Extending MPLS Traffic Engineering to deal with QoS

Alessio Botta[3], Paola Iovanna[1], Roberto Mameli[1], Giovanna Piantanida[1], Stefano Salsano[2]

[1] Ericsson Lab Italy S.p.A. - [2] DIE – Università di Roma "Tor Vergata" - [3] CoRiTeL - Consorzio di Ricerca sulle Telecomunicazioni

***Abstract* - Traffic Engineering (TE) deals with the performance optimization in operational networks, aiming for example at the fair distribution of traffic in order to avoid or minimize congestion. Multi Protocol Label Switching (MPLS) is a key technology that enables advanced TE functionality. The MPLS framework consists of several components, such as the label switched forwarding plane, the label distribution and routing protocols and the "routing decision engine". Extensions to these components to fully support Traffic Engineering are currently under discussion within standardization bodies. Combining the proposed extensions to different control plane components in a consistent architecture is not a trivial operation. This paper defines the architecture of a Traffic Engineered MPLS network and describes its implementation in a test-bed composed of Linux PCs acting as MPLS routers. The architecture is aligned with current discussion within IETF and includes as an option an extension to RSVP-TE proposed by the authors. The prototype implementation allows verifying the correct functional behavior of control plane protocols. The test-bed provides a flexible platform where further extensions can be tested and some kinds of performance tests are possible.**

1. INTRODUCTION

Traditional IP networks are typically based on a connection-less packet-switching paradigm: individual packets are routed separately and independently on a hop-by-hop basis, and there is no guarantee of timely delivery of packets (best effort service). However, the rapid growth of Internet traffic and the corresponding migration of real time and multimedia services towards IP, requires the introduction of technologies able to provide users with Quality of Service (QoS) and network operators with more dynamic and flexible resource utilization. In this scenario Traffic Engineering (TE – see [1]) plays certainly a key role and attracts a lot of interest. The aim of this paper is to describe an architecture for TE and to present its prototype implementation in a test-bed. The architecture is based on the drafts currently under discussion within IETF and it optionally includes some extensions to RSVP-TE proposed by the authors in [2]. The test-bed is realized by means of Linux-based PCs acting as MPLS routers and it provides a platform for control plane emulation of a TE-enabled network. This platform allows functional and performance tests and constitutes a flexible starting point for further extensions.

This work has been partially carried out in the context of the ASTERIX project (Advanced Solutions for Traffic Engineering: Research, Implementation and eXtensions). This project is sponsored by Ericsson and is done by Ericsson Lab Italy, in cooperation with the the Universities of Rome "Tor Vergata", "La Sapienza" and CoRiTeL.

The paper is structured as follows: chapter 2 provides a brief explanation of the basic issues related to TE and QoS and introduces MPLS as suitable technology to support them. Chapter 3 focuses on the main functional elements of a TE enabled network, describing how they act and how they cooperate. Chapter 4 discusses some issues related to the TE extension of control plane protocols currently under discussion in the IETF community and presents a proposal made by the authors. Chapter 5 focuses on the Linux implementation of single nodes, while chapter 6 describes the overall test-bed and illustrates the trials performed. Finally chapter 7 reports conclusions and illustrates the open issues left for future work.

2. TRAFFIC ENGINEERING AND QOS

MPLS [3] is a key technology able to enhance IP networks with advanced TE and QoS functionalities. It introduces a "connection-oriented" forwarding plane based on a label-switching paradigm. An MPLS label ("shim header") is inserted before IP header as shown in Figure 1. MPLS nodes use this label to route packets along pre-established paths called Label Switched Paths. Note that specific Layer 2 technologies (e.g. ATM) may have different ways to carry the MPLS label.

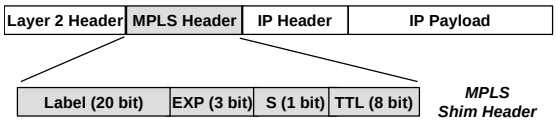

Figure 1 – MPLS Label format

Figure 2 represents the MPLS network elements:

- Label Switching Routers (LSR), which forward packets on a label switching basis;
- Label Edge Routers (LER), which classify incoming IP packets associating them with a label at the ingress, and strip off the label at the egress;
- Label Switched Paths (LSP), i.e. the virtual connections represented by the concatenation of labels along the path.

In the MPLS architecture there is a separation between the data plane and the control plane, at the purpose of reusing the existing signaling and routing protocols of the TCP/IP suite (with proper extensions). Figure 3 shows the control plane protocols in an MPLS network. Each node supports both a routing protocol and a label distribution protocol. As far as the routing protocol is concerned, the existing IP intra-domain routing protocols (OSPF, IS-IS) are used. The label distribution protocol, instead, is used to distribute labels during LSP setup,

possibly following an explicitly routed path. The main example is LDP [4].

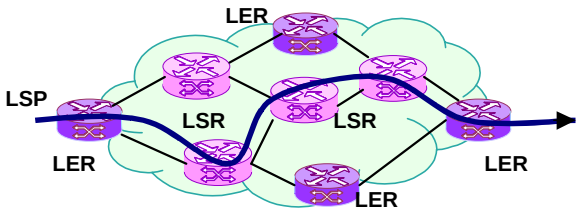

Figure 2 –MPLS network

It is worth noting that MPLS was originally designed to support faster packet switching, in order to overcome the performance limitation of IP routers, while Traffic Engineering was not among the design goals. Therefore MPLS control protocols did not natively support TE. However, when moving to a Traffic Engineering capable network, both the routing protocols and the label distribution protocols should properly evolve (MPLS-TE - [5]). Specifically, the routing protocols must be enhanced through the ability to carry information related to link attributes/states, to be used for explicit route calculation (e.g. available/reserved bandwidth). The TE enhanced versions of intra-domain routing protocol are OSPF-TE [6] and ISIS-TE [7]. In the following, we will refer to the TE enhanced version of routing protocol as "TE-e routing protocols". The label distribution protocols must be extended to support both explicit route indication (for efficient load distribution) and reservation of resources during dynamic LSP setup (for QoS support). RSVP-TE [8] and CR-LDP [9] are the two candidates for TE enhanced label distribution protocols. In the following, we will refer to the TE enhanced version of label distribution protocol as and "TE-e signaling protocol".

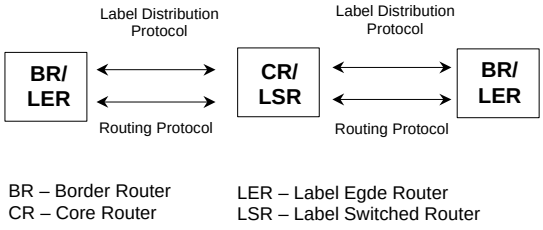

Figure 3 – MPLS: Control Plane Protocols

The MPLS-TE application described in [5] requires proper routing algorithms, able to find routes according to some optimization criteria. Such algorithms, often termed Constraint Based Routing (CBR), perform path calculation by taking into account various constraints, typically related to resource requirements of the flow, network status and administrative permissions. The discussion of these algorithms is out of the scope of this paper.

MPLS-TE was originally designed just for the purpose of performance monitoring and optimization of a TE-enabled network, leaving apart any type of QoS support. The interaction with a QoS mechanism, in particular the Differentiated Service paradigm was introduced only in a second time ([10]). Specifically, [10] defines two types of DiffServ enabled LSPs:

- E-LSP (EXP-inferred-PSC LSP): LSP able to carry up to 8 DiffServ Ordered Aggregates, whose PHBs are specified in the EXP field of the MPLS header (see Figure 1);
- L-LSP (Label-only-inferred-PSC LSP): LSP able to carry a single DiffServ Ordered Aggregate

More intuitively, L-LSPs support a single class of service and allow traffic engineering at a very fine-grained level, i.e. routing, protection & restoration and preemption separately for each service class. In contrast, E-LSPs are able to support a set of service classes; they are traffic engineered (routed, protected, restored, etc.) all together, thus providing less flexibility. However E-LSPs could perform better in terms of scalability; in fact, the management of a single E-LSP encompassing N classes (instead of N L-LSPs) requires less signaling and smaller routing tables. Of course the choice of either one or the other is up to the provider's needs.

Taking into account the above discussion, the evolution of MPLS functionality and related standardization can be graphically represented as in Figure 4.

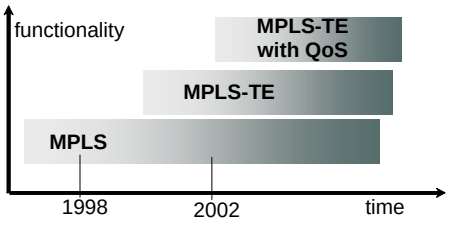

Figure 4 – MPLS evolution

3. FUNCTIONAL ELEMENTS IN A TE-ENABLED NETWORK

In a network performing Traffic Engineering functions, the setup of an LSP is the result of a TE decision taken by a "Route Decision Engine". We assume that this decision process is distributed, therefore route decision engines are logically coupled with network edge nodes, where the LSP must be originated. A Route Decision Engine (RDE) is a logical process, from the physical standpoint it can either run "on" the LER or it can run on a separate machine connected to the edge node. The RDE interacts with the TE-e routing protocol to gather information related to network topology and current resource utilization and with the TE-e signaling protocol to request the setup of the LSPs. The RDE uses Constraint Based Routing (CBR) algorithms that try to optimize the resource utilization in the network.

The Route Decision Engine may act on behalf of some controlling entity that tells which LSPs need to be setup and

which are the LSP requirements (e.g. required bw). In our prototype this entity is an operator console with a Graphical User interface (GUI). This is only one of the possible solutions in a real network. Other solution could include classification/measurement tools that can evaluate the requirements for LSPs from running traffic [11].

Hereafter we analyze the interaction in the control plane between the TE-e routing protocol, the TE-e label distribution protocol and the Route Decision Engine to perform the TE functions. This constitutes a relevant contribution of this paper, because it gives a simple consistent vision of the TE process.

Within a Label Edge Router, the TE-e routing protocol and the TE-e signaling protocol co-operate to perform Traffic Engineering functions and interact with the Route Decision Engine that takes the decisions on LSP setup. Moreover, looking at the data plane aspects, the routing protocol and the signaling protocol need to interact with the routing table and with the MPLS connection table. Within a core Label Switching Router the control interaction is restricted among the TE-e routing protocol and the TE-e signaling protocol, as the RDE is missing. The data plane interactions are the same as in the LER.

A functional representation of a Label Edge Router is depicted in Figure 5. A core LSR, which does not originates LSP, has a similar architecture, but it is missing of the Route Decision Engine.

In the following description of the interaction between functional elements, we will follow the choices we have made in the ASTERIX test-bed: OSPF-TE is used as TE-e routing protocol and RSVP-TE as TE-e signaling protocol. Anyway, similar considerations apply to the general scheme depicted in Figure 5. For simplicity we will use "OSFP-TE"/"RSVP-TE" to refer also to the entity that implements the OSPF-TE/RSVP-TE protocol (e.g. the "protocol daemon" process using Operating System jargon).

The RDE gathers the information related to topology and resource usage by the OSPF-TE. This is a continuous process, because the RDE should always use the most updated information when it evaluates a new route. Basically, the OSPF-TE communicates the complete topology and resource database to the RDE.

When the RDE has evaluated a route for an LSP to be setup, it communicates a request to the RSVP-TE. The request includes the explicit route, the traffic class and the amount of required resources. The RSVP-TE protocol propagates the LSP setup request along the requested route. The RSVP-TE *PATH* messages are sent from the source LER up to the destination LER, then the destination LER is in charge to send back the RSVP-TE *RESV* messages. These messages are propagated backward, so that each LSR can actually reserve the resources on each outgoing link. During the *RESV* phase an admission control procedure is performed in order to check the actual availability of resources on the outgoing link. It is possible that an LSR rejects the LSP setup, despite the fact that the route was requested by RDE after performing a proper routing algorithm. Basically, there are two reasons: first, the resource and topology information available at a given RDE cannot be always updated and accurate; second, the distributed nature of the architecture makes it possible that two RDEs request for the same resource at the same time. These error situations must be properly handled in the LSRs by releasing the previously allocated resources and in the requesting LER by communicating it to the RDE.

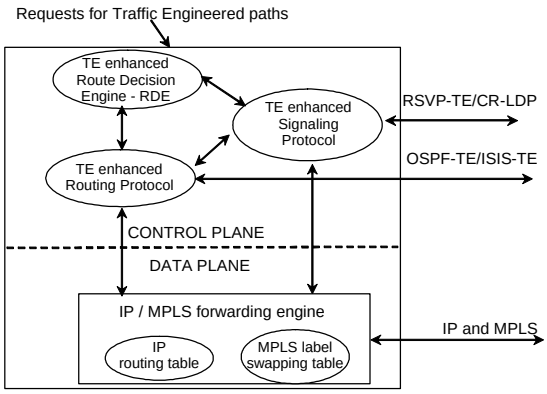

Figure 5 – Architecture of a TE enabled node (LER case)

Each time that a node receives and accepts the setup of an LSP by means of a *RESV* message, its resource allocation status is changed. This should be communicated to all other nodes by means of the OSPF-TE protocol. Therefore the RSVP-TE internally interacts with OSPF-TE, communicating the current status of resource allocation.

Once the OSPF-TE has been notified of a change, it should advertise this change to all other LSRs by sending a special kind of Link State Advertisement (LSA) messages called *opaque LSA* and using the OSPF "flooding" procedure. Mechanisms are needed to avoid that this flooding is executed for each minimal change, leading to an avalanche of OSPF-TE messages in the network.

After that the OSPF-TE flooding procedure is executed, all LERs have been updated with the new status and the RDE have the correct information to evaluate the best route for the setup of further LSPs.

4. STANDARDIZATION ISSUES AND PROPOSED EXTENSION TO RSVP-TE

The framework described above is currently under standardization by IETF. As a consequence there are several details that have not been mentioned in the previous paragraphs, but that are still under analysis and evaluation both by the research community and the standardization bodies. Moreover, there are some inconsistencies among different proposals from different working groups that need to be solved

in order to reach a coherent network model. A typical example is given by OSPF-TE and RSVP-TE. In fact, while RSVP-TE signal bandwidth requirements associated to service classes, OSPF-TE floods bandwidth information on a preemption priority basis, i.e. following different criteria. This topic is currently under discussion in IETF within the thread related to Class-Type definition and application .

The role of an experimental test-bed is of key importance in such cases, since it constitutes a valuable experimental tool and it allows the evaluation of the effectiveness of a solution on a real prototype with minimal effort. Specifically, until now we have mainly focused on two major aspects. The first one concerns the introduction of Class-types, according to the proposals discussed in IETF (discussed hereafter in section 4.1). The second one, instead, represents the original contribution previously mentioned and proposed by the authors in [2]. It aims at introducing in the RSVP-TE protocol a new functionality for signaling separate bandwidth requirements for different service classes within the same E-LSP, at the purpose of achieving better aggregate TE (this is discussed hereafter in section 4.2).

4.1 Class-type for flooding reduction

The concept of Class-type (CT) was originally introduced in [1] at the purpose of improving IGP flooding scalability while allowing better bandwidth sharing among service classes. Basically a Class-type is a set of traffic classes whose requirements (e.g. bandwidth) can be aggregated. . Note however that it could be possible to implement setup priority policies for classes in the same Class-type in order to support preemption.

At the time of writing there are several proposals in the IETF TE Working Group about Class-types (see e.g. [12], [13], [14] and [15]) and the concept is not yet completely standardized. Within MPLS a traffic class is called “Per-hop Scheduling Class” (PSC). We have chosen to implement up to 8 Class-Types with static mapping of traffic classes (PSC) into Class Types. This means that the nodes in the network manage resources on a per-CT basis, i.e. each time an LSP is set-up or released, the corresponding bandwidths are borrowed from or returned to the CTs corresponding to the set of supported PSCs (according to a pre-configured mapping). Similarly, IGP flooding is performed on a per-CT basis, i.e. OSPF-TE advertises link state information separately for each CT, and not for each class, thus enhancing scalability.

4.2 Introduction of the MPLS FlowSpec for RSVP-TE

As stated above, the traditional DiffServ extension for MPLS ([10]) supports two types of DiffServ capable LSPs, namely E-LSPs and L-LSPs. However the DiffServ enabled TE solution described in [12] seems to privilege the use of L-LSPs (or E-LSPs carrying a single service class), which provide fine-grained TE (per-class routing and protection/restoration), at the price of increased number of LSPs. The current standard also allows coarse-grained aggregate TE by means of E-LSPs carrying multiple service classes. However, this option lacks of an important feature, since the different service classes carried within the same E-LSP are characterized by a single aggregate bandwidth requirement. This limitation allows a relatively poor form of aggregate TE, because Constraint Based Routing can only be performed assuming a fixed ratio among the bandwidth occupancies of all the classes that belong to the same E-LSP.

To solve this problem and to provide an enhanced form of aggregate TE, the authors have proposed the introduction of some limited extensions to the RSVP-TE protocol ([2]). Basically this proposal introduces a new FlowSpec for RSVP-TE, which allows signaling and reservation of resources on a per-class basis within an E-LSP. The proposed extension has been implemented in the test-bed, introducing the new object and managing all the issues related to the new functionality of per-class resource management and admission control.

5. THE IMPLEMENTATION ON THE LINUX PLATFORM

The prototype implementation has been realized by means of general purpose PCs with a Linux Operating System. The reason for choosing this platform is mainly due to the huge availability of open source software and by the possibility to modify and extend existing protocols. The PCs (Pentium III 350MHz and 600MHz) are interconnected by means of fast Ethernet point-to-point links, and are equipped with Linux RedHat7.1 distribution and kernel 2.4.9. The starting point for our work was a set of open source software packages leading to the configuration shown in Figure 6:

- MPLS provided by Sourceforge ([16]);
- RSVP-TE daemon from Tequila project ([17], [18]);
- OSPF daemon by Zebra ([19]).

As shown in Figure 6 in this configuration there is no integration among all the elements. RSVP-TE and OSPF-TE run separately, without the possibility to communicate one another, and there is no functional element like a Route Decision Engine to control the set up of LSPs. In order to support the functional architecture described in Figure 5, the following elements have been developed:

- a socket based interface between OSPF-TE and RSVP-TE to allow inter-communication (see Figure 5);
- a Java based Route Decision Engine, with a “catalogue” of some well-known CBR algorithms like min hop and widest shortest path implemented in C;
- a Java based interface between the RDE and OSPF-TE to gather network topology and resource information
- an interface between the RDA and the RSVP-TE to send LSP setup request
- a Java based Graphical User Interface giving the operator the ability setup and tear-down LSPs.

Moreover, some additional functionality has been added both to RSVP-TE and to OSPF-TE and contributions have been given to the open source development process.

In every router, when a PATH message is received, the RSVP module has to evaluate if there is resource availability for the Class Type (or Types in case of E-LSP) carried by the LSP, sufficient to accept the reservation. Therefore the Admission Control is performed recording resource usage per CT.

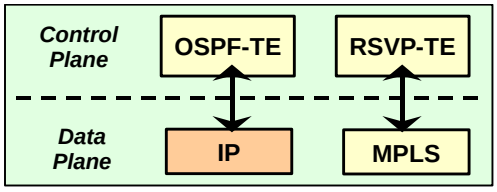

Figure 6 – Available Open Source Components

The possibility to set up E-LSP, with multiple Class Types reservations, led to enhance the the OSPF-TE daemon with the capability to change several CT bandwidth values simultaneously. This results in the flooding of a single Link State Advertisement (LSA) instead of sending an LSA for each CT bandwidth value changed. .

The resulting LER architecture is depicted in Figure 7.

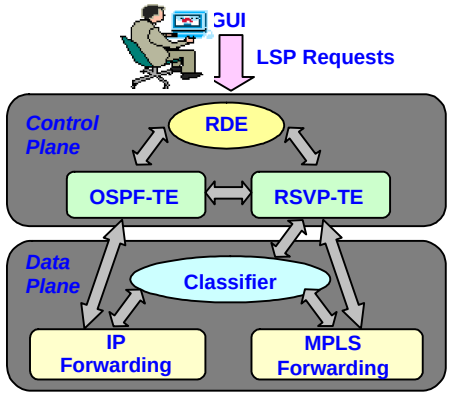

Figure 7 –MPLS TE component in the Linux box

Looking at the Control Plane, in order to set up a LSP requested by the GUI, the RDE needs to know the state of the entire network in term of resource (bandwidth) availability. This is achieved by the interface with OSPF-TE and the optimal path is chosen according to the specific CBR algorithm. This this path is given to RSVP-TE, to start the signalling (bw reservation and label distribution) needed to set up the LSP.

Once all the routers along the path have accepted the request, the LSP is set-up and each router can communicate its new bandwidth availability values. This is achieved by means of the communication on interface between RSVP-TE and OSPF-TE, followed by OSPF-TE flooding.

As said before, a Java based GUI has been realised to interact manually with the RDE using simple commands. The GUI appearance is shown in Figure 8.

An operator can choose the ingress and the egress LER of the LSP, the type of CBR he wants the RDE to implement, which kind of Diff-Serv LSP (E-LSP, L-LSP, or no DiffServ) to set up, and finally the type and the amount of bandwidth reservation.

Moreover the operator can monitor the status of the network. He can see all the LSPs installed with their attributes and can tear them down. He can also have a complete view of the network topology with bandwidth attributes for each link that is used by the RDE for path selection.

6. TESTBED

The testbed is composed of 7 PCs, interconnected according to the topology shown in Figure 9. We have emulated a Traffic Engineered network with 2 classes of traffic, realizing a static mapping of traffic classes (PSCs) into 2 CTs. Each node is configured with a maximum bandwidth available for each traffic class.

The OSPF-TE distributes the topology and the information of resource utilization per traffic class. It is able to send the unreserved bandwidth values for every CTs in a single *opaque* LSA. This will lead to a consistent reduction of overhead messages.

At the beginning all resources are available. Following the LSP setup procedures, resources are removed along the paths taken by the LSPs. The new bandwidth availability values are communicated through OSPF flooding, with proper mechanisms to control the amount of the exchanged information. In this way, each LER is able to select a constraint based route for every LSP request it receive, with a good approximation of resources occupation in the whole network at the moment the request is received.

We assumed that all the routers in the network are Label Edge Routers, except Panoramix, which is a core LSR. As a consequence, there is the possibility to set up a large number of LSPs with different sources and destinations. It is also possible to let the RDE to select its best paths according to the CBR algorithm implemented, or to force the LSPs to pass in specific routes. This leads to an accurate ability of controlling the behavior of any single component of the control plane implemented in the routers, especially in critical situations.

It is possible either set up and release single LSPs via the GUI, or run a simulation where a number of LSP setups and releases are automatically performed by LER at given times. In this way, it is possible to monitor the performances of our control plane architecture in critical situations and to take strategic decisions to improve the implementation or to guide the refinement of the specifications.

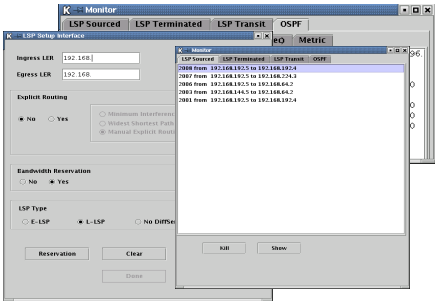

Figure 8 – Graphical User Interface

Both L-LSP with a single traffic class, and E-LSP with multiple traffic classes reservation and signaling can be used. In this second case, the RSVP-TE extensions proposed by the authors ([2]) allows a per class resource management and admission control. This means that each LSP setup request will be accepted only if there are resources available for every CT carried by the LSP. When the request is accepted resources will be removed from the available resource for each CT and all the new bw values will be communicated at once by OSPF with a single opaque LSA.

The correct setup of the LSPs on the data plane has been also checked with monitoring tools like Tcpdump and Ethreal.

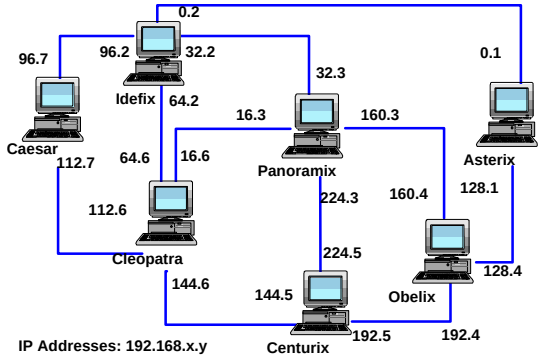

Figure 9 – Testbed topology

7. CONCLUSIONS AND FUTURE WORK

This paper presented an architecture for Traffic Engineering in MPLS networks and its implementation in a control plane test-bed. The test-bed is open and flexible to experiment control plane extensions, new strategies, the evolution towards GMPLS.

The effort to put all the "pieces" together is very important and we feel that special care is needed to coordinate the activity on the standardization of: TE extensions of routing protocol, TE extensions of label distribution protocol and of the architectural framework for Diffserv over MPLS.

An extension to RSVP-TE protocols has been proposed [2] and implemented to easily support the resource requirements of multiple class types in a single E-LSP

We are currently working in order to test the performance of the aggregate TE allowed by the proposed RSVP-TE extension.

8. ACKNOWLEDGEMENTS

Authors would like to thank all the participants of the ASTERIX project that contributed with their work and suggestions. Special thanks to Gianfranco Fallucca and Lorenzo Scaloni for their work on the test-bed implementation and to Eleonora Manconi for her support in the analysis of Label Distribution Protocols.

9. REFERENCES

[1] D. Awduche et al., "Overview and Principles of Internet Traffic Engineering", IETF RFC 3722, May 2002

[2] P. Iovanna, R. Mameli, G. Piantanida, S. Salsano, "Definition of the MPLS FlowSpec for RSVP-TE", work in progress, available at ftp://ftp.coritel.it/pub/Drafts/draft-iovanna-rsvp-mpls-flowspec-00.txt

[3] E. Rosen et al., "Multiprotocol Label Switching Architecture", IETF RFC 3031, January 2001

[4] L. Andersson et al., "LDP Specification", IETF RFC 3036, January 2001

[5] D. Awduche et al., "Requirements for Traffic Engineering Over MPLS", IETF RFC 2702, September 1999

[6] N. Bitar et al., "Traffic Engineering Extensions to OSPF", IETF <draft-bitar-rao-ospf-diffserv-mpls–02.txt>, work in progress

[7] T. Li, H. Smith, "IS-IS extensions for Traffic Engineering", IETF <draft-ietf-isis-traffic-04.txt>, work in progress

[8] D. Awduche et al., "RSVP-TE: Extensions to RSVP for LSP Tunnels", IETF RFC 3209, December 2001

[9] B. Jamoussi et al., "Constraint-Based LSP Setup using LDP", IETF RFC 3212, January 2002

[10] F. Le Faucheur et al., "MPLS Support of Differentiated Services", IETF RFC 3270, May 2002

[11] X. Xiao, A. Hannan, B. Bailey, L. M. Ni, "Traffic Engineering with MPLS in the Internet", IEEE Network, March/April 2000

[12] F. Le Faucheur et al., "Requirements for support of DiffServ aware MPLS Traffic Engineering", IETF <draft-ietf-tewg-diff-te-reqts-07.txt>, work in progress

[13] F. Le Faucheur et al., "Protocol extensions for support of Diff-Serv-aware MPLS Traffic Engineering", IETF <draft-ietf-tewg-diff-te-proto-03.txt >, work in progress

[14] J. Ash et al, "Proposed MPLS/DiffServ TE Class Types", IETF <draft-ash-mpls-diffserv-te-class-types-00.txt>, work in progress

[15] J. Ash et al, "Alternative Technical Solution for MPLS DiffServ TE", IETF <draft-ash-mpls-diffserv-te-alternative-02.txt >, work in progress

[16] MPLS for Linux Home Page (SourceForge), http://mpls-linux.sourceforge.net/

[17] RSVP-TE daemon for DiffServ over MPLS under Linux (IBCN), http://dsmpls.atlantis.rug.ac.be

[18] IST Tequila Project Home Page, http://www.ist-tequila.org

[19] Zebra OSPF Home page, http://www.zebra.org/