Experience from implementation of a distributed MPLS-TE control plane in the TANGO testbed

R.Albanese*, A.Bosco+, A.Botta+, P.Iovanna+, A.Liburdi*, M.Listanti*, U.Monaco*, F.Ricciato*
(*) INFO-COM dept. of University "La Sapienza" of Rome, Italy
(+) CoRiTeL, Rome, Italy

***Abstract*— This paper reports on the design and implementation of an experimental testbed of PC/Linux routers with a dynamic control plane based on MPLS-TE. We present some key aspects of the architecture and report on our experience in their prototypical implementation. In particular, we focus on two main topics: multiclass bandwidth allocation and fault protection. Regarding the former, we describe a novel allocation model that is versatile still simple to implement within the existing protocols. Regarding the latter, we detail here the issue of fault notification, achieved through IGP flooding. Finally, we present some preliminary experimental results. We believe that the ideas and experimental insight contained in this work will be helpful to other people involved in standardization and implementation of MPLS-related control plane.**

I. Introduction

The definition and design of an advanced dynamic control plane for future MPLS network has attracted a large interest in the nowadays networking arena. This challenge is addressed by extending and integrating existing signaling and routing protocols into the so called MPLS-TE protocol suite [1] [2]. We are currently involved in the design and implementation of an experimental testbed - on PC/Linux platform - with a dynamic control plane based on MPLS-TE. This activity is part of a broader research project called TANGO [3]. In this paper we present some key aspects of the network architecture and report on our experience in a prototypical implementation. In particular, we focus on two main topics: multi-class bandwidth allocation and fault protection. Regarding the latter, we detail here the issue of fault notification. We believe that the ideas and experimental insight contained in this work will be helpful to other people involved in standardization and implementation of MPLS-related control plane.

The rest of the paper is organized as follows. In section II we briefly describe the overall architecture, which includes an end-to-end protection scheme. In section III we present a novel multi-class bandwidth allocation that is rather versatile still rather simple to implement in the MPLS-TE framework. In section IV we discuss our choice to implement fault notification scheme based on OSPF-TE flooding, along with some implementation details. Finally, in section V we report some early experimental results and in section VI we conclude.

[1]This work was supported by the Italian Ministry for University and Scientific Research through the TANGO Project (PNR 2001-2003, FIRB).

II. Overview of the Architecture

The traffic engineering scheme implemented in the TANGO testbed is fully distributed, without any centralized element. We consider a Diffserv over MPLS domain with on-demand provisioning of several kinds of end-to-end connections. First, each connection is associated to a single Diffserv class (corresponding to the "class-type" in [4]). In our testbed we support EF, AF1x, AF2x and standard best-effort. Second, each connection can be associated to one out of three different *protection classes*: Unprotected (UP), Single-Fault Protected (SFP) and Dual-Fault Protected (DFP). We adopt a end-to-end fault protection scheme, so that one (for SFP) or two (for DFP) disjoint backup LSPs have to be installed for each protected connection along with the working LSP. It is assumed that each connection request is processed by the ingress edge router, which is the only network element responsible for the whole set of associated LSPs. Fig. 1 shows the logical organization of the software modules at the generic edge node. The Node Manager (NM) is in charge of receiving connection requests from the external and coordinate the information flow between the modules. Therefore, for each incoming connection request, the NM will first compute the end-to-end routes of both the working and backup LSPs (arrow a). The route computation is local to the edge node (source routing) and run by a separate module called Route Selection Engine, RSE. The route selection is based (arrow b) on the information about network topology and residual link bandwidth that is disseminated by OSPF-TE through the flooding of Opaque LSAs (arrow c), and maintained at the local Network State Database.

The route selection algorithm running at the RSE is the one described in [5] and [6]. Basically it jointly selects the routes of the working and backup LSP for the new connection taking into account i) the disjointedness constraint, ii) the available bandwidth constraint and iii) a bandwidth minimization / balancing objective. The latter is achieved by simply minimizing a link-cost metric that is inversely proportional to the residual reservable bandwidth for the specific Diffserv class. In fact, this information is carried in a specific element of the Opaque-LSA, as detailed in section III. Additionally, the RSE algorithm supports Shared-Risk Link Groups (SRLG) and SRLG-disjointedness for protected demands. (see [6] for details).

After selected the routes, the NM triggers the RSVP-TE

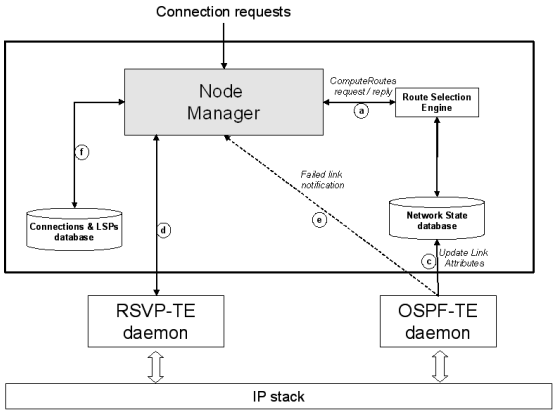

Fig. 1. Structure of logical modules at the edge node.

daemon (arrow d) to start the setup signaling procedures for the working and backup LSPs. During the signaling phase, each node along the path enforces Admission Control in order to check current bandwidth availability. This step is necessary because the link load information available at the edge node may be not synchronized with the current network state. This can be due to the intrinsic delays in the flooding process and / or to the adoption of some conservative link-state update policy aimed at controlling the flooding overhead. Such policies (also called "flooding reduction" policies) restrict the generation of new Opaque LSA upon reaching some critical watermarks and / or on the basis of pacing timers, rather than upon *each* new reservation, and they are essential to control the flooding overhead associated to link load dissemination. Examples of flooding reduction mechanisms can be found in [7] [8] [9]. We adopted the algorithm proposed in [9] (see also [5]), adapted to be applied in a multi-class environment as detailed in section III.

Upon occurrence of link failure, the edge node must be notified the event and promptly switch the incoming packets from the working to the backup LSP. Therefore, a mechanism is needed to convey the failure notification from the internal node to the edge nodes. To date no mechanism has been standardized in MPLS-TE to achieve that. Instead, this issue is being discussed in the IETF [10] in the framework of GMPLS, and two mainstream approaches have been proposed so far: signalling-based and flooding-based fault notification. We preferred the latter, and we implemented in OSPF-TE the flooding of fault notification messages via Opaque LSA. In section IV we discuss the reasons in support of this choice, and report on the related implementation issues.

The development of the TANGO testbed foresees a set of basic functionality that are currently under implementation, and a set of additional capabilities that are included in the architecture but deferred for implementation in a second phase. Among those, we cite the support for setup and holding priorities, and the support for shared protection.

III. The Multi-class Bandwidth Allocation Scheme

In this section we provide some details about our proposal for bandwidth allocation, that is how link capacity is shared among multiple traffic classes. In the following we will index by i the Diffserv classes supported within the domain, included the best-effort. In particular, in the TANGO testbed $i = 1, 2, 3, 4$ will refer to EF, AF1x, AF2x and BE respectively. For each generic link, we will denote by v_i the current *assigned bandwidth* to class i, i.e. the sum of the bandwidth values associated to all the current LSPs of class i. Typically, since no bandwidth reservation is associated to best-effort LSPs, the assigned bandwidth is always zero for BE, nevertheless a counter has been associated to it for sake of compactness of the notation.

Noticeably, we do not distinguish between the bandwidth assigned to working and backup LSPs within one class, therefore both are accounted for in v_i. In fact, the route selection algorithm running at the edge nodes does not distinguish between working and backup bandwidth components, and it simply prefers the links with the largest residual bandwidth.

Additionally we will denote by u_i the *minimum guaranteed bandwidth* for class i. This is a manually configurable parameter offering to the operator the possibility to enforce a minimum guaranteed cushion to class i on the specific link. In other words, even in lack of any request for bandwidth of class i, such bandwidth cushion can not be taken by LSPs of other classes, but remains available for future requests of class i. This is useful to apply bandwidth isolation between classes, which is an important requirement as stated for instance in [11]. Whether or not the provider is willing to apply such minimum bandwidth cushion is a matter of business policy. The default value for u_i is zero, except for best-effort traffic. In fact, it is likely that any provider might want to let some percentage of link capacity available to the public best effort traffic.

From v_i and u_i, the generic node responsible for the link will extract the value of the current *reserved bandwidth* as

$$r_i = \max(v_i, u_i). \tag{1}$$

For each new LSP, the r_i counters are used to decide about its acceptance through the specific link, that is to enforce local Admission Control (AC).

A. The Admission Control algorithm

The AC function must ensure that at any time the reserved bandwidth components meet a set of constraints. These can be defined on the single values of r_i (e.g., "the EF class can not exceed 50% of the link capacity"), or on some partial combinations (e.g., "the AF1x and AF2x classes jointly can not exceed 70% of the link capacity"), or on their complete sum (e.g., "the sum of reserved bandwidth for all the classes can not exceed the link capacity"). Each of such constraints can be dictated by business related policies or by QoS related considerations - for example, the upper limit for the EF class,

whose packets are assumed to be prioritized over the rest of the traffic, and belong to delay-sensitive applications, can be obtained from simple queuing theory so as to bound some statistical delay metric.

In general, the full set of constraints can be written in a formal way as follows:

$$\mathbf{r}'\mathbf{L} \leq \mathbf{c} \tag{2}$$

wherein $\mathbf{r}$ is the column vector $\mathbf{r} = \{r_1, r_2, ..\}$ collecting the single values of reserved bandwidth for each class. The matrix $\mathbf{L}$ is a matrix of binary elements, each row represents a single constraint, and $\mathbf{c}$ is the column vector of associated limits. Similarly to $\mathbf{r}$, we will denote by $\mathbf{v}$ and $\mathbf{u}$ the vectors collecting the v_i and u_i components.

With the above positions, the local AC algorithm can be described in a very simple way. During the signaling setup phase for a new LSP of class j and bandwidth b, the local node will compute the tentative new value $v_j^* = v_j + b$ ("update rule") and since the new value of vector $\mathbf{r}^*$. Therefore, he will check whether constraint (2) holds for the new tentative vector $\mathbf{r}^*$. In the affirmative case the request can be accepted and the new value of v_j recorded. Conversely, the counters must be updated also when a LSP is removed from the link. In the case that backup bandwidth sharing is NOT applied, the simple update rule given above is applied to both working and backup LSPs. On the other hand, in case of bandwidth sharing, the update rule for v_j^* for the setup of a backup LSP must be revised accordingly to the algorithm detailed in [6] [1].

B. Advertising the residual bandwidth

When a LSP is installed / removed from the link, the local node should advertise through OSPF-TE flooding the new state of the link, in terms of unreserved bandwidth, in order to let the edge nodes to update their network-state database and make their route selection processes to be coherent with the current network state. More formally, if r is the currently reserved bandwidth and c the maximum reservable bandwidth (not necessarily the link capacity), the value of $g = c - r$ is advertised, where g represents the *residual reservable bandwidth*, that is the amount of additional bandwidth that can be assigned to future requests. Moreover, the new value of g is not advertised upon *each* LSP installation / removal, as this would cause a large amount of flooding overhead. Rather, the process of Opaque LSA generation is done according to local update policies embedding watermark-based algorithms and / or hold-down timers [7] [8] [9]. As a consequence, it follows the process of variation of g less accurately, but with much less flooding overhead. In our model we extend this approach to a multi-class environment. Accordingly, the "residual bandwidth" becomes a vector $\mathbf{g} = \{g_1, g_2, ...\}$, whose generic component g_i represents the amount of additional bandwidth that can be assigned to future requests of class i *in absence of new requests from other classes*. At any time, the computation of each component g_i is done independently from the others and involves a very simple manipulation of $\mathbf{r}$, $\mathbf{L}$ and $\mathbf{c}$, as described in the following. Consider the case that we want to compute the new value of g_j because the vector $\mathbf{v}$ (and eventually $\mathbf{r}$) has changed due to LSP installation / removal for some class - not necessarily j - on the specific link. We replace the component r_j with $r_j + x$ in vector $\mathbf{r}$, where $x \geq 0$ is a temporary support variable, and solve the simple optimization problem yielding the maximum value of x holding constraint (2), with parameters r_i set to their current values. This is a trivial task since the optimization involves a single variable and linear constraints. Finally, we derive $g_j = x - v_i$. This computation is repeated independently for each class except best-effort, so as to build the vector $\mathbf{g}$. This vector can be advertised in the Opaque LSA of OSPF-TE. Specifically, we use the sub-TLV `Unreserved-bandwidth` to carry it in conformance with the semantic defined in [13].

Given the independence between the g_i components, the same flooding reduction policy used for g in the single-class environment can be straightforwardly applied to each component separately. In particular, we used the same mechanism described in [9] based on adaptive watermarks.

We notice that in general a new reservation in some class impacts the residual bandwidth of all the other classes, in force of the composed constraints. We also remark that the sum of the components $\sum_i g_i$ can exceed the link capacity, since each element has been computed *in absence of new requests from other classes*. Such semantic greatly simplifies the derivation of $\mathbf{g}$ itself and the application of the flooding reduction algorithm, and most importantly it is fully coherent with the usage that the route selection algorithm described above will make of such information. Finally, we remark that the model proposed here comprises as special cases the models being currently discussed in IETF, [11] [14] [15], and is compliant with the requirements given in [11]. In particular, isolation is provided by u_i setting, while any inter-class sharing can be obtained by appropriate design of the constraint (2). The only difference between those drafts and our proposal regards the support of holding and setup priority, which is foreseen in our testbed as a future-phase development.

Finally, let us consider the possibility to apply traffic engineering to LSPs carrying best-effort (BE) traffic. We recall that there is no bandwidth reservation associated to best-effort

[1] In the current implemented version, we do not support bandwidth sharing, which has been referred to a future development phase. However, in the protection model presented in [5] [6] - originally inspired by the proposal in [12] - the sharing functionality is handled locally at the intermediate nodes along the paths, and does not require the separate dissemination of working and backup reserved bandwidth components. With this approach the future implementation of the sharing capability only requires an additional element to the RSVP-TE messages in order to distribute the working LSP path to the nodes along the backup LSP (as first proposed in [12]), and a different update rule for v_j^* when a backup LSP is installed / removed. For sake of space, we refer to [6] for further details on this point. What is important here to stress is that the implementation of bandwidth sharing can be regarded as a future add-on, and does not requires changes to the allocation constraints nor to the derivation of residual bandwidth described in the text.

traffic, still it is possible to envisage a model where explicit routing capability is applied to BE LSPs. That means that the new BE LSP can be routed over a more convenient route than the default shortest-path, through a RSVP-TE signaling with a valid Explicit Route Object (EROA and null reserved bandwidth. There are several ways to define a goodness criterium for best-effort paths, but in any case one has to take into account the current network load *as seen from the perspective of best-effort traffic*. To this purpose, one could think to define the "residual bandwidth" component for best effort (g_4 in our case) as the "measured residual capacity" of the link, that is $C - m$, with C the link capacity and m the last value of the whole aggregated consumed bandwidth. The value of m can be derived for example from the local SNMP entry accounting for the total bytes sent in the last measurement interval. The dissemination of such value in the `Unreserved-bandwidth` sub-TLV would be helpful to provide the edge nodes with a view of the real bandwidth usage on the links, in addition to the amount of bandwidth reservation.

IV. The Fault Protection scheme

The end-to-end protection approach requires some mechanism to convey the fault notification from the internal node(s) detecting the fault to the edge nodes which are in charge of switching the traffic onto the backup LSPs. In the MPLS-TE standards no solution has been given to address this point. Instead, this topic is currently being discussed in the IETF working groups in the context of GMPLS [10]. Basically two approaches are being compared: signaling-based and flooding-based. For our testbed we have preferred the latter, and we have implemented a prototype of such functionality in the OSPF-TE daemon.

A. The fault notification dilemma: signaling-based or flooding-based ?

The signaling-based approach foresees the fault notification message be carried by the RSVP-TE protocol, from the detecting node upstream along the LSP to the ingress edge node. This approach has several drawbacks. For instance the number of messages generated upon a failure equals the number of impacted LSPs, say N. Thus, the detecting node will generate N different messages, while each of its reachable neighbors (say k) will have to process N/k different messages on average. In order to diminish the processing overhead associated with this approach, one could think to the aggregation of such messages, similarly to the proposal made in [16]. On the other hand, this would require additional capabilities at the RSVP-TE daemon, namely message aggregation and branching, which are not currently foreseen in the protocol. Also, this solution would lead the signaling-based dissemination process to closely reassemble a sort of "partial flooding" along the network subgraph constituted by the upstream LSP tree, an approach that does not participate of the additional advantages of the complete flooding scheme discussed below. In any case, the signaling-based approach would require the detecting node to parse the entire set of supported LSPs so as to identify those impacted by the fault (as made explicit in [17]), and this would add processing burden to the internal router.

With the flooding-based approach, the fault notification message is flooded throughout the network. An attractive possibility would be to reuse the existing IGP (OSPF-TE or ISIS-TE) flooding process of Opaque LSAs, a capability that is intrinsically present in the MPLS-TE model. Recall that the Opaque LSAs are flooded transparently through the network, without triggering the re-computation of the routing / forwarding tables, and are used to disseminate link load information in MPLS-TE. With this approach, there is no need for the detecting node to parse the entire set of LSPs. Also, the per-node processing load after a failure is limited to one single message, while in the signaling-based approach without aggregation the number of messages can span from zero to N in the worst case.

Two additional advantages of the flooding-based approach are the *minimum notification delay* and *complete dissemination*. In fact, by assuming an equal processing time of the Opaque LSA and RSVP-TE messages, and that each node floods / forwards the received message immediately, it is easy to recognize that *IGP flooding always achieves the minimum possible notification delay towards any edge node*. In fact, the path followed by a flooded message to reach the generic edge node from the detecting one is always the minimum-delay path - not necessarily coinciding with the minimum-hop path, unless the transmission and propagation delay are the same for all the links - while in the signaling-based approach it depends on the actual length of the impacted LSP paths. These are computed according to bandwidth constraints and minimization / balancing objectives, which in general result in non-minimum delay paths. This point was already suggested in [18]: "[...] notification message exchanges through a GMPLS control plane may not follow the same path as the LSP/spans for which these messages carry the status. In turn, this ensures a fast, reliable [...] and efficient [...] failure notification mechanism".

Another general advantage of the flooding-based approach versus the signaling-based is that the immediate dissemination of the failure notification is complete, that is reaches all the edge nodes and not only those responsible for the impacted LSPs. This is helpful in preventing from erroneous route selections in the aftermath of the failure. In fact, with the signaling-based approach, those edge nodes not having any LSP crossing the failed link l will be not notified the failure. Therefore there is a potential for them to include the failed link into the selected path for a new LSP, which would result in fastidious signaling overhead and re-computation procedure. The potential for erroneous route selections is even higher if we consider the possibility of multiple contemporary failures. In fact, consider two generic links along the path of a generic LSP, say l_1 and l_2, the former being upstream with respect to the latter. If both links fail simultaneously, the propagation of l_2 failure will not reach the edge node, due to upstream

interruption of the path. Again, there is a potential for this edge node to include the failed link l_2 in the new selected routes. With flooding-based approach such potentials are eliminated as all edge nodes are immediately notified any failure.
The usage of link state IGP for fault notification is compliant with RFC 3272 [19], where it is stated that network state information may be distributed by link state advertisements also under exceptional conditions.
As an alternative to IGP flooding, in the GMPLS context it has been proposed to add flooding capabilities to the Link Management Protocol (LMP) [20]. Despite this approach has the undoubtedly advantage to avoid further modifications to the IGP platform, on the other hand it would fatally lead to the duplication of many flooding-related mechanisms between IGP and LMP, and at the same time breaks the local nature of the LMP protocol. Our preference for IGP was dictated by the fact that LMP does not belong to the MPLS-TE suite. One also has to consider the dramatic gap between the amount of implementation workload required to inject fault notification semantic in existing OPSF-TE message, compared to the implementation from-the-scratch of a new flooding process on LMP, that in the best case would achieve the same performances of the well-tested IGP flooding.

B. Details of the prototypical implementation

In this section we report on some details of our prototypical implementation of flooding-based fault notification through OSPF-TE Opaque LSAs.
The Opaque LSA option (O-LSA for short) has been defined in [21], where three types of O-LSAs are defined with different scope: type 9 (link-local, flooded only within the subnetwork), type 10 (area-local, flooded within the associated area) and type 11 (flooded throughout the entire AS). The usage of O-LSAs for traffic engineering purposes has been described in [13], that specifically prescribes to use type 10 LSAs. These are considered for the dissemination of link attributes, included currently unreserved bandwidth, and introduces a number of nine message elements associated to the link. These are included in the O-LSA, and are called the sub-TLVs of the "Link TLV". All such features were already supported by the OSPF implementation [22] that was used in the testbed.
We implemented the fault notification by means of type 10 LSA, therefore with area-level scope. This choice was mainly dictated by ease of implementation. We are aware that this solution only works if the whole network is included in a single OSPF area. This is not a problem on our small scale testbed, but it might be a serious restriction in large domains. Therefore, in that case, one should probably consider to use type 11 LSA [2].
In our implementation, we used the sub-TLV `TE-metric` introduced in [13] to carry the fault notification semantic: we arbitrary set the default value for such field to be 0, while the value 1 indicate failure of the link. We also manipulated the sub-TLV `Administrative-Group` to carry additional information about the membership of the link to some Shared-Risk Link Group (SRLG). This information is essential to the edge nodes to select SRLG-disjoint routes.
A key point we had to cope with is the presence of hold-down timers in OSPF. In OSPF [23] two timers are present to enforce a minimum spacing between consecutive LSAs referred to the same link. The first one, `MinLSInterval`, inhibits the generation of new LSAs for 5 seconds after transmitting a LSA. The second one, `MinLSArrival`, imposes the discarding of any new LSA received within 1 second since the last received LSA. With such timers, the flooding of an O-LSA advertising the failure of link j could be delayed in the case that the previous Opaque LSA for the same link was generated within the latter 5 seconds - for example to advertise a change in the reserved bandwidth. In order to eliminate this possibility, we introduced a mechanism called "timer forcing". That means the generation / reception of a new O-LSAs carrying a fault notification semantic (i.e., `TE-metric` set to 1) can force the hold-down timer to expire immediately and be reset. The timer is associated to a flag variable, which is set to 1 when the timer is forced, and returns back to its default value 0 when the timer expires normally. The arriving O-LSA can force the timer only if *i)* it carries fault notification semantic and *ii)* if the flag variable is set to 0. This again enforces a minimum spacing between O-LSA with a failure notification content. This is explained in fig. 2 limited to the `MinLSInterval` timer. With this mechanism we eliminated the interspacing between a generic O-LSA and the first fault notification O-LSA, but apart this "exception" the timer behavior remains compliant with the standard.

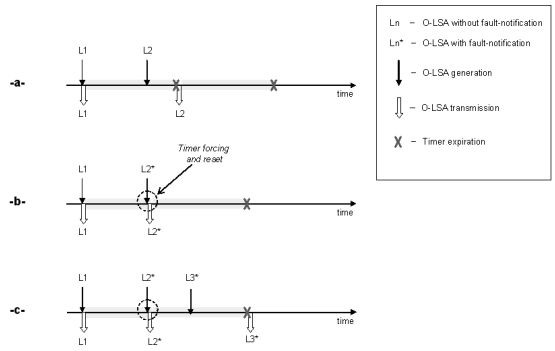

Fig. 2. Hold-down timer in OSPF-TE. According to the standard [23], after transmission of a LSA ($L1$) the subsequent LSA generated within 5 seconds ($L2$) is delayed fro transmission until timer expiration (a). We introduced timer-forcing for LSAs with a fault-notification content (b): in this case $L2^*$ is sent immediately and the timer is reset. A state flag ensures that a further LSAs ($L3^*$), still with a fault-notification content, can not force again the timer (c).

[2] As an alternative, one could think to a different protection / restoration model, intermediate between the per-link (or local) protection and the path-based (or global, or end-to-end) protection, where protection switching is enforced on a per-area basis. In this case, the switching nodes are not located at the edges of a single link, nor at the edges of the network, but rather at the edges of the IGP areas, so that area-local flooding would be sufficient to disseminate the fault notification message.

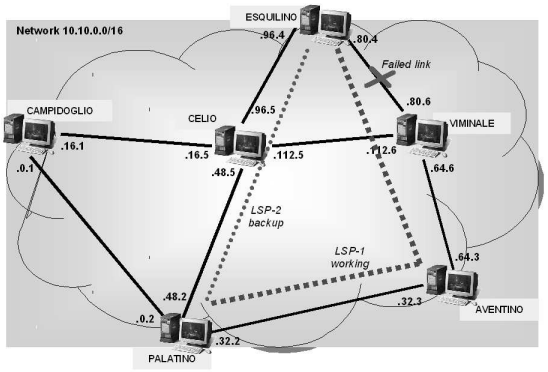

Fig. 3. Testbed topology.

V. Preliminary experimental results

In this section we briefly report the results of some preliminary experiments. In particular, we were interested in assessing the effectiveness of the flooding-based implementation of the fault notification, and the achievable recovery delay after a single link failure. To this purpose, we prepared the testbed topology depicted in fig. 3, where a single SFP connection is installed from host PALATINO to ESQUILINO, involving LSP-1 (working LSP) and LSP-2 (backup LSP). All PC are 2.0 Ghz CPU with Linux 2.4.20 kernel, equipped with Ethernet 10/100 cards. A packet generator is installed in CAMPIDOGLIO, sending UDP packets to PALATINO at the fixed rate 1 packet/ms. These packet are mapped by PALATINO into the working LSP-1 towards the destination. A sniffer records all the packets arriving to ESQUILINO on both its interfaces. At some instant, we disconnect manually the link between VIMINALE and ESQUILINO dividing LSP-1, and we count the number of packets that are lost in the transitory. For example, if the last packet received from LSP-1 has order number n, and the first packet received from LSP-2 has order number $n+m$, that means m packets in total were lost, corresponding to a recovery delay of approximately m milliseconds.

We repeated this experiments 25 times, and in most trials we observed a recovery delay between 75 ms and 90 ms. But in 3 trials out of 25 the measured values were between 100 ms and 230 ms. We do not yet have an explication for these occasional large values. Our preliminary investigations suggest that this might be due to occasional longer delays imposed to the packets by the forwarding module within the operating system. When this happens to the packets carrying the fault notification Opaque LSA, this translates into a longer notification delay. However, we are carrying further investigations in order to ascertain the cause of this occasional behavior.

VI. Conclusions

In this paper we briefly described the TANGO architecture, with a focus on two keypoints: the multi-class allocation model, and the fault notification mechanism based on OSPF-TE flooding. The proposed solutions are original, and we showed how they can be implemented within the existing protocols for MPLS traffic engineering, particularly OSPF-TE. On the basis of a prototypical implementation, we have run some simple experiments in order to assess a very basic performance metric, namely the achievable recovery delay.

We stress that such results are preliminary and were obtained in ideal conditions: a small unloaded network, without packets nor LSPs other than those under test. Collectively they confirm the expectation that the reference of 50 ms of Sonet/SDH can not be achieved at the packet layer. On the other hand, they seem to be encouraging towards the possibility of achieving end-to-end MPLS recovery in the order of few hundreds of milliseconds in operational conditions.

Our current efforts are directed to repeat the measurements in heavy-load network conditions, in order to assess the robustness of the architecture, particularly as regards the fault recovery scheme.

Acknowledgments

The authors would like to thank A. Bosco, A. Botta, N. Caione, G. Conte, P. Iovanna, A. Liburdi, A. Proia for their active contribution to the implementation and experimental activities reported in this work.

References

[1] D. Awduche et al. Requirements for Traffic Engineering Over MPLS. *RFC 2702*, September 1999.

[2] X. Xiao et al. Traffic Engineering with MPLS in the Internet. *IEEE Network*, March/April 2000.

[3] TANGO project homepage. http://tango.isti.cnr.it.

[4] F. Le Faucheur, W. Lai. Requirements for Support of Differentiated Services-aware MPLS Traffic Engineering. *RFC 3564*, July 2003.

[5] F. Ricciato, M. Listanti, A. Belmonte, D. Perla. Performance Evaluation of a Distributed Scheme for Protection against Single and Double Faults for MPLS. *2nd Int'l Workshop on Quality of Service in Multiservice IP Networks (QoS-IP 2003), Milano*, February 2003. Published in Lecture Notes in Computer Science, vol. 2601, Springer, pp. 218-232.

[6] F. Ricciato, S. Salsano, M. Listanti. An Architecture for Differentiated Protection agains Single and Double Faults in GMPLS. *to appear in Optical Networks Magazine*, 2003.

[7] A. Shainkh, J. Rexford, K. G. Shin. Evaluating the Overheads of Source-Directed Quality-of-Service Routing. *Int'l Conference on Network Protocols (ICNP)*, 1998.

[8] G. Apostolopoulos, R. Guerin, S. Kamat, S.K.Tripathi. Quality of Service Based Routing: A Performance Perspective. *SIGCOMM*, 1999.

[9] A. Botta, P. Iovanna, M. Intermite, S. Salsano. Traffic Engineering with OSPF-TE and RSVP-TE: Flooding Reduction Techniques and Evaluation of Processing Cost. *CoRiTeL Report. Submitted.*, 2003.

[10] Mail archive of the CCAMP working group. http://ops.ietf.org/lists/ccamp.

[11] J. Ash. Max Allocation with Reservation Bandwidth Constraint Model for MPLS/DiffServ TE and Performance Comparisons. *draft-ietf-tewg-diff-te-mar-01.txt. Work in progress*, June 2003.

[12] R. Doverspike, C. Kalmanek, G. Li, D. Wang. Efficient Distributed Path Selection for Shared Restoration Connections. *INFOCOM'02*, June 2002.

[13] D. Katz, K. Kompella, D. Yeung. Traffic Engineering Extensions to OSPF Version 2). *draft-katz-yeung-ospf-traffic-10.txt. Work in progress*, June 2003.

[14] F. Le Faucheur, W. Lai. Maximum Allocation Bandwidth Constraints Model for Diff-Serv-aware MPLS Traffic Engineering . *draft-ietf-tewg-diff-te-mam-00.txt. Work in progress*, June 2003.

[15] J. Ash. Russian Dolls Bandwidth Constraints Model for Diff-Serv-aware MPLS Traffic Engineering. *draft-ietf-tewg-diff-te-russian-03.txt. Work in progress*, June 2003.

[16] C. Huang, V. Sharma, K. Owens, S. Makam. Building Reliable MPLS Networks Using a Path Protection Mechanism). *IEEE Communications Magazine*, March 2002.
[17] J. P. Lang, B. Rajagopalan eds. Generalized MPLS Recovery Functional Specification. *draft-ietf-ccamp-gmpls-recovery-functional-00.txt. Work in progress*, January 2003.
[18] D. Papadimitriou, E. Mannie eds. Analysis of Generalized MPLS-based Recovery Mechanisms (including Protection and Restoration). *draft-ietf-ccamp-gmpls-recovery-analysis-01.txt. Work in progress*, May 2003.
[19] D. Awduche et al. Overview and Principles of Internet Traffic Engineering. *RFC 3272*, May 2002.
[20] T. Soumiya, R. Rabbat eds. Extensions to LMP for Flooding-based Fault Notification. *draft-soumiya-lmp-fault-notification-ext-01. Work in progress*, June 2003.
[21] R. Coltun. The OSPF Opaque LSA Option. *RFC 2370*, July 1998.
[22] Zebra Home Page. http://www.zebra.org/.
[23] J. Moy. OSPF Version 2. *RFC 2328*, April 1998.