

Performance Evaluation of a Distributed Scheme for Protection against Single and Double Faults for MPLS

Fabio Ricciato, Marco Listanti, Angelo Belmonte, Daniele Perla

INFOCOM dept. – University of Rome “La Sapienza”
{fabio, listanti}@infocom.uniroma1.it

Abstract. MPLS can be used to provide network robustness to faults through path protection techniques. In this paper we present a dynamic model supporting different classes of end-to-end protection, including protection against Single Fault and Dual Fault, with and without sharing of backup bandwidth. Beyond link and node failures we also consider protection against Shared Risk Link Group (SLRG) failure. An interesting feature of the proposed scheme is the ability to offer service differentiation with respect to the recovery probability, by coupling the differentiation on the number of backup paths with bandwidth assignment policy. In this paper we describe the underlying algorithms for route selection and backup bandwidth sharing. The route selection is based on explicit load-dependent routing of service and backup paths. We show by simulation that the proposed route selection algorithm is effective in improving the network utilization. We discuss two alternative implementations of our model: distributed and partially centralized. The primary concern with the distributed approach is the message overhead implied by link-load dissemination, e.g. by flooding. However we show by simulation that message overhead can be taken under control by adopting a well-tuned adaptive overhead reduction algorithm. Our conclusion is that both distributed and partially-centralized implementation are feasible.

1 Introduction

Telecommunication networks are facing today an evolutionary trend towards a dynamic paradigm: more and more network functionalities are being shifted from the management plane - that implicitly foresees a sensible human intervention - towards the control plane - with a minimal human intervention. At the same time, the migration of new critical application on IP networks is changing the service paradigm associated to such networks. From delivering simple best-effort service, IP networks are challenged today to offer bandwidth and reliability guarantees, along with service differentiation. Given the dimensions of current networks in size and bandwidth, each network operation model aimed at answering this challenge should be scalable first, then effective in resource usage. In this framework, MPLS and the related dynamic protocols constitute a useful add-on to traditional IP platforms. MPLS is currently being implemented in several IP networks, and it can be exploited to implement connection-oriented traffic engineering techniques.

In the framework of this evolutionary trends, this paper proposes a scheme to achieve service robustness to faults by means of end-to-end path-based protection schemes. Beyond traditional single link / single node failures, we also consider protection against failure of Shared Risk Link Group (SRLG, [1]). Additionally, we consider the possibility that some network links are not prone to failure at all at the MPLS layer. We denote them as No-Risk Links (NRL): they can model for example a link protected at lower layer (e.g. a Sonet/SDH ring). The service and the backup LSPs associated to a generic demand can share a same NRL. The adopted route selection algorithm is based on the joint computation of service and backup paths that are *opportunistically* disjoint, i.e. they can not share a same link or SRLG, but can share a same NRL. We further extend our model to deliver service survivability guarantees under Dual-Fault. The basic idea is to associate to a service LSP *two* disjoint backup LSPs, rather than just one. Moreover, our model supports sharing of backup bandwidth.

The detailed description of the underlying algorithms (e.g. for disjoint-routes selection, for bandwidth sharing) can be found in [4]. Here we present the overall model and briefly revise the underlying algorithms, and focus on performances and applicability aspects. We discuss several issues regarding the applicability of this scheme to packet network. In particular we present two alternative implementations of our model: distributed and partially-centralized. The primary concern with the distributed approach is the message overhead implied by link-load dissemination, e.g. by flooding. However we show by simulations that the message overhead can be taken under control by adopting a well-tuned overhead reduction algorithm based on adaptive thresholds. Our conclusion is that both distributed and partially-centralized implementation are feasible.

The rest of the paper is organized as follows. First in section 2 we provide a global description of the proposed protection scheme. In section 3 we discuss the opportunity of load-aware routing also for protected demands, supported by results from simulations. In section 4 we present the distributed and the partially-centralized alternatives for implementation. In section 5 we present simulation results about the impact of an overhead reduction mechanism that can be exploited in the distributed approach. Finally, in section 6 we conclude.

2 An Architecture of Differentiated End-to-End Protection

In the proposed scheme each connection request can be associated to three different *protection types*: Unprotected (UP), Single-Fault Protected (SFP), Dual-Faults Protected (DFP). For UP demands only a service circuit P_s is established, and no service continuity is guaranteed after the occurrence of a fault along P_s . For SFP demands a service circuit P_s plus a single backup path P_r are allocated: upon failure of P_s the traffic can be readily switched on P_r by the ingress edge node. In case of DFP demands, a service circuit P_s plus *two* backup paths are allocated: a *primary* backup P_{r1} and a *secondary* one P_{r2} . Upon failure of P_s the traffic is switched on P_{r1} , and in case of contemporary interruption of P_s and P_{r1} it is switched to P_{r2} . Note that for DFP the order of preference between the two backup paths is fixed *a priori*.

Both SFP and DFP schemes can be implemented as Dedicated or Shared Protection (following the terminology in [2]) resulting in a total of five *protection classes*. In our model both Dedicated and Shared alternatives are supported for SFP and DFP, resulting in four different protection classes in addition to the basic unprotected one, as summarized in Table 1.

The amount of reserved backup resources on each link and the reaction procedures to the faults will be designed in order to provide the following resilience guarantees with respect to the number φ of contemporary faults in place in the network:

- In case of one single fault ($\varphi=1$) all the affected SFP and DFP demands are guaranteed service continuity over the corresponding *primary* backup circuit.
- In case of two faults ($\varphi=2$) all the DFP demands experiencing interruption of the service circuit P_s are guaranteed service continuity over the primary P_{r1} or secondary P_{r2} backup circuit, depending on the integrity of P_{r1} . There are no guarantees of successful recovery for *all* the SFP traffic, but the network procedures are designed to attempt to recover SFP traffic to the possible extent, in a sort of “best-effort recovery” fashion.
- In case of three or more contemporary faults ($\varphi>2$) no demand is *guaranteed* service continuity, but all are recovered in a “best-effort” fashion. Nevertheless, DFP demands are more likely to be successfully recovered than SFP ones because they have more alternative paths (3 vs. 2) and prioritized access to backup bandwidth in case of contention (see section 2.3). This preserves a certain degree of service differentiation under multiple contemporary faults.

Table 1. The five considered Protection Classes

H	Acronym	Protection Class	number of End-to-end LSPs	backup bandwidth
0	UP	Unprotected	1 (service path)	n.a.
1	Sh-SFP	Shared Single-Fault P.	2 (1 service + 1 backup)	shared
2	De-SFP	Dedicated Single-Fault P.		dedicated
3	Sh-DFP	Shared Dual-Fault P.	3 (1 service + 2 backup)	shared
4	De-DFP	Dedicated Dual-Fault P.		dedicated

2.1 Functional Description

Hereafter we present a scheme for dynamic on-demand allocation of protected connections. Such scheme can be implemented either in a *distributed* or *partially-centralized* fashion. In both cases we have a decisional entity, the Route Selection Engine (RSE), which is in charge of computing the routes for the circuits (service and backup) associated to each new demand. The RSE has an associated Network-State Database (NSD), collecting information about the current network state. For each network link m , the NSD includes information about its capacity u_m and the totally reserved bandwidth b_m^{tot} . In the *partially-centralized* model there is only a single RSE entity processing all the connection requests (hereafter called “demands”) arriving to

the network, while in the *distributed* approach each edge nodes has an associated RSE processing the demands arriving to it.

In both cases the edge nodes are in charge of the actual handling of such circuits (included their setup and release), as well as of the reaction procedures to the faults (i.e., traffic switching onto backup paths). Therefore, in the partially-centralized case some signaling is needed for the central RSE to communicate the selected routes for the new demands to the edge nodes. Also, in both cases the management of backup bandwidth sharing, i.e. the determination of backup reserved bandwidth and handling of bandwidth contention policies, is located at the intermediate nodes. In section 4 we discuss several aspects related to the two approaches, in particular regarding the updating of the NSD(s) associated to the RSE(s).

Each request i has the following associated attributes: ingress node S_i , egress node O_i , requested bandwidth B_i and protection class H_i . It can be initiated directly by the customer or by the network operator. In the following we assume that each demand i arrives to a RSE (centralized or local to S_i). Five different values of H are used to discriminate between the five different protection classes reported in Table 1. For each incoming request, the RSE computes a number of paths towards between S_i and O_i based on the current representation of the network state enclosed in the NSD. The number of computed paths depends on the requested protection class: one, two or three respectively for UP, SFP and DFP demands. For protected demands the service and backup paths must be *fault-disjoint*, i.e. they can not share a same link nor a same SRLG. The algorithm used for finding a minimum-cost set of fault-disjoint path can be found in [4]. Basically, the algorithm applies elementary graph transformations on each SRLG and NRL, then run classical disjoint-paths algorithms (e.g. Suurballe [5]) on the transformed graph. The same approach was independently proposed by other authors [6] limited to the case of single-fault protection. Additionally, in our model a particular graph transformation is used to handle the case of “No-Risk Links” (NRL). A NRL is a link which is not prone to any failure at all at MPLS level. This is for example the case of a transmission link that is protected at lower layer than IP/MPLS, typically a Sonet/SDH ring with Automatic Protection Switching. The fault-disjoint condition does not avoid that the service and backup paths of a single demand share the same link if it is not prone to be interrupted. Therefore the presence of NRLs in the network gives more flexibility in the route selection process.

Here follows a high-level description of the algorithm used by the RSE module to select the set of paths for the new request, further details can be found in [4]:

- **Step 1.** From the Network State Database produce a weighted graph G_1 by associating to each link m a weight w_m which is inversely proportional to the link load (see below). Links with no enough spare bandwidth are eliminated from the graph.
- **Step 2.** From the graph G_1 apply some basic transformations to each SRLG and RL, and produce the auxiliary virtual graph G_2 . Such transformations are designed so that link-disjoint paths in G_2 correspond to fault-disjoint path in G_1 .
- **Step 3.** Find the best set of link-disjoint routes (two for SFP, three for DFP) on the auxiliary virtual G_2 , using the Suurballe algorithm - for a pair of disjoint paths - or its extension - for a triple.
- **Step 4.** Find the correspondent routes on the real graph G_1 . Such routes are fault-disjoint. Order the paths with respect to ascending hop-length and assign the first, second and third respectively to P_s , P_{r1} and P_{r2} .

The path selection algorithm is designed in order to *minimize* and at the same time *balance* the overall resources usage. This is achieved by associating to each link a weight inversely proportional to the current spare bandwidth. As the route selection algorithms pick minimum-cost set of paths, the less loaded links are preferred in the selection of the new paths. This helps in avoiding saturation points (i.e. links exhausting their bandwidth) which could be critical for the allocation of future demands. The exact form of the link-weight function used by the RSE is:

$$w_m = \frac{1}{u_m - b_m^{tot} - B_i} + \varepsilon \quad (1)$$

wherein b_m^{tot} is the current value of reserved bandwidth on link m stored in the local TD, u_m the link capacity and B_i is the requested bandwidth by the new demand i . The term $\varepsilon \ll 1/u_m$ is a small value extracted randomly for each link at each run of the RSE. Its role is to scramble the path selection between the existing equal cost shortest paths for successive demands between the same node pair when the values of b_m^{tot} in the databases are null. This occurs at the very beginning of network operation, when all the links are unloaded, and during the whole network operation in lack of link-load dissemination. Such scrambling improves the level of backup bandwidth sharing. In fact, the potential bandwidth sharing is null when the active paths between a given node pair follow the same route, and increases with the spatial diversity of active paths between the same endpoints. Also, in case of inhibited flooding, such small randomization in the path selection is helpful in distributing the network load over the existing number of equal cost shortest paths, therefore achieving a minimum level of load-balancing (“blind balancing”).

If the RSE fails in finding a suitable set of paths the new request is immediately rejected. This can be due to either a poor topological connectivity (i.e. no disjoint paths exist between the given nodes) or to current heavy resource usage. In case of successful path computation the ingress node initiates the setup signaling phase. The signaling setup (e.g. by RSVP-TE) for service and backup circuits can be run in parallel to speed up the overall procedure. During the setup phase each intermediate node along the involved paths checks for actual resources availability on the local link, and eventually rejects the setup attempt in case of lack of bandwidth. In case that the setup of one or more of the involved paths fails, the request is rejected.

In case of Shared Protection certain information must be conveyed in the setup messages along the *backup* path(s) in order to allow the intermediate nodes to compute the exact amount of additional backup bandwidth that needs to be reserved to support the new demand. Such information substantially identifies the *service* path, which was computed jointly with the backup one(s). Details about the sharing mechanism will be given below in section 2.3. For instance it can be noted that the RSE does not consider the potential bandwidth sharing in selecting the backup path, but lets the intermediate nodes to evaluate and implement bandwidth sharing independently. This mechanism is precise and robust with respect to routing information uncertainty: in facts the evaluation of the potential bandwidth sharing is done locally by intermediate nodes based on locally collected information. Furthermore, this mechanism fits well in a migration scenario where not all intermediate nodes implement backup bandwidth sharing. In fact, the handling of shared reservations requires additional capabilities to be installed at intermediate nodes (e.g., bandwidth evalua-

tion, see section 2.3), which could be deployed incrementally node-by-node in an operational network.

2.2 Algorithm for Backup Bandwidth Sharing

Each network node n maintains the following state information for each outbound link m attached to it: the amount of resources (bandwidth) currently allocated to service circuits b_m^s , to dedicated backup circuits b_m^d , to shared backup circuits b_m^r , and the total link capacity u_m . The total reserved bandwidth is defined at any time as $b_m^{tot} = b_m^s + b_m^d + b_m^r$. The components b_m^s and b_m^d are updated during the signaling procedures in a very simple way: they are incremented / decremented by B_i at every circuit setup / release for a demand with associated B_i bandwidth. Instead, the determination of the component b_m^r relevant to the shared backup bandwidth requires the maintenance of additional data structures. In this subsection we provide a detailed description of such structures and the related update algorithms run during the signaling procedures. In the following E will denote the number of possible fault events (link, node and SRLG failures) that are considered for the specific network, and M the total number of network links. It is reasonable to expect that E is not much greater than M , or at least in the same order of magnitude.

In order to implement Shared-Single Fault Protection (Sh-SFP), a single data structure must be maintained by node n for each outbound link m : the vector FS_m of size E whose generic component $FS_m[k]$ represents the bandwidth needed on link m to carry all the Sh-SFP traffic rerouted on m after the *single* fault event k . Obviously, $FS_m[k]=0$ if k affects m itself. The vector FS_m is maintained by letting the ingress node include in the signaling messages along the *backup path* the list V_s of the links constituting the associated *service path*. From V_s and from the knowledge of the SRLGs associated to each network link the intermediate node can easily derive the list W_s of the *faults* affecting the service path. During the backup setup phase, the intermediate node will increment by B the components $FS_m[k]$ for each $k \in W_s$. Remarkably, the same information V_s should be advertised also during the circuit release phase, in order to decrement the relevant components of FS_m .

In order to implement Shared Dual-Faults Protection (Sh-DFP) two different data structure must be maintained by node n for each outbound link m : the vector FDI_m and the $E \times E$ matrix $FD2_m$. The vector FDI_m for Sh-DFP has exactly the same meaning as FS_m for Sh-SFP. Additionally, the generic component $FD2_m[k_1, k_2]$ represents the bandwidth needed on m to support the demands whose *service* and *primary backup* paths are interrupted by the fault events k_1 and k_2 respectively. The vector FDI_m is updated in the same way as FS_m , therefore the ingress node has to advertise the list V_s of the *links* constituting the *service path* in the signaling messages along the *primary backup* path. On the other hand, along the *secondary backup* path it will advertise both the lists V_s and V_{r1} , the latter referring to the links constituting the *primary backup* path. Similarly to above, the intermediate node n will derive the fault lists W_s and W_{r1} from the link lists V_s and V_{r1} , and during the setup [release] phase will increment [decrement] by B the component $FD2_m[k_1, k_2]$ for each $k_1 \in W_s$, $k_2 \in W_{r1}$.

At every update of these data structures, the new value of the bandwidth reserved to shared backup circuits b_m^r is computed according to the following **allocation rule**:

$$b_m^r = \max_{\substack{k_1, k_2 \\ k_1 \neq k_2}} \{FS_m[k_1] + FDI_m[k_1] + FDI_m[k_2] + FD2_m[k_1, k_2] + FD2_m[k_2, k_1]\} \quad (2)$$

It can be seen that with the above allocation rule the amount of shared backup resources is slightly overestimated with respect to the minimum amount required to meet the service requirements expressed above. This is because the information included in the data structures introduced above is *partial* and *aggregated*. In fact, FS_m , FDI_m and $FD2_m$ enclose global information about the *per-link* reserved resources, not about the full set of *per-demand* paths. While this is enough for evaluating *exactly* the bandwidth needed to protect all the SFP demands against a single network fault, the lack of complete information fatally leads to an imprecise evaluation of the minimum amount of bandwidth required to protect all the DFP demands against a dual network fault. Consider for example two network links a_1 and a_2 , impacted by faults k_1 and k_2 respectively. Denote by D_i ($i=1,2$) the set of demands whose service path includes link a_i , and by $D_{1,2} = D_1 \cap D_2$ the set of demands whose service path include *both* a_1 and a_2 . Denote by $Size(D)$ the sum of bandwidths associated to the demands in the set D . Consider a third link m : the allocation rule given above will reserve for b_m^r an amount of bandwidth equal to $Size(D_1) + Size(D_2)$, while just $Size(D_{1,2})$ would suffice. Therefore there is a potential over-reservation of $Size(D_{1,2})$ resources, as the demands in the set $D_{1,2}$ are counted twice. In order to eliminate such an inefficiency, the intermediate node n should maintain per-demand information, i.e. the vectors W_s and eventually W_{r1} for all the demands having a shared backup path routed through it. This would increase the amount of state information required at each network node, and add complexity (and time) to the computation of b_m^r . On the other hand we found by simulation that the advantage that can be expected from this refinement is modest. We run simulations in the same scenario described below for Fig. 3, with Sh-DFP demands. After K allocated demands, and for different values of K from 200 to 600 (i.e. just before the first blocked request) we compared the amount of backup bandwidth reserved by our scheme with the minimum amount needed to protect all the Sh-DFP demands from any possible pair of faults, as evaluated by simulating *all* the possible fault pairs and considering the worst-case for each link. The results show that our scheme reserves about 4÷6% more backup bandwidth than the minimum required, corresponding to about +2÷4% more total bandwidth usage. Such modest values mitigate the interest towards further refinements of the sharing mechanism based on complete state information.

It is a fact that under dual contemporary faults the allocation rule given above reserves enough bandwidth to protect *all the DFP demands*, but *not all the DFP and SFP demands*. Therefore, in case of dual faults there is a potential contention on backup reserved bandwidth, as each edge nodes will switch traffic onto the backup LSP without coordination with the other edge nodes. Such contention should be resolved by the intermediate nodes in favor of DFP demands. In a IP/MPLS network a possible way to achieve contention resolution is by means of packet level priority-scheduling: each router interface should support a number of queues served with priority scheduling, and Sh-SFP LSPs should be assigned to a lower priority queue than Sh-DFP ones. Further investigations on that topic are out of the scope of this paper.

The important point here is to show how different protection classes and packet-level bandwidth assignment prioritization can be combined to achieve a sensible degree of service level differentiation.

3 Effectiveness of Load-aware Routing for Protected Demands

The distributed model presented above is based on explicit load-aware routing. In our model a generic demand can be blocked during the route computation (we call it a “immediate blocking”) or during the setup phase (“setup blocking”). The knowledge of network link loads by the RSE is helpful in a twofold sense. First, it diminishes the probability that a route without enough spare bandwidth is selected by the RSE, thus diminishing the incidence of setup blocking. In this sense the gain in resources usage is immediate: the “better” is the link load information available at the RSE – i.e., more accurate and more updated - the lower the total blocking rate. Secondly, the availability of link-load information allows to pursue load distribution policies that are helpful to prevent – or at least to delay - the emergence of saturated link that could block future demands. In this sense the achievable gain in resource usage depends not only on the quality of the link-load information but also on the effectiveness of the decisional allocation algorithms. In [7] it was noted that some algorithms could perform worse than the simple shortest-path criterion, particularly those underscoring the path length in favor of path width (e.g. shortest-widest-path). In our model the route selection relies on a minimum-cost paths with load-dependent link weights (eq. 1). With regards to this aspect, we are interested here in assessing the relative gain in resource usage that can be achieved by our model with respect to a simple load-unaware shortest-path routing.

The assessment of such gain is also important from the perspective of a cost-benefit analysis of link-load dissemination. In fact, the distribution of updated link-load information at the route decision point has an associated cost in terms of message overhead, especially for the distributed implementation. The issues related to link-load dissemination and associated overhead costs will be discussed in section 4. In this section we focus on the assessment of the achievable gain. Several previous works (see e.g. [7]) found that a well designed load-aware routing can outperform the load-independent shortest-path algorithm. However, those works apply to unprotected demands, and the applicability of such result is not straightforward when considering DFP (and SFP) demands. In facts, three fault-disjoint routes have to be selected for each DFP demand by the RSE. In generally meshed networks the disjointedness constraint could be so restrictive that very few (or even just one) alternative solutions exist between the requested node pair, so that the possibility to exploit link-load information to “optimize” the choice is very poor or null at all.

In order to assess the convenience of our load-aware routing algorithm we carried a comparative investigation by means of simulations. We considered the extreme cases of “with” and “without” link-load dissemination (LLD). In the first case an exact and update view of the network link loads (i.e. b_m^{tot}) is available to the RSE. In the second case instead the RSE always consider unloaded links (i.e. $b_m^{tot}=0$), then the link weight is fixed for all links as they all have the same capacity.

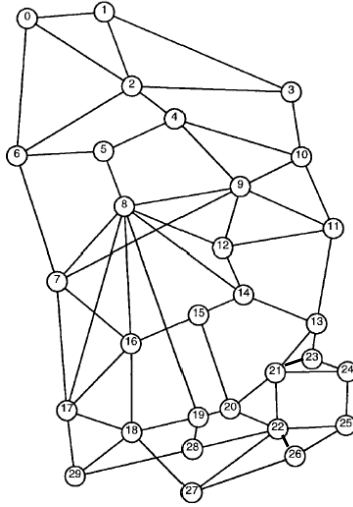


Fig. 1. Network topology used in simulations. It is the same found in [8] with arbitrary addition of two links to make it 3-connected as required for DFP demands.

The simulations were run on the network depicted in Fig. 1. We assumed a simple failure scenario where each bi-directional link is associated to a single elementary fault event affecting both directions at the same time. This choice was driven by the consideration that currently available commercial equipment maintains circuit symmetry in both directions, however this is not a requirement for our model. We fed each network with homogeneous traffic: for each node pair (i,j) requests arrive randomly with mean frequency $\lambda_{ij}=\lambda = 1/N(N-1)$ where N denotes the number of network nodes. In each experiment we considered the homogeneous case where all demands request the same protection class. The link capacity is fixed to 10 Gbps. Each demand is bi-directional and symmetric, and requests a random bandwidth extracted uniformly in [50,150] Mbps. We considered demands with infinite holding time, therefore after a transitory period the network gets to saturation. After the k -th accepted demand, we computed the total bandwidth reserved in the network (i.e. $b(k)=\sum_m b_m^{tot}$ after k allocated demands), and the number $r(k)$ of blocked request until that moment. In Fig. 2-above we plot $b(k)$ versus k for the cases De-DFP and Sh-DFP (Fig. 2-bottom shows the same for the remaining protection classes). Each curve represents the average values over ten different experiments. Let's focus for the moment on Sh-DFP demands. The upper and lower curves refer respectively to the cases with and without link-load dissemination. For each curve, we marked from left to right the points at the 1st, 10th, 50th, 100th, 500th and 1000th blocked demand.

It can be seen that the occurrence of the first blocking occurs after ~ 400 requests without LLD, and ~ 600 with LLD. We also notice that after the first blocking, the slopes of the curves diminishes, indicating that less additional bandwidth-per-demand is reserved for the subsequent arrivals. This is due to the selective filtering of “longer” requests, i.e. those between farther endpoints. In fact, when the network is close to saturation longer requests experience a higher blocking probability, so that residual

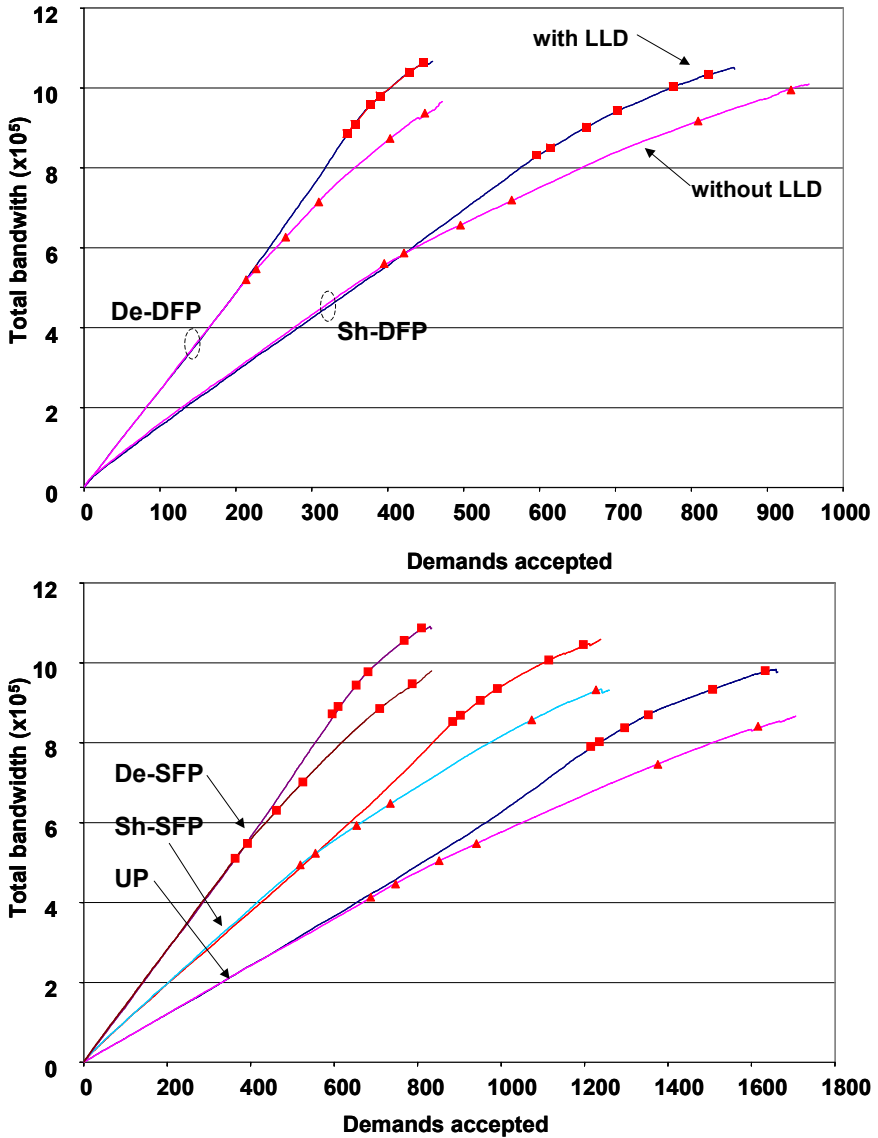


Fig. 2. Total bandwidth usage vs. number of accepted demands for each protection class, with (square spots) and without (triangle spot) Link-Load Dissemination. The spots indicate the 1st – 10th – 50th – 100th – 500th – 1000th blocked demand

shorter request can be admitted with lower bandwidth consumption. Clearly the selective blocking effect is highly undesirable, and the network operation point – in terms of allocated demands - should not exceed the region identified by the first few blocking. Therefore we could take the value of k at the 1st-10th blocking as an estimate of the number of demands that the network can support before the emergence of saturation effects. In other words, such value can be considered as an estimation of the net-

work *net capacity limit*. In Fig. 3 we reported such limit for the cases with and without LLD, for each protection class. It can be seen that the gain in terms of *net capacity limit* is sensible with LLD, between +50% and +70% for protected demands.

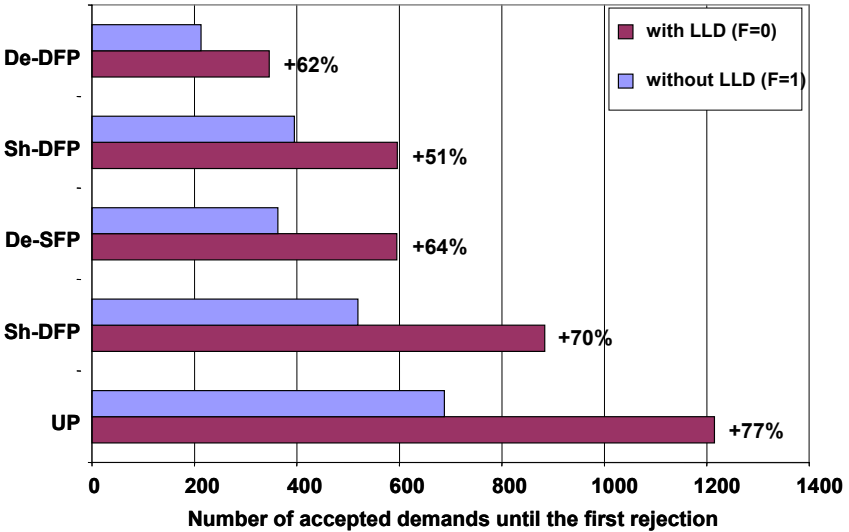


Fig. 3. Comparison of the *net capacity limit* (defined as the number of accepted demands until the 1st blocked demand) for different protection classes, with and without Link-Load Dissemination.

The initial slope of the curves in Fig. 2 represents the average per-demand network bandwidth allocated with the specific protection scheme. The values (in Mbps) are 2558 for De-DFP and 1395 for Sh-DFP, indicating a relative saving of ~45% bandwidth due to sharing. For SFP demands such values decrease to 1465 (De-SFP) and 965 (Sh-SFP), with a saving of ~34%. Finally, for UD demands the per-demand consumed bandwidth is 650, which is consistent with the fact that each demand consumes an average bandwidth of 100 Mbps in each direction, and that the average path length is about three hops.

From such values it can be seen that the relative resources usage normalized to the UP case is ~1.5 for Sh-SFP, ~2.25 for De-SFP, ~2.15 for Sh-DFP and ~3.9 for De-DFP. Such values should be carefully considered in the definition of the billing model for each protection class.

4 Distributed or Partially-Centralized ?

In the previous section we assessed that for each protection class the gain achievable with the proposed load-aware routing model is substantial. In a partially-centralized implementation of such model, an updated view of the network state can be maintained by the centralized route server, by keeping memory of each request arrival /

departure. This is possible because a single entity manages all the network requests. Note that in this case the central RSE has a complete knowledge of network state, and can also compute the amount of backup bandwidth sharing on each link in case of Shared protection. In this approach, that reminds the concept of *Bandwidth Broker* in the Diffserv context, some signaling is needed between the central server and the edge nodes that are in charge of the LSP setup.

On the other hand, in the distributed implementation link-load dissemination is needed to all edge-nodes. Each internal network node must advertise link load changes to the edge nodes hosting the RSEs, and this can be achieved in two ways: *flooding* or *multicasting*. Both approaches have pros and cons. Link-load flooding can be implemented with slight modification to the existing routing protocols, and this feature is already being considered for the traffic-engineering extensions to such protocols (e.g. OSPF-TE [3]). On the other hand, link-load flooding requires implies more processing overhead as *all* network nodes, not just edge ones, have to process flooded messages. On the other hand, multicasting link-load advertisements to edge nodes only requires support for multicast forwarding.

Independently on the dissemination technique, flooding or multicasting, an important overhead metric is the frequency of *link-load advertisements* (LLA), i.e. how many LLA are generated per unit of time. By assuming a mean arrival rate of R requests per second to the network, the mean rate of LLA generation is $G = L \cdot R$, with L denoting the average number of links involved in supporting each demand. For UP demands L roughly approaches the average shortest-path D , as most of the selected routes will not depart too much from the min-hop shortest paths (this is a convenient feature, as observed in [7]). On the other hand, for SFP and DFP demands that involve two and three paths respectively for each demand, the value of L should be close to $2 \cdot D$ and $3 \cdot D$. In case that requests with finite holding time are considered, the value of G must be doubled, as also the release of a circuit triggers a (negative) link-load change. Therefore we end up with a mean LLA frequency of $G \approx 2 R \cdot n \cdot D$ ($n=1,2,3$). The volume of such overhead depends on the network size (factor D) and on the time-scale of demands arrivals (factor R), which in turn is likely to depend on the bandwidth granularity of the request. It is reasonable to expect that a high request rate R is associated to small requested bandwidth B , and *vice versa*. Therefore it is possible that the distributed model presented so far can not scale up to large networks delivering dynamic on-demand service at small bandwidth granularity, unless some mechanism is implemented to control such overhead.

5 Impact of Overhead Reduction Mechanism

In case the network operator is willing to maintain a distributed approach, some *advertisement reduction* mechanism must be implemented to control the LLA overhead. Several such mechanisms were proposed in [9] as *flooding reduction* algorithms. Basically, with such algorithms LLAs are not generated upon each link-load modification (i.e., for each LSP setup / release), but only when certain thresholds are crossed. Typically, hysteresis cycles (based on double thresholds) and/or hold-down timers are used to avoid that fluctuations around a single threshold trigger a large

number of LSAs. Eventually, such algorithm can be *adaptive*, i.e. they include some parameter subject to dynamic adjustment.

With any flooding reduction mechanism the diminution in the overall flooding rate comes at the cost of a degradation in the quality of the link-load information disseminated in the network. In other words, the view of the network state maintained by the edge nodes get coarser, i.e. less precise and less updated. On the other hand such degradation does not necessarily results in a diminution of bandwidth allocation efficiency, as in most cases a coarse link-load information is enough to take “good” routing decisions [7]. Intuitively, in order to take a good routing decision it is often enough to know which links are lightly loaded and which ones are heavily loaded, rather than know the exact load value of each link. Also, the knowledge of the exact value of spare bandwidth is useful for heavily loaded links, much less for low loaded ones.

Based on such considerations, and comforted by some previous results for non-protected paths, we chose a particular algorithm based on double adaptive thresholds, and we analyzed its impact on our distributed SFP / DFP protection schemes. The algorithm was first presented in [9], to which the reader is referred for algorithmic details. Basically, the algorithm is based on two thresholds, t_{high} and t_{low} . For each new reservation, the new value of total bandwidth b^{tot} is computed (for Shared protection this involves the sharing algorithm presented in section 2.2) and compared with the thresholds. A new LLA advertising the new value of b^{tot} is generated only if $b^{tot} < t_{low}$ or $b^{tot} > t_{high}$. As the distance between the thresholds is non-null ($t_{high} > t_{low}$) an hysteresis cycle is introduced. The threshold values are dynamically updated after each reservation as (u denotes the link capacity):

$$\begin{aligned} t_{high} &= b^{tot} + (u - b^{tot}) \cdot F \\ t_{low} &= \max(0, b^{tot} - (u - b^{tot}) \cdot F) \end{aligned} \quad (3)$$

where F is fixed parameter with $0 \leq F \leq 1$. The setting of parameter F governs the trade-off between link-load dissemination overhead and precision. At one extreme $F=1$, and the lower and upper thresholds are initialized to 0 and u (link capacity) respectively, so that no LLA will never be triggered. At the opposite extreme $F=0$, and each new reservation will trigger a new LSA. Therefore $F=0$ and $F=1$ denote respectively the “exact” and “null” link-load dissemination cases introduced above. Simulations were run to evaluate the impact of such overhead reduction algorithm on the performances of our distributed SFP / DFP scheme. We run simulations in the same scenario considered for Figg. 2-3, but with a random holding time exponentially distributed with mean T . We will denote the first set of simulations with infinite holding time as “static simulations”, and this second set as “dynamic”. In the dynamic simulations it is meaningful to determine the mean input traffic intensity as $Q_B = \sum_{ij} \lambda_{ij} T \cdot B$ (in Mbps), with B the mean requested bandwidth ($B = 100$ Mbps in our case), T the mean holding time and λ_{ij} the mean arrival rate between nodes i,j . In the simulations we normalized $\sum_{ij} \lambda_{ij} = 1$, and tuned T so that the mean input traffic was equal to the net capacity limit found in the first set of static simulation for the “without LLD” reported in Fig. 3 (corresponding to setting $F=1$ in the considered flooding reduction scheme). We considered this choice point as a satisfactory network operation point.

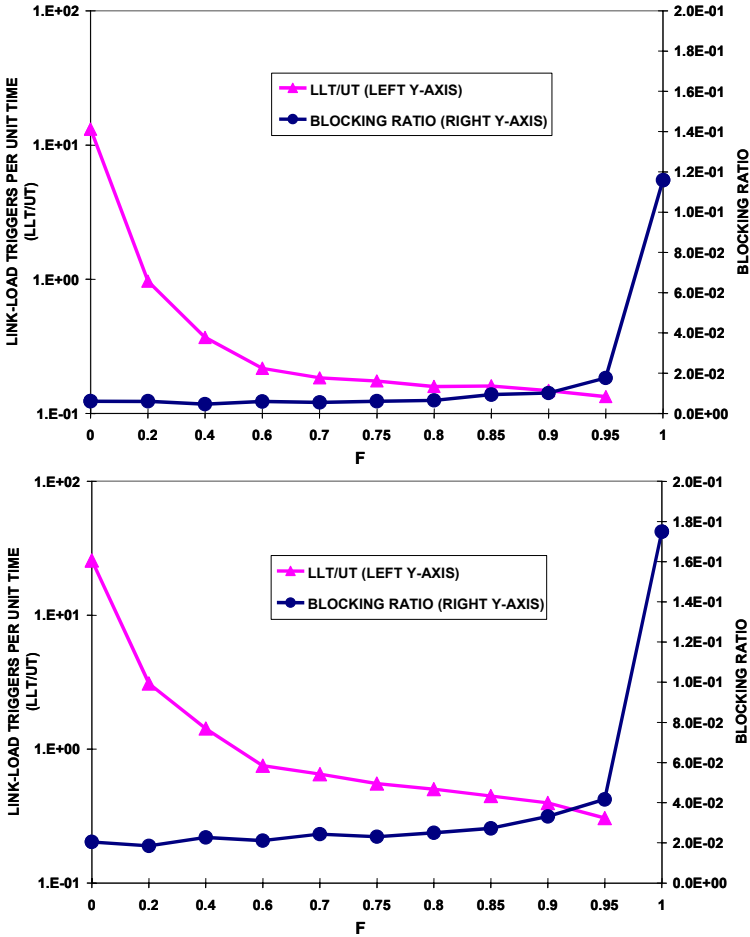


Fig. 4. Call blocking and message overhead vs. parameter F (above: Sh-DFP with $T=600$ time-units; bottom: De-DFP with $T=350$ time-units)

We repeated the simulation for different values of parameter F, and measured i) the experimental blocking ratio and ii) the mean frequency of LLAs. The values of both metrics, averaged over 10 different experiments, are reported in Fig. 4 for a network loaded with DFP requests. Very similar results were found for the remaining protection classes. The graphs show that with the appropriate setting of parameter $F = 0.8$ it is possible to achieve a reduction in LLA frequency of two order of magnitude with a negligible impact on the blocking probability. Notice that with the adopted overhead reduction mechanism the LLA rate has dropped below the request arrival rate. On the other hand, by completely eliminating LLAs (i.e., for $F=1$) the blocking probability increases by a factor of 10. This is a further confirmation about the usefulness of load-aware routing also in case of protection routing schemes.

6 Conclusions

In this work we presented a protection scheme for against single and single fault for IP/MPLS networks. Our scheme is base on load-dependent explicit routing. We showed by simulation that link-load dissemination has a dramatic positive impact on the efficiency of network utilization also in case of protected demands.

Our model can be implemented in a *distributed* or in a *partially-centralized* fashion. In the former case, a major issue is the message overhead implied by link-load dissemination. However, we showed that the message overhead can be taken under control by adopting a smart flooding reduction mechanism. Therefore, the results presented here suggest that both approaches are feasible for practical implementation.

References

- [1] J.Luciani et al., IP over Optical Networks A Framework, Internet Draft, Work in progress, draft-ietf-ipo-framework-03.txt.
- [2] D. Zhou, S. Subramaniam, "Survivability in Optical Networks", IEEE Network, Dec. 2000.
- [3] K. Kompella et al. "OSPF Extensions in Support of Generalized MPLS", <draft-ietf-ccamp-ospf-gmpls-extensions-07.txt>, work in progress.
- [4] F. Ricciato "An Architecture for Dynamic Differentiated End-to-End Protection for Connection-oriented Networks", CoRiTeL report (available at <ftp://ftp.coritel.it/pub/Publications/DynamicDifferentiated.pdf>).
- [5] J. W. Suurballe, R. E. Tarjan. A Quick Method for Finding Shortest Pairs of Disjoint Paths. Networks, 14:325–336, 1984
- [6] G. Ellinas, E. Bouillet, R. Ramamurthy, J.F. Labourdette, S. Chaudhuri, K. Bala "Routing and Restoration Architectures in Mesh Optical Networks", to appear in Optical Networks Magazine.
- [7] A. Shaikh, J. Rexford, K. G. Shin, "Evaluating the Overheads of Source-Directed Quality-of-Service Routing", Proceedings of IEEE International Conference on Network Protocols, October 1998.
- [8] R. R. Irashko, W. D. Grover, M. H. MacGregor, "Optimal Capacity Placement for Path Restoration in STM or ATM Mesh-Survivable Networks", IEEE/ACM Transactions on Networking, June 1998.
- [9] A.Botta, M. Intermite, P.Iovanna, S.Salsano, "Traffic Engineering with OSPF-TE and RSVP-TE: flooding reduction techniques and evaluation of processing cost", CoRiTeL report, available at <ftp://ftp.coritel.it/pub/Publications/thresholds.pdf>.