ELSEVIER

Computer Communications 25 (2002) 1076–1084

computer communications

www.elsevier.com/locate/comcom

Minimizing re-routing in MPLS networks with preemption-aware constraint-based routing

Balázs Szviatovszki[a,*], Áron Szentesi[a], Alpár Jüttner[a,b]

[a]*Traffic Analysis and Network Performance Laboratory, Ericsson Research, P.O. Box 107, H-1300 Budapest, Hungary*
[b]*Communication Networks Lab, Eötvös University, Pázmány Péter sétány 1/a, H-1117 Budapest, Hungary*

Received 1 August 2001; revised 30 October 2001; accepted 28 November 2001

Abstract

In this paper, we study the effect of distributed constrained shortest path first (CSPF)-based path selection on the dynamics of label switched path (LSP) preemption. We propose new CSPF algorithms for *minimizing preemption of lower priority LSPs* without requiring any enhancements to the recently proposed link-state parameters. The difference between priority-based path selection methods and previously proposed CSPF methods lies in the way the selection is done among equal cost shortest paths. Our priority-aware CSPF algorithms decrease the number of preempted lower priority LSPs while the LSP setup success ratio is basically the same for all methods. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Path selection; Constrained shortest path first; Traffic engineering; Routing; Priority

1. Introduction

The differentiated services and multi-protocol label switching (MPLS) working groups (WG) of IETF are proposing architectural enhancements to the best-effort IP infrastructure that make bandwidth reservation for aggregate flows a reality. The changes introduced to the IP network architecture depart from the well-understood best-effort IP world. Therefore, building stable, optimally provisioned networks is a great challenge. Particularly, computation of explicit paths when label switched paths (LSPs) have bandwidth requirements is a complex issue. There are basically two possibilities for path calculation: local strategies for routing a single LSP versus global path computation for network-wide optimization of resource usage.

The main benefit of a *local constraint-based routing* method is that each label edge router can automatically compute explicit routes for LSPs. Constrained shortest path first (CSPF) routing implements a simple algorithm to find a feasible bandwidth constrained path in a capacitated graph. When doing distributed path selection, there is no information on future LSP arrivals. Therefore, CSPF aims at minimizing resource usage by restricting path selection to a shortest path that satisfies the bandwidth constraints. However, CSPF in itself cannot always yield network-wide optimal paths, since it uses only local information. It is the role of *global path optimization* to achieve network-wide optimal usage of resources.

In MPLS—as we will discuss in more detail in Section 2—LSPs, beside having a bandwidth requirement, have also setup and holding priorities. All MPLS protocol components support bandwidth reservation on different priority levels. Between these levels LSP preemption is supported. If at path setup, there is not enough free bandwidth available on a link, lower priority LSPs will be preempted [1]. MPLS specifies eight different priority levels; therefore, it can happen that by taking another path a preempted LSP again preempts lower priority LSPs. Although in current MPLS deployments mostly a single priority level is used, there is significant ongoing work at the traffic engineering WG aiming at the development of efficient priority-aware LSP handling in MPLS networks, in order to be able to support traffic engineering in a differentiated services environment. Thus, LSP-level traffic prioritization and preemption will play an important role in the networks of tomorrow.

To the best knowledge of the authors, there have been little work on the *dynamics of LSP preemption*. A notable exception is Ref. [2], in which Villamizar studies failure scenarios and the effect of independent path re-computations and delayed re-flooding of new resource

* Corresponding author. Tel.: +36-1437-7631; fax: +36-1437-7767.
E-mail addresses: balazs.szviatovszki@eth.ericsson.se (B. Szviatovszki), aron.szentesi@eth.ericsson.se (A. Szentesi), alpar.juttner@eth.ericsson.se (A. Jüttner).

PII: S0140-3664(02)00025-7

reservations on overall network performance. This work suggests timing and ordering strategies for re-routing and re-optimization of LSPs after failures, but does not deal with path computation methods.

There are global optimization algorithms that consider the issue of preemption. Garay and Gopal in Ref. [3] studies preemption in asynchronous transfer mode (ATM) networks. Their objective is to find the minimum number of calls that should be removed from the network in order to have enough capacity to accept a new call. This optimization method can only be implemented in a central place, since detailed knowledge is needed about the path of each and every call in the network. Similarly, Mitra and Ramakrishnan [4] provide multi-commodity flow solutions for the global LSP optimization problem which can be used for multi-priority traffic as well.

As we will discuss in more detail in Section 3, all proposed CSPF algorithms operate in a simple way regarding LSP priorities. When computing the path for an LSP, reservations of lower priority LSPs are not taken into account. Known CSPF methods try to maximize the success probability of future connection establishments, thus prepare for future connection arrivals.

In this piece of work, we study the effect of CSPF-based path selection methods on the dynamics of LSP preemption in MPLS networks. As the main contribution of this paper, in Section 4 we propose new CSPF algorithms that take into account the present state of the network by considering the available resource reservation information of lower priority LSPs as well. By doing this, our algorithms aim at *minimizing preemption of lower priority LSPs* and thus, enhance the stability of multi-priority MPLS networks. In Section 4 we first show that, in order to have some idea of affected lower priority LSPs, useful measures can be deduced from the currently flooded link-state information of interior routing protocols (IGPs). Based on these measures, we propose two methods to select such a shortest path on which the probability of preempting lower priority traffic is the lowest.

In Section 5 we compare the preemption performance of the CPSF algorithms proposed in this paper to the most important methods discussed in the literature. To help the understanding of LSP preemption and thus, the dynamics of multi-priority MPLS networks, we first show results that are general for all CSPF methods. Then we study in detail, how the performance of CSPF algorithms differs, in terms of preemption. Presented results show that by concentrating on preemption minimization we do not adversely affect the success rate of the CSPF algorithm, however, the probability of preempting lower level LSPs during an LSP establishment is improved. Finally in Section 6 we conclude the paper and propose future research topics.

2. Preemption in MPLS networks

In this section, we give an overview of the bandwidth reservation and preemption related attributes and mechanisms of MPLS and show how these are used by constraint-based routing to select paths for traffic trunks, and by other traffic engineering mechanisms (e.g. preemption), to achieve service differentiation.

2.1. Traffic trunk attributes

Traffic engineering extensions of the resource reservation protocol (RSVP) [5] and the label distribution protocol (LDP) [6] include 'holding' and 'setup' priorities. Setup priority specifies the importance of an LSP establishment, while holding priority specifies how important it is for an established LSP to hold on to its reserved resources. Both priorities have a range of 0 (highest priority) to 7 (lowest priority). An LSP with higher (numerically lower) setup priority can preempt an LSP with lower (numerically higher) holding priority. To avoid continuous preemption, holding priority should never be lower than setup priority.

2.2. Resource related attributes

IGP extensions of the open shortest path first (OSPF) [7] and the intermediate system to intermediate system (ISIS) [8] routing protocols propose to flood the *maximum bandwidth* (B_{MAX})and the *maximum reservable bandwidth* ($B_{\max}$) of a resource. The latter may differ from the actual maximum bandwidth, because administrators may choose to dedicate only part of the link's bandwidth to traffic engineering, or, to exploit statistical multiplexing gain the reservable bandwidth may exceed the actual bandwidth of the link.

The most important IGP extension to enable preemption is the one in which actual resource reservations are distributed. In Refs. [7,8] *unreserved bandwidth* ($\mathbf{B}_{\mathrm{u}} = (B_{\mathrm{u}0}, B_{\mathrm{u}1}, \ldots, B_{\mathrm{u}7})$) is specified as the amount of bandwidth not yet reserved at each of the eight priority levels on a specific link. This is counted in an accumulative way i.e. if a highest priority LSP is established, all elements of this vector will decrease.

All flooded resource information is stored in a traffic engineering database (TE-DB) of label edge routers. The TE-DB is in turn used by constraint-based routing to select paths.

2.3. Constraint-based routing

As we have mentioned in Section 1, the aim of CSPF in MPLS networks is to automate the path selection for LSPs. To achieve this, operators should configure resource and traffic trunk attributes and deploy a path selection algorithm that matches these attributes against each other. In MPLS the simplest bandwidth constrained CSPF works as follows.

Considering an LSP with setup priority s and bandwidth requirement B_{LSP},

1. in the router's TE-DB mark all links as 'invalid' where

the link's unreserved bandwidth at the priority level of the LSP's setup priority is less than the LSP's bandwidth requirement ($B_{us} < B_{LSP}$),
2. run Dijkstra's shortest path algorithm on the graph composed of the links not marked as invalid. The resulting path (if there is any) will be the LSP's path.

The above algorithm when looking at the B_{us} level, treats lower level reservations as if they were not present, therefore, when the LSP's PATH message traverses the explicit path, preemption decision and admission control is needed at each hop. The former determines which lower priority LSPs will be preempted. The later is essential, since in case of inaccurate link-state information, it can happen that there is not enough unreserved bandwidth at the LSP's priority level.

2.4. Admission control and preemption decision

The admission control function at each router checks whether there is enough unreserved bandwidth to support the setup of the new LSP. In case there is enough totally unreserved bandwidth at an interface, no preemption is needed, so admission is allowed. If this is not the case, the setup priority in the PATH message is taken and a check is made to see if established lower priority LSPs can be preempted. If $B_{us} < B_{LSP}$ there are not enough preemptable LSPs so the PATH message is refused. Otherwise, as many lower holding priority LSPs are preempted as needed.

The selection of the LSPs to be preempted is a local matter. Local preemption strategies for two priorities have been studied by Peyravian and Kshemkalyani in Ref. [9]. We use in the rest of this paper a straightforward extension of their *Min_Conn* algorithm to support the eight priority levels of MPLS. In this strategy until there are LSPs on the $(n + 1)$th priority level, LSPs cannot be preempted from the nth level.

An LSP may preempt more LSPs when moving downstream. For the preempted LSPs, a notification is sent upstream. Preemption does not differ from any other kind of failure, so the head-end tries to re-route the LSP. Preempted LSPs when re-established may further preempt other ones, so a preemption chain may start.

3. Previous work on bandwidth constrained path computation methods

In this section we give an overview of bandwidth constrained path selection methods proposed in the literature. Since most of the proposed methods can be realized as an extension to Dijkstra's well-known shortest path algorithm, we use the terminology of Ref. [10] for describing differences between presented algorithms. To make path selection bandwidth constrained, all methods start by pruning non conforming links based on the bandwidth check $B_{us} < B_{LSP}$. In Ref. [10] this operation is described with an *acceptor function*.

The Dijkstra algorithm aims at finding the best candidate path to a node. For this, at each step two paths are compared with the help of a *comparator function* [10]. At this point, the used metrics have an important role. In case of single or mixed metrics, the comparator function simply evaluates which path's metric is smaller. In the original Dijkstra algorithm this means the simple arithmetic comparison of two real numbers (weights or 'distances'), but for more complicated metrics this comparison can be more advanced.

To provide more path optimization possibilities besides a simple shortest path selection, many algorithms use the concept of multiple metrics [11] which can be easily incorporated into Dijkstra's algorithm. Simply, when comparing the metrics of two links or paths, the initial comparison is done according to the first metric (e.g. the configured link cost or hop count as a metric), and in case of equality, the value of the second-level metric breaks the tie (i.e. the available bandwidth in case of the widest shortest path algorithm).

To make the picture complete, for each metric an *accumulation function* [10] defines how the metric is accumulated along a path. When we know the distance to a node and also we know the cost (metric) of the edges starting from that node to the neighbors, we use this function in the Dijkstra algorithm to determine the candidate distances to the neighboring nodes. In an accumulation function, depending on the type of metric, different operations are done. Hop count or cost is an *additive* metric, so simply the link costs should be summed to get the path cost. Bottleneck bandwidth is the minimum of available bandwidth metrics along a path, so this is a *concave* metric.

Guerin et al. [12] propose OSPF extensions to support QoS routing. They propose the *widest-shortest* path algorithm (WSPF) for the bandwidth-constrained routing problem. The main idea of this algorithm is to first prune links without sufficient unreserved bandwidth and then compute a shortest path on the remaining graph. When several shortest paths are available, the preference is for the path whose bottleneck link unreserved bandwidth is maximal. This strategy aims at using minimal amount of network resources and at balancing load.

The *residual bandwidth ratio* method (RB-CSPF) [10] selects the bandwidth constrained shortest path for an LSP such that as the second metric instead of the bottleneck link unreserved bandwidth, it takes into account the bandwidth that is still unreserved after the LSP setup, normalized by the total reservable bandwidth $(B_{us} - B_{LSP})/(B_{max})$. Moreover, instead of treating only a single bottleneck link, storing and using a configurable number of bottleneck links is described in Ref. [10] (e.g. four).

The WSPF and the RB-CSPF methods both aim at balancing load. However, there are many load-based routing algorithms proposed in the past for different networks [13–15] that use longer paths than the topological shortest one.

All such algorithms are common in their basic idea to prepare for future call arrivals.

In Ref. [16] Shaikh, Rexford and Shin propose a load-based additive link-cost metric as a second metric in the Dijkstra algorithm. They propose this to be used after the original link-cost metric which restricts path selection to shortest paths. Their link-cost metric uses discretization, in order to increase the probability of finding equal cost paths.

In Ref. [17] Shaikh, Rexford and Shin propose bandwidth constrained path selection for long-lived IP flows. Their path selection method is a widest-shortest path method. However, pruning of bottleneck links are done after the Dijkstra computation. Therefore, it can happen that among the topological shortest paths none can support as much bandwidth as requested. For this reason, Ref. [17] permits the use of non minimal paths whose hop-count is one more than the shortest path's hop-count.

In Ref. [11] Wang and Crowcroft studied the complexity of the multi-constraint QoS routing problem. They proposed the *shortest-widest* path algorithm, i.e. their first constraint is to find a path with maximum bottleneck bandwidth, and when there are more such widest paths, choose the one which is the shortest.

Ma, Steenkiste and Zhang in Ref. [18] propose the 'shortest-distance' algorithm that can dynamically balance the impact of hop count and path load. In their link cost formula a variable n can be used to tune the algorithm, i.e. by choosing $n = 0$, we get the shortest path, and if $n \rightarrow \infty$, we get the widest path.

Ma and Steenkiste in another work [19] compare path selection algorithms for traffic with bandwidth guarantees. The widest-shortest, the shortest-widest, the shortest-distance and a variant of the 'dynamic-alternative routing' methods were simulated. The experiments showed that algorithms limiting the hop count and, by this, resource consumption—e.g. the widest-shortest path and the dynamic-alternative routing method—provide good results when the network is becoming overloaded. On the other hand, algorithms aiming at balancing the network load—the shortest-widest and the shortest-distance algorithm—perform better in light and medium load situations. Recently, authors of Ref. [14] showed that 'minimum interference routing' outperforms both simple 'minimum-hop' routing and widest-shortest path routing. The price of better results is, however, the complexity of the proposed algorithm compared to simple Dijkstra-based methods.

4. Proposed preemption-aware CSPF methods

In this section we propose preemption-aware CSPF methods that aim at minimizing unnecessary preemption and thus enhancing the stability of MPLS networks. First in Section 4.1 we derive new measures to estimate how much preemption will occur on a path at LSP establishment. Then in Sections 4.2 and 4.3 we show that these measures can be used to construct new metrics and CPSF algorithms, respectively.

4.1. Preemption measures

To have practically relevant results we have to take into account technological limitations present in MPLS networks. Currently proposed IGP extensions [7,8] provide only *summarized information* about the reserved resources. This means that they do not provide per-LSP information on distant links, i.e. neither the number of LSPs nor their bandwidth values are available. This summarized information is sufficient for CSPF route computation, i.e. to tell if a link has the required resources to accommodate a new LSP on a certain priority level. However, it is insufficient for determining how many and how big LSPs will be preempted due to a setup of the new connection. Moreover, when traversing more links, we cannot tell whether e.g. two times 20 Mb/s will be preempted or only once. The latter case occurs when LSPs preempted on the first link automatically free resources on the second link, thus eliminating the need for further preemption.

Although detailed LSP information is not available at path computation, it is still possible to develop *heuristic methods* for minimizing the volume of preemption, based on strictly the currently available standard IGP extensions.

4.1.1. Free bandwidth

In order to implement priority-aware CSPF algorithms in a label edge router, we should identify useful preemption measures. It is easy to see that if we are given a link and an LSP to be established on it (for which $B_{\text{us}} < B_{\text{LSP}}$ holds), the bandwidth that should be preempted on the link will be larger if we have smaller amount of free bandwidth (unreserved bandwidth on the 7th priority level: $B_{\text{free}} = B_{\text{u7}}$). By using free bandwidth as a measure we actually achieve the simplest preemption-aware widest path selection. This allows us to take into account lower priority reservations and to minimize the amount of bandwidth preempted by the new LSP.

4.1.2. Per priority preempted bandwidth

Besides the above discussed measures, we show that per-priority preempted bandwidth also has an important role. Let's define $\mathbf{B}_{\text{p}}$ as the vector of bandwidth values that should

procedure CalcBwPreemptionVector($\mathbf{B}_u, B_{LSP}$)
  $\mathbf{B}_p = 0$
  $B_{u(-1)} = B_{max}$
  for ($i = 7, i \geq 0, i$–)
    if $B_{LSP} \leq B_{ui}$ **return** $\mathbf{B}_p$
    $B_{pi} = min((B_{LSP} - B_{ui}), (B_{u(i-1)} - B_{ui}))$
  end for
  return $\mathbf{B}_p$
end procedure

Fig. 1. Calculating the bandwidth preemption vector.

be preempted at each priority level of a link by the new LSP:

$$\mathbf{B}_p = (B_{p0}, B_{p1}, \ldots, B_{p7}). \quad (1)$$

The above *bandwidth preemption vector* provides us useful information. With its help, we can derive for example which is the *highest affected priority level* on a link, i.e. the smallest such number i for which $B_{pi} \neq 0$. Fig. 1 shows a procedure for estimating the bandwidth preemption vector (considering the preemption strategy defined in Section 2.4). For an explanation of how this procedure works, the reader is referred to Ref. [20].

4.1.3. Example

Suppose we have an LSP with $B_{LSP} = 70$ Mb/s bandwidth requirement and $s = 3$ setup priority. Now let us take three 100 Mb/s links and suppose that from the flooded unreserved bandwidth vectors we have calculated the following bandwidth preemption vectors:

$$\mathbf{B}_p^1 = (0,0,0,0,30,20,0,0), \qquad B_{u7}^1 = 20 \text{ Mb/s}$$

$$\mathbf{B}_p^2 = (0,0,0,0,0,10,30,30), \qquad B_{u7}^2 = 0 \text{ Mb/s}$$

$$\mathbf{B}_p^3 = (0,0,0,0,0,20,20,30), \qquad B_{u7}^3 = 0 \text{ Mb/s}.$$

We show how the discussed measures—free bandwidth and bandwidth preemption vector—can be used, to decide which link is more desirable to be used at path selection.

By looking at the affected priority levels, we can see that on the first link there is bandwidth to be preempted on the lowest four priority levels, while on the second and the third one only the fifth level is affected. It may be important to use such links for path setup on which only lower priority levels are affected. We can also notice from the bandwidth preemption vector that the second link has less affected bandwidth on the fifth priority level than the third one. This makes the second link more desirable than the third one.

However, if we consider free bandwidth, we can notice that on the first link there is 20 Mb/s free bandwidth, while on the other two links, there is no free bandwidth at all. This can make the first link the most desirable.

4.2. Priority-aware CSPF metrics

Based on the above measures ($\mathbf{B}_p$ and B_{free}) we construct preemption minimization metrics that can be used in CSPF algorithms. We define how the above derived preemption measures are used as path metrics in the Dijkstra algorithm with the help of *comparator* and *accumulator functions*.

4.2.1. Maximize free bandwidth

By maximizing *free bandwidth* (B_{sum}) as a link metric, we aim at preempting the fewest possible lower priority LSPs in terms of sum bandwidth and among those paths on which no lower priority LSPs are preempted, we choose the widest one. Since bandwidth is a concave metric we define such an accumulator function in which the bottleneck link's free bandwidth determines the path's metric. The comparator function is defined such that it chooses the path with larger free bandwidth as a better candidate path.

4.2.2. Minimize affected priority levels

When using the bandwidth preemption vector ($\mathbf{B}_p$) as a link metric, we aim at minimizing the affected priority levels, by preempting only as low priority LSPs as possible. We define the comparison operation for this metric as $\mathbf{B}_p^1 < \mathbf{B}_p^2$ iff for their first (from 0) different coordinate with index i, $B_{pi}^1 < B_{pi}^2$.

With this definition the comparator function actually implements two tie-breaking concepts. If two paths have different highest affected priority levels, the choice is for the path on which the affected highest priority level is smaller. But if the affected levels are the same, selection of the candidate path is done by selecting the path which has smaller preempted bandwidth on the highest affected priority level.

In order to minimize the *affected priority levels* ($\mathbf{B}_p$) along the path of the LSP setup, we propose to treat $\mathbf{B}_p$ as a concave metric in the accumulator function (i.e. the larger vector is taken as the path's metric).

When no preemption is needed, we would like to use the widest path. We achieve this by incorporating free bandwidth with negative sign as a last (ninth) element in the bandwidth preemption vector ($-B_{u7}$ is used since preempted bandwidth is to be minimized, while free bandwidth is to be maximized).

4.3. Priority-aware CSPF algorithms

The order of metrics in the comparator function of CSPF algorithms has huge significance. We propose such an ordering for our new metrics that minimizes preemption without adversely affecting the CSPF success ratio and path length of high priority LSPs. To achieve this, in both algorithms we first prune links for which $B_{us} < B_{LSP}$ (s is the setup priority of the LSP for which the path is calculated). On the resulting graph we restrict path selection to shortest paths based on the original OSPF metric. By using the link cost as the first metric in the comparator function of the Dijkstra algorithm, we achieve that the LSPs are always routed on shortest paths irrespective of lower priority traffic. We utilize preemption information only after this, i.e. when selecting a candidate path among otherwise shortest feasible ones.

5. Numerical results

We have conducted numerical investigations in order to show the real improvements resulting from the use of our algorithms. In Section 5.2 we describe the simulation environment, survey the implemented algorithms and

measurements, then in Section 5.3 we introduce our simulation experiments.

5.1. Performance evaluation methodology

Several different approaches are used for the performance evaluation of routing strategies. For example, Plotkin [13] proposes an analytical approach for this purpose. Another method is to use discrete event simulation, assuming a stochastic process for the arrival of demands [15].

In our case, it was hard to find a tractable analytical approach that can be used to obtain practical results. So, we decided to use flow-level simulation for our performance evaluation purposes. In the literature there has not been published any well-established and justified LSP demand arrival model for MPLS, nor was such available from operators. Therefore we have decided to omit the time factor from our simulations. This means that instead of discrete event simulation we evaluated the performance of the algorithms by observing their behavior in a series of 'static' traffic configurations.

5.2. Simulation model

5.2.1. Network and traffic model

In order to have practically relevant results we carried out our simulations on the *cable and wireless* backbone network topology. This real life data network contains 31 backbone nodes and 102 links, resulting in an average node degree of 3.29. The link capacities vary between DS-3 and OC-192, with the majority of link having OC-12 capacity. The exact topology and capacity values of this network (valid as of June 26, 2000) are publicly available in Ref. [22]. We supposed that demands are generated from every node to every other node.

Since it is hard to get traffic statistics from real MPLS networks, we created randomly generated traffic situations. This means that we have loaded the network to a certain extent with randomly placed LSPs having random bandwidth values. LSP bandwidth is uniformly distributed within the interval $(0, B_{\mathrm{MaxLSP}}]$. In our experiments we focus on such situations when B_{MaxLSP} is around 8–10% of the most common OC-12 link capacity. The priority levels of the LSPs were also set randomly with uniform distribution in [0–7].

5.2.2. Simulation setup

In our investigations we were interested how different performance metrics change as network load increases. Therefore, in our simulations, first, we loaded the network to a certain level by generating LSPs randomly between all nodes. After this, we randomly generated several new LSPs, and tried to route them with the selected CSPF method. In case load increased above the required level after a successful LSP setup, we randomly released some LSPs from the network, until we returned to the operation point used in the given simulation experiment. This test has been conducted at different traffic situations, resulting in a series of probability estimates describing the quality of the routing strategy at different points. Numerous measurements have been taken at each point in order to ensure 95% confidence intervals. In fact, in our graphs confidence intervals are very small and would not provide too much additional information, so, for the sake of better visibility, we decided not to show them in the figures.

We characterize a traffic situation by the *total throughput*, i.e. the sum of all established LSPs' bandwidth. We use total throughput as a measure instead of average link utilization on the x-axis, since we believe that carried traffic is more important to operators than link load. The CSPF failure ratio is influenced primarily by the average link load in the network. However, at a given average link load, different CSPF algorithms may have different amount of carried LSP volumes. For example if one algorithm does ineffective path selection at an early stage it results in longer paths for LSPs routed afterwards. This means that at the same average link load less amount of carried LSP volume is achieved. Consequently if we measure the total throughput instead of average link utilization, we may also investigate the effect of higher average link loads—caused by ineffective path selection—on the CSPF failure ratio.

5.2.3. Compared algorithms

We investigated the performance of the proposed priority-aware CSPF algorithms and compared it to the most promising CSPF algorithms surveyed in Section 3. According to Ref. [21] multiple priority levels, and preemption among them can be used to assure that high priority traffic trunks are always routed through relatively favorable paths (i.e. shortest path). This suggested that we should concentrate only on such algorithms that use strictly shortest feasible paths.

Therefore we simulated the following algorithms proposed in the literature: the basic *shortest* path first algorithm as a reference ('random'), the widest-shortest algorithm ('widest') [12], the *residual bandwidth ratio* method ('residual bw') [10] and the *discrete link cost* method [16]. In the plots we name the algorithms based on the used second level metrics shown in brackets. We use the term random for the simple Dijkstra algorithm because without a second level metric it does a random selection among equal cost paths.

Moreover, our two preemption-aware algorithms were implemented: the *maximize free bandwidth method* ('max free bw') and the *minimize affected priority levels method* ('min affected levels'). The latter was implemented with the help of the bandwidth preemption vector measure. We have incorporated the free bandwidth measure as the last element. Therefore, at light loads when no preemption is needed, a widest path selection is done, based on the free bandwidth.

5.2.4. Performance metrics

We compared the behavior of the implemented

algorithms with the help of the following empirical measures:

- *Success ratio*: measure of CSPF path computation effectiveness. This measure provides information about how many path computation attempts failed due to that CSPF could not find a feasible path for the LSP with the required bandwidth.
- *Success ratio per priority*: path setup success ratio differences are measured for the eight priority levels.
- *Preemption ratio*: the probability that during an LSP establishment at least one lower priority LSP is preempted.
- *Distribution of preempted LSPs between the priority levels*: in the nominator we count how many times an LSP with a given priority has been preempted. In the denominator we have the total number of preempted LSPs.
- *Path length per priority*: average path length of LSPs of different priority levels.

5.3. Performance evaluation

In Section 5.3.1 we first show the effects of preemption on the widest-shortest path method, then we present experiments of the priority-aware CSPF algorithm in Section 5.3.2.

5.3.1. General preemption effects

In our first experiments we show plots for the widest method to demonstrate general effects of preemption on the performance of LSPs with different priority levels.

In Fig. 2 we can see the success ratio seen by the eight priority levels (priorities are represented on the z-axis, with '0' representing the highest priority, and '7' the lowest one). We can notice that success ratio (the complement of CSPF failure ratio) starts to decrease at much smaller total throughput levels for the low priority LSPs than for higher ones. The reservation differentiation concept with (i) multiple priority levels, (ii) link pruning and (iii) cumulative calculation of unreserved bandwidth values achieves that

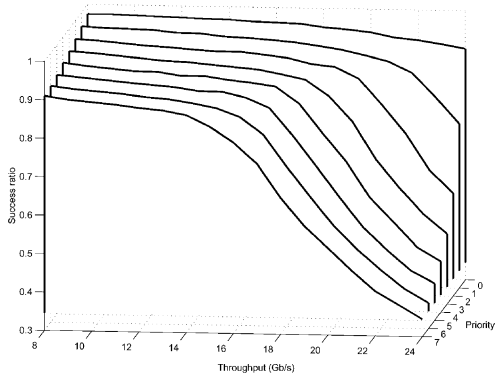

Fig. 2. Success ratio per priority (for the widest method).

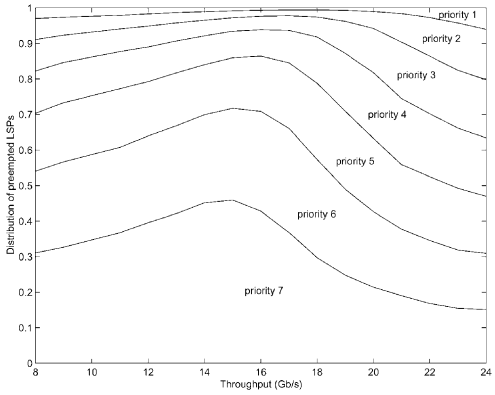

Fig. 3. Distribution of preempted LSPs between the priority levels (for the widest method).

reserved resources of higher level LSPs cannot be used by lowest level LSPs. The effect of this can be seen on the decreased success ratio of lower priority LSPs.

Per priority success ratio provides information about such LSPs that are to be established. However, lower priority LSPs are not only affected by CSPF failure, but also by being preempted by higher level LSPs. To have an idea of which priority levels were preempted, we counted altogether how much LSPs were preempted, and also, how much LSPs were preempted at each priority level. From this we determined the distribution of preempted LSPs between the priority levels, as shown in Fig. 3. Highest priority LSPs can never be preempted, therefore, 'priority 0' is not shown in the figure. LSPs with 'priority 1' are on the next level, thus, these can be only preempted by level 0 LSPs. However, on a link before these relatively high priority LSPs are preempted, the local preemption algorithm discussed in Section 2.4 always tries to preempt lower levels. The probability that among all preempted LSPs a 'level 1' LSPs is preempted increases only when actually almost all lower levels are removed from network links. At this time traffic is dominated by 'level 0' and level 1 LSPs. On the contrary, at low total throughput values, 35–45% of the preempted LSPs are 'priority 7' LSPs.

It is interesting to check the effect of preemption on the path length of LSPs in Fig. 4. In case of constraint-based

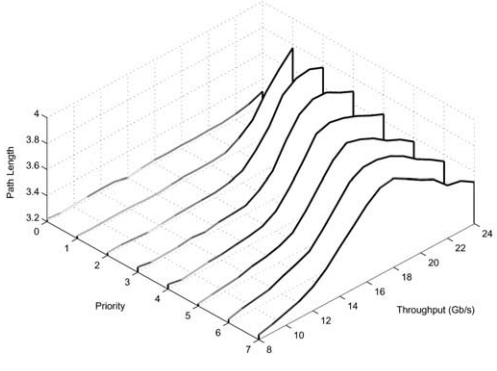

Fig. 4. Path lengths of different priority levels (for the widest method).

routing, path lengths are also influenced by the bandwidth constraints, i.e. the link loads. In Fig. 4 we can observe that at light link loads, for all priority levels the path length is around 3.2 hops. As link load increases first the path length increases, then it starts to decrease. The basic reason for this is the following: in Fig. 2 when for a priority level CSPF failure increases, it means that it is hard to find a feasible path for LSPs. At this stage, most probably the topologically shortest paths do not have enough unreserved bandwidth, so only longer bypass paths can be used. Typically, for all priority levels, when success ratio decreases to 60–80%, the path length increases. However, after a given load level, preemption effects path length, since it is more probable to preempt LSPs established on long paths. At the same time LSPs having their source and destination nodes closer to each other will have a bigger chance to be established successfully. Therefore, after a critical level, path length of established LSPs decreases.

5.3.2. Impact of preemption minimization

In this section we present main preemption performance differences between CSPF algorithms. As we see in Fig. 5, the most important measure, the overall LSP establishment success ratio is roughly the same with all CSPF methods. This means that the probability that the LSP can indeed be established on the computed path is the same with our proposed preemption minimization methods as with e.g. the widest method. This result is not surprising, since previous simulations have also shown [19] that CSPF algorithms (shortest-distance, widest-shortest and dynamic-alternative routing methods) achieved almost the same success rate. If we take into account that in our case all the simulated algorithms restrict path selection to shortest paths, it is not surprising that the success rate differences are small.

The difference between widest and random methods (roughly 3%) are much larger than differences between the proposed preemption minimization methods and previously proposed CSPF methods. The success ratio is lower for the simple random method compared e.g. to the widest method, since the random method may block links at

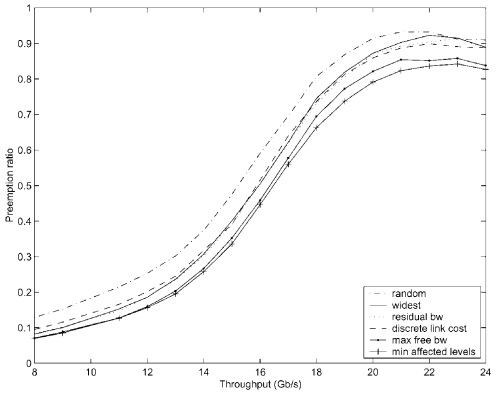

Fig. 6. Preemption ratio.

an early stage, which forces LSPs arriving later to use longer paths. We can observe that our preemption measures, by trying to avoid such links on which preemption is probable, actually balance load similarly to the widest path and other load-based CSPF algorithms.

To quantify the gain in preemption minimization, we used *preemption ratio* as a basic measure. In Fig.6 it is shown that the probability of preemption is significantly lower for our proposed methods. Moreover, we can notice that our strategy to minimize the affected priority levels is more effective than the simpler one that maximizes the bottleneck free bandwidth of the path. We determined from numerical data that at high loads (18–22 Gb/s throughput) the former achieves 10%, while the latter 5% improvement compared e.g. to the widest method. At light loads—which is more important since this is in fact the normal operational range of a typical network—both strategies decrease the preemption ratio by approximately 15%, which is a significant improvement.

When preemption occurred, we measured the average number of affected LSPs (Fig. 7). This includes the directly affected LSPs and also the LSPs preempted by the preempted LSPs that were re-established (chain effect). We found that when preemption occurs, the number of LSPs in the preemption chain does not differ significantly for the different methods. Consequently, we can say that the

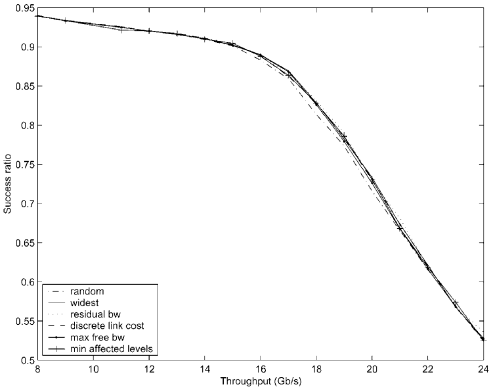

Fig. 5. Success ratio.

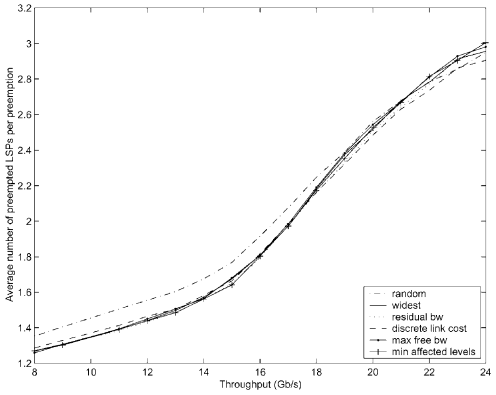

Fig. 7. Average number of preempted LSPs per preemption.

main benefit of using our method can be found in decreasing the probability of preemption, and not in preempting less LSPs when preemption is unavoidable.

6. Conclusions and future work

In networks supporting traffic with diverse QoS requirements, setup of low priority traffic should be done in such a way that subsequently arriving higher priority traffic is not effected adversely. A simple way to solve this problem is to allow preemption between the priority levels. In this paper we have investigated the effects of bandwidth constrained path calculation on the preemption process. The cornerstone of our method is to do path selection by taking into account resource utilization of lower priority traffic. In the studied MPLS environment, we have shown that premium quality can be achieved for high quality LSPs, even if we target preemption minimization.

As a basic step for our preemption minimization algorithms, we have specified preemption measures calculated from standard flooded unreserved bandwidth information. One measure estimates the amount of bandwidth that will be affected, while another one quantifies how many levels will be affected by the new LSP's setup. These measures are then used to construct metrics that are directly applicable in a modified shortest path first algorithm. We build on Dijkstra's well-known shortest path first algorithm, because of its speed. Our modifications increase the running time of the original algorithm by a constant factor.

Our simulation experiments demonstrate that the proposed priority-aware path selection algorithms significantly outperform traditional load-balancing CSPF methods in terms of the total number of preempted lower priority LSPs, thus, it results in *less re-routing* in the network. In addition, we have found that this is achieved by retaining *equal path establishment success ratio*. We obtained the best results when, with the help of a second metric, we aimed at minimizing the affected priority levels.

We have not carried out experiments to determine the performance of our algorithms in case of *inaccurate* link-state information. In order to carry out valuable simulations, deeper understanding is needed about LSP setup arrival processes and bandwidth distributions of LSPs, which significantly influences the inaccuracy of reservation information. As part of ongoing work, we would like to carry out experiments by using more realistic traffic models in dynamic (time variant) configurations.

References

[1] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, X. Xiao, A frame-work for Internet traffic engineering, Internet Engineering Task Force, Internet Draft, July 2000, Work in progress.

[2] C. Villamizar, MPLS traffic engineering in a QoS capable network, Slide presentation given at the MPLS'2000 Conference, October 2000, http://www.ail.gmu.edu/MPLS2000.

[3] J.A. Garay, I.S. Gopal, Call preemption in communication networks, IEEE INFOCOM'92, 3 vol. xxvii + 2515, pp. 1043–50, vol. 3.

[4] D. Mitra, K.G. Ramakrishnan, A case study of multiservice, multi-priority traffic engineering design for data networks, IEEE Globecom'99 (1999).

[5] D. Awduche, L. Berger, D.H. Gan, T. Li, G. Swallow, V. Srinivasan, RSVP-TE: extensions to RSVP for LSP tunnels, Internet Engineering Task Force, Internet Draft, August 2000, Work in progress.

[6] B. Jamoussi (Ed.), Constraint-based LSP setup using LDP, Internet Engineering Task Force, Internet Draft, July 2000, Work in progress.

[7] D. Katz, D. Yeung, Traffic engineering extensions to OSPF, Internet Engineering Task Force, Internet Draft, April 2000, Work in progress.

[8] T. Li, H. Smit, IS–IS extensions for traffic engineering, Internet Engineering Task Force, Internet Draft, September 2000, Work in progress.

[9] M. Peyravian, A.D. Kshemkalyani, Decentralised network connection preemption algorithms, Computer Networks and ISDN Systems 30 (11) (1998) 1029–1043.

[10] K. Kompella, D. Awduche, Notes on path computation in constraint-based routing, Internet Engineering Task Force, Internet Draft, September 2000, Work in progress.

[11] Z. Wang, J. Crowcroft, Quality-of-service routing for supporting multimedia applications, IEEE JSAC 14 (7) (1996) 1228–1234.

[12] G. Apostolopoulos, R. Williams, S. Kamat, R. Guerin, A. Orda, A. Przygienda, QoS routing mechanisms and OSPF extensions, Internet Engineering Task Force, Request For Comments (Proposed Standard) 2676, August 1999.

[13] S. Plotkin, Competitive routing of virtual circuits in ATM networks, IEEE JSAC 13 (6) (1995) 1128–1136.

[14] M. Kodialam, T.V. Lakshman, Minimum interference routing with applications to MPLS traffic engineering, INFOCOM 2000, Tel Aviv, Israel, March 2000.

[15] Á. Magi, Á. Szentesi, B. Szviatovszki, Analysis of link cost functions for PNNI routing, Computer Networks 34 (1) (2000) 181–197.

[16] A. Shaikh, J. Rexford, K.G. Shin, Evaluating the impact of stale link state on quality-of-service routing, IEEE/ACM Transactions on Networking 9 (2) (2001) 162–176.

[17] A. Shaikh, J. Rexford, K.G. Shin, Load-sensitive routing of long-lived IP flows, ACM SIGCOMM (1999) 215–226.

[18] Q. Ma, P. Steenkiste, H. Zhang, Routing high-bandwidth traffic in max–min fair share networks, ACM SIGCOMM (1996) 206–217.

[19] Q. Ma, P. Steenkiste, On path selection for traffic with bandwidth guarantees, IEEE International Conference on Network Protocols, October 1997, pp. 191–202.

[20] B. Szviatovszki, Á. Szentesi, A. Jüttner, Minimizing re-routing in MPLS networks with preemption-aware constraint-based routing 2001, International Symposium of Performance Evaluation of Computer and Telecommunication Systems, July, 2001, pp. 249–261.

[21] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, Requirements for traffic engineering over MPLS, Internet Engineering Task Force, Request For Comments (Proposed Standard) 2702, September 1999.

[22] Russ Haynal's ISP Page, http://navigators.com/isp.html.