPF when generating more than 41 LSPs requests. However, BCRA-DSTE activates preemption mechanisms after the establishment of more than 66 LSPs in the network.

Figure 20 depicts the numerical values of the preempted LSPs given by the different algorithms as function of the number of requested LSPs. It is clear that our algorithm performs better than CSPF in reducing the number of preempted LSPs. Therefore, our algorithm can achieve a kind of safeguard against rerouting explosion.

Figure 21 shows the amount of preempted bandwidth. As we can see, CSPF generates in its solutions more bandwidth to preempt to satisfy high priority LSPs. In the other hand, BCRA-DSTE maximizes efficient bandwidth usage and minimizes LSPs rerouting to gain more network stability.
In summary, simulation study improves not only bandwidth efficiency of our proposed routing algorithm, but also robustness and fairness. Besides, the admission control proceedings based on preemption algorithms achieves significant performance improvement for the well-behaving traffic classes, in term of both bandwidth blocking and LSPs reject. Bandwidth preemption algorithm and LSPs preemption one support greater efficiency in both bandwidth and LSPs protection against QoS degradation under overload conditions.

9. Conclusion

In this paper, we have presented a bandwidth management framework for the support of Diffserv aware MPLS Traffic Engineering. We have essentially developed preemption algorithms for admission control procedur.

Simulations study proved the performance of the route computation algorithm in normal load and overload conditions in terms of reducing network bottlenecks and distributing network load.

To support the high diversity of network applications and to provide available and reliable services to high priority applications, the route computation algorithm is complemented by a rigid admission control mechanism. In fact, our admission control is based on preemption in which the link bandwidth is managed in two dimensions: Class Type and preemption priority. Preemption is classified into two algorithms: Bandwidth preemption algorithm and LSPs preemption one. The bandwidth preemption algorithm is suitable for the Maximum Allocation Model. Our choice of MAM model is justified by the fact that it achieves a robust class protection/isolation under overload conditions.
Simulation study has been carried out to show the performance of the admission control mechanism especially under overload conditions. First, simulation results show that using preemption across CTs and within CTs gives some degree of immunity to higher priority traffic. Moreover, the danger of the preemption aggressivity by rerouting LSPs with lower precedence has been treated elegantly in this paper. In fact, preemption policy combines the three main optimization criteria: number of LSPs to be preempted, the amount of bandwidth to be preempted, and evidently the precedence level.

Our bandwidth management framework performs much better than the standard approach of CSPF in terms of both route computation and admission control policy. It achieves significant performance improvement for the well behaving traffic classes in terms of bandwidth utilization, bandwidth blocking, and preemption availability.

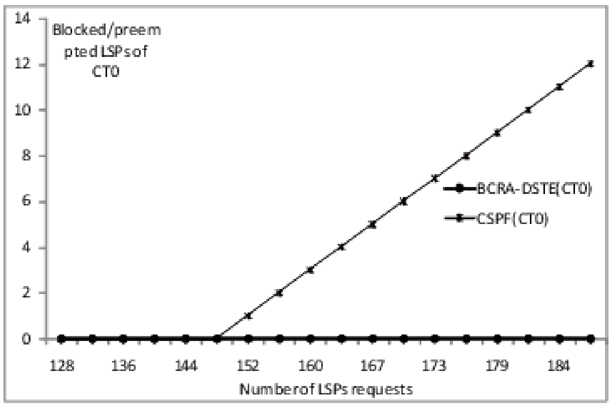

Fig. 16 Number of blocked LSPs of CT0 as function of the number of LSPs requests.

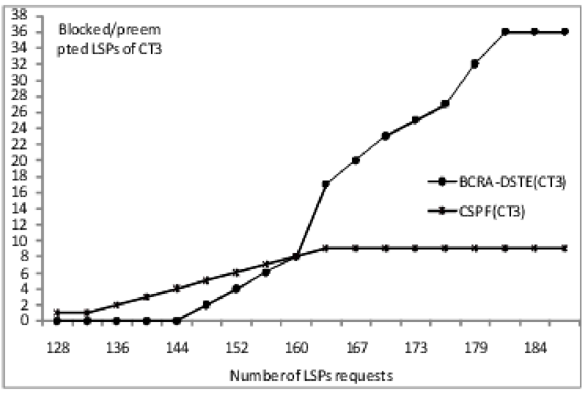

Fig. 19Number of blocked LSPs of CT3 as function of the number of LSPs requests.

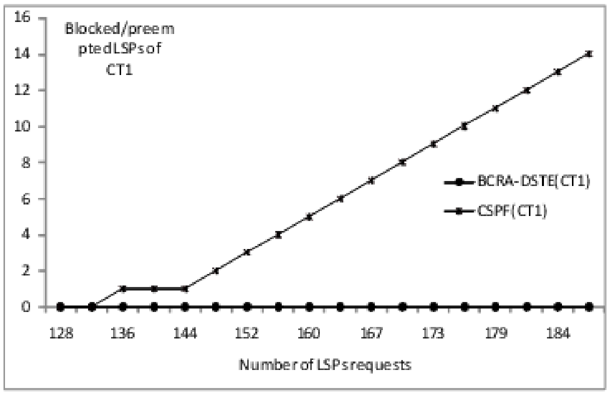

Fig. 17 Number of blocked LSPs of CT1 as function of the number of LSPs requests.

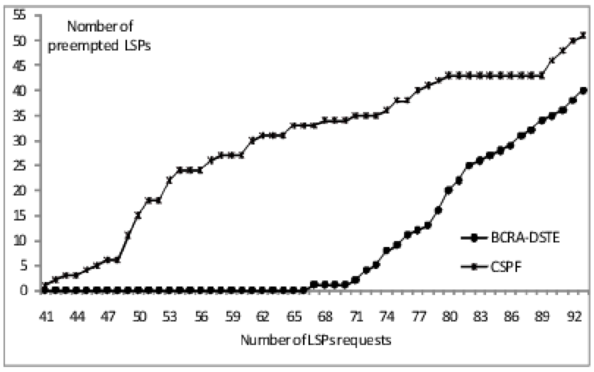

Fig. 20 Number of preempted LSP as function of the number of LSPs requests.

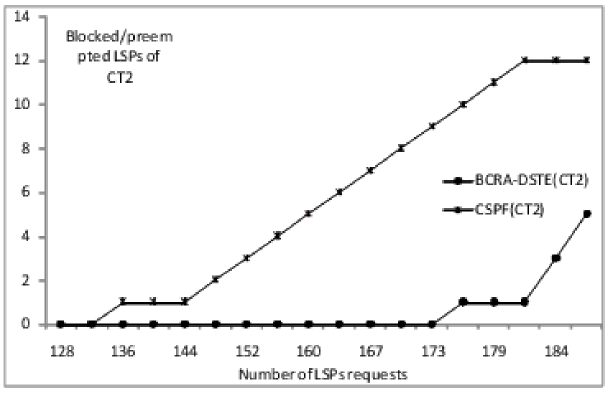

Fig. 18 Number of blocked LSPs of CT2 as function of the number of LSPs requests.

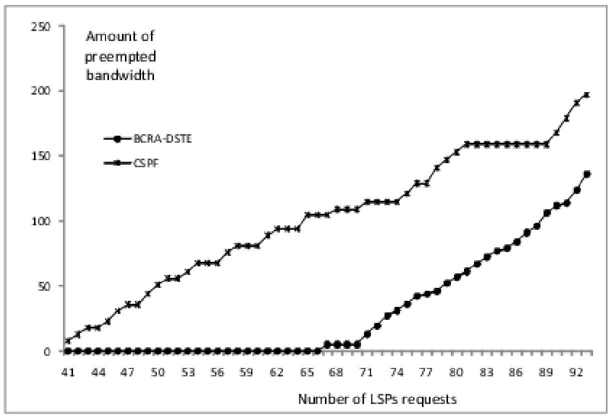

Fig. 21 Amount of preempted bandwidth as function of the number of LSPs requests.

References

[1] Awduche, A. Chiu, A. Elwalid, tains I. Widjaja, and X. Xiao, "Overview and principles of Internet Engineering", RFC 3272, Internet Engineering Task Force, May 2002.

[2] F. Le Faucheur, and W. Lai, "Maximum allocation bandwidth constraints model for diffserv-aware MPLS traffic engineering", Internet Engineering Task Force, 2004.

[3] F. Le Faucheur, and W. Lai, "Russian dolls bandwidth constraints model for diffserv-aware MPLS traffic engineering", Internet Engineering Task Force, 2004.

[4] J. Ash, "Max allocation with reservation bandwidth constraints model for diffserv-aware MPLS traffic engineering and performance comparison", Internet Engineering Task Force, January 2004.

[5] Z. Wang, and J. Crowcroft, "Bandwidth-delay routing algorithm", IEEE GLOBECOM, vol.3, pp.2129—2133, November 1955.

[6] F. Blanchy, L. Melon, and G. Leduc, "An efficient decentralized on-line traffic engineering algorithm for MPLS networks", Proceedings of 18th ITC, vol. 5a, pp. 451—460, September 2003.

[7] M. Kodialam, and T.V. Lakshman,"Minimum interference routing with applications to MPLS traffic engineering", IEEE INFOCOM, vol. 2, pp. 884—893, Mars 2000.

[8] B. Wang, X. Su,and C.L. Philip Chen,"A new Bandwidth Guaranteed Routing Algorithm for MPLS Traffic Engineering", IEEE ICC, vol. 2, pp. 1001--1005 , April 2002.

[9] W.Lai,"Bandwidth Constraints Models for Differentiated Services (Diffserv)-aware MPLS Traffic Engineering: Performance Evaluation", Internet Engineering Task Force , June 2002.

[10] G. Leduc, H. Abrahamsson, S. Balon, S. Bessler, M. D'Arienzo, O. Delcourt, J. Domingo-Pascal, S. Cerav-Erbas, I. Gojmerac, X. Masip, A. Pescaph, B. Quoitin, S.F Romano, E. Salvatori, F. Skivée, H.T. Tran, S. Uhlig, and H. Ûmit, " An open source traffic engineering toolbox", Computer Communications, vol. 29, pp. 593—610, March 2006.

[11] A. Kotti, R. Hamza, and K. Bouleimen, "Bandwidth Constrained Routing Algorithm for MPLS Traffic Engineering", ICNS, June 2007.

[12] A. Kotti, R. Hamza, and K. Bouleimen, "New preemption algorithm supporting differentiated services aware traffic engineering", ICSNC, 2008.

Afef KOTTI was born in Sfax, Tunisia. She received his engineer diploma in Computer Engineering and Master degree in new technologies in computer systems from the Ecole Nationale d'Ingénieurs de Sfax (ENIS) in 2003, and 2004 respectively. She received his Ph.D in information and telecommunication technologies in 2011. He was been an assistant professor at University of sciences of Monastir (FSM) in 2010. His interests include traffic engineering and quality of services in IP/MPLS networks and mobile networks.

Rached Hamza was born in Korba, Tunisia. He received his engineer diploma in Electrical Engineering and Master degree in telecommunication from the Ecole Nationale d'Ingénieurs de Tunis (ENIT) in 1991, and 1994 respectively. He received his Ph.D in telecommunication in 2001. He was been a professor at Ecole Supérieure des Communications de Tunis (Sup'Com) in 2002 and General Manager of Telecommunications Research and Studies Center in 2011. His interests include traffic engineering and quality of services in IP/MPLS networks and mobile networks.