RSVP-TE Extensions to Provide Guarantee of Service to MPLS [1]

Francisco J. Rodriguez-Perez, Jose Luis Gonzalez-Sanchez, Alfonso Gazo-Cervero
University of Extremadura - Escuela Politecnica, Avda. Universidad s/n
10071 Caceres, Spain
{fjrodri, jlgs, agazo}@unex.es

Abstract. Independent Quality of Service (QoS) models need to be set up in IP and ATM integration and they are difficult to coordinate. This gap is bridged when MultiProtocol Label Switching (MPLS) is used for this purpose. We propose Guarantee of Service (GoS) to improve performance of privileged flows in congested MPLS networks. We first discuss the GoS requirements for the use in conjunction with MPLS. Then we propose a minimum set of extensions to RSVP-TE that allow signaling of GoS information across the MPLS domain.

Keywords: Guarantee of Service, MPLS, RSVP-TE, local recoveries.

1 Introduction

Multiprotocol Label Switching (MPLS) is currently mainly used to provide Virtual Private Networks (VPNs) services or in IP-ATM with QoS integration purposes [1], also combining ATM traffic engineering capabilities with flexibility of IP and class-of-service differentiation. MPLS bridges the gap between IP and ATM avoiding the need of setting up independent QoS models for IP and for ATM, which are difficult to match. ATM switches can dynamically assign Virtual Path Identifier/Virtual Channel Identifier (VPI/VCI) values which can be used as labels for cells. This solution solves the problem without the need for centralized ATM-IP integration servers.

Like ATM Virtual Circuits (VCs), MPLS Label Switched Paths (LSPs) let the headend Label Edge Routers (LER) to control the path that traffic uses towards a specific sink. LSP tunnels also allow a variety of policies related to network performance optimization [2]. Resource ReSerVation Protocol (RSVP) is a signaling mechanism used to reserve resources for these LSP tunnels. MPLS can reserve bandwidth on the network when it uses RSVP to build LSPs. Unlike ATM, there is no forwarding-plane enforcement of the reservation. A reservation is made in the *control plane* only, which means that if a Label Switch Router (LSR) makes an RSVP reservation and later it needs a bigger bandwidth, it will congest that LSP, damaging performance of other flows which can have even more priority, unless we attempt to police the flows using QoS techniques. Although RSVP with Traffic Engineering (RSVP-TE), is expected to be an important application in such problematic [3], an extended RSVP-TE protocol can be used in a much wider context for performance

improvement. MPLS-TE is providing fast networks, but assuming that devices are not going to fail and without data loss. However, resource failures and unexpected congestions cause a great part of lost traffic. In these cases, upper layers protocols can request lost data retransmissions at end points, but the time interval to get retransmitted data can be significant. For some types of services with high requirements of delay and reliability, as stock-exchange data or medical information, MPLS is not able to ensure that performance will not be worse due to lost traffic end-to-end (E2E) retransmissions.

In this work we describe a set of extensions to MPLS RSVP-TE signaling to provide GoS over MPLS. Thus, we will allow to offer GoS to privileged data flows [4][5], making discarded packets due to congestion to be locally recovered, avoiding in this way, as far as possible, E2E retransmissions requested by upper layers.

Following section shows how GoS can be applied to privileged MPLS flows. In the third section we study the RSVP-TE extensions to transport GoS information through the domain. In fourth section an analysis of the proposal is shown and finally this article concludes indicating the contributions of the research.

2 GoS over MPLS

The GoS capabilities for a MPLS privileged data flow is the capacity of a specific node to local recovering of discarded packets belonging to such flow. This work proposes up to four GoS levels (see Table 1), codified with two bits; so each packet can be marked with this information throughout all the route. A greater GoS level implies a greater probability that a packet can be found in the GoS buffer of any node of its LSP. Thus the need of end to end retransmissions is avoided, recovering lost data in a much rather local environment.

Implementation of GoS levels is carried out by means of the MPLS packet header, in the network level header and upper layers headers too. The main implied levels in an MPLS communication are *Network*, *Link* and level 2+ or MPLS. However, we have to bear in mind the possibility of marking GoS levels in *Transport* layer for *Application* level packets. Thus, following the TCP/IP model, data is marked with GoS at *Application* level directly by user and after that, the process would mark the TCP segments to be encapsulated in IP packets, which finally would receive a label to be switched across the MPLS domain.

At *Application* level, a GoS capability session can be started selecting a specific port when opening a TCP socket.

Table 1. GoS Levels Codification.

GoS_1	GoS_0	Meaning
0	0	No GoS packet.
0	1	Level 1 of GoS.
1	0	Level 2 of GoS.
1	1	Level 3 of GoS.

For example, in order to use email service we access to the port 110 or we use port 22 to SSH services. In this way, GoS use three ports to open TCP sessions, mapped with each one of the three GoS available levels. This will cause the *Transport* and upper levels to be marked with GoS. Moreover, at *Network* level the GoS mark has been implemented in the IP *Options* field, which has a size of at most 40 bytes. However only the first byte of this field is needed to codify the two bits for GoS. Finally, to mark a packet with GoS in MPLS level, the *label* field has value 1, which has been defined as a special value for MPLS labels. The *EXP* field (see figure 1) can transport the two bits needed for GoS. This mark can be set by the ingress LER.

2.1 GoS packets identification

In GoS nodes, a temporal buffer called NonStop-Forwarding Memory for GoS PDUs (NMGP) is needed. Moreover, GoS packets buffered in these nodes must also be identified to allow a GoS packet which satisfy a local retransmission request can be found. So privileged PDUs will be indexed in these buffers, allowing all sent and received GoS packets are globally identified in the MPLS domain, for nodes which request local retransmissions recognize each packet whose retransmission is needed as well as for upstream nodes to find stored GoS marked packets.

The IP address from *Network* layer allows to identify each node in a network topology so it can identify data flows, but it can not identify each packet sent by a specific node. An *id* identifier will go with each GoS packet and will be assigned by the sender node that generates it. A four octets identifier allows to recognize up to 2^{32} = 4,294,967,296 packets sent by a node. So *Network* level address of the sender, and this four bytes *id* will be considered as unique identifier for a GoS packet. This *id* field will also be marked in the *Options* field, after the GoS level field (see figure 2). In case of IPv6, GoS information can be forwarded with the *Hop-by-Hop* optional header, to be processed in every node throughout the LSP. This header also allows a No-GoS node to ignore GoS data and to continue processing the IPv6 packet.

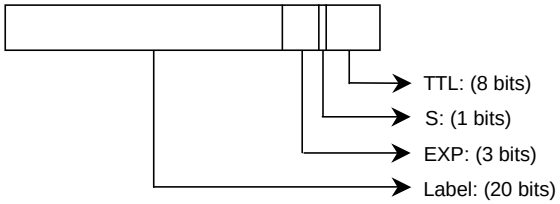

Fig. 1. MPLS packet header structure.

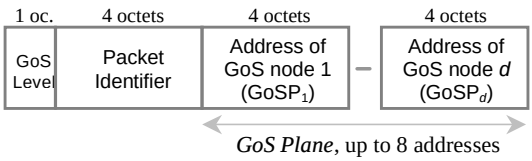

Fig. 2. *IP Options* field format for characterization of GoS packets.

2.2 GoS Path Marking and Local Recoveries

We consider a domain $G(U)$, with a set of nodes U and a data flow $j(G)=j(x_i, x_n)$ in $G(U)$ across a path $LSP_{i,n}$, with origin in node x_i and destination in node x_n, with $\{x_i, x_n\} \subset U$. Node x_n only knows incoming port and incoming label of every arrived packet of $j(G)$, i.e., x_n only knows that x_{n-1} is the sender of $j(x_i, x_n)$. It could know which node is the sender of a packet basing on label information, but this is not a reliable strategy because node x_{n-1} could use flow aggregation mechanisms to merge k flows coming from other nodes into a unique flow, in the form:

$$j(x_{n-1}, x_n) = \sum_{i=1}^{k} j_i(x_{n-1}, x_n) \cdot \quad (1)$$

On the other hand, if x_n, due to congestion, do not keep Flow Conservation Law:

$$\sum_{j=1}^{k} p_{nj} < \sum_{i=1}^{k} p_{in} \ , \quad (2)$$

being p_{ij} the traffic volume sent from x_i to x_j through x_n; so node is discarding one or more packets. In this case x_n cannot find any node to request local retransmissions of lost packets. It is very important to know the set of nodes by which a specific GoS packet has passed through and this is known as *GoS Path Marking*. Thus, x_n will know that discarded traffic can have been stored in upstream GoS nodes in $LSP_{i,n}$. The first node to request a local retransmission will be the starting node of the *GoS Plane*, i.e., its previous GoS neighbor. To get this stack of GoS nodes we have to obtain the set of nodes X such that $X \acute{I} LSP_{i,n} = \{(x_i, x_{i+1}), (x_{i+1}, x_{i+2}), ..., (x_{n-1}, x_n)\} \acute{I} U$, of $G(U)$ domain, with maximum *diameter* $d(x_i, x_n)=n-i$ such that X are *GoS* capable. In this way, with packet discarding, a local retransmission could be requested to any node belonging to X, avoiding requests to the head end and bringing a lesser increment of global $j(G)$ in the domain.

Path marking at MPLS level implies using of several bits from the label, making that Non-GoS nodes ($LSP_{i,n} - X$) do not know how to handle GoS traffic. So working at network level is a better strategy, i. e., GoS nodes of $LSP_{i,n}$ mark its network level address in the IP *Options* field of the GoS privileged packets. This stack of network address of nodes that have switched the packet is known as *GoS Plane* and the number of elements of this stack is the *diameter (d)* of the *GoS Plane*. Maximum value of d is $max\ d = ((OS-BU)/BpA)$, where *OS* is the IP *Options* field size (40 bytes); *BU* is the number of bytes used in the GoS proposal for packet characterization (1 byte for GoS level and 4 bytes for packet identification); *BpA* (*Bytes per Address*) is the number of bytes needed to codify an IP address (4 bytes). The value $d = 8$ is the maximum supported *GoSP diameter*. The objective of GoS is not to propose the replacement of all the nodes in a MPLS domain but the incorporation of several GoS capable MPLS nodes. In this way, in case a local retransmission was necessary in a node, there is a *GoS Plane* of at most 8 nodes to go upstream, increasing possibilities of finding lost packet. Moreover, *Internet Effective Diameter* (*IED*), that is defined as the maximum number of indispensable hops that are needed to reach to any other node in Internet [6], shows that rounding to 4, approximately 80% of the pairs of nodes in Internet are reachable in this distance. If

we consider an effective diameter of 5, it covers more than 95% of the pairs of nodes so a *GoSP diameter* of at most 8 nodes is a suitable size.

The last *d* GoS nodes which have switched a specific GoS packet is always known. This stack will also be marked in the *Options* field, after the GoS level field and after the four bytes packet identifier. So, in order to support GoS, the IP *Options* field of a packet will be formatted like in figure 2 is shown.

3 GoS Signaling

The specification of RSVP-TE [7] defines extensions to the Resource reSerVation Protocol (RSVP) in order to make network resources reservations and to distribute labels, establishing LSPs with traffic engineering capabilities. Among these extensions are the ability to specify a strict path to be followed by an LSP or supporting of state recoveries.

RSVP messages are send encapsulated in IP packets and are composed of header and a set of objects. For example the Hello Object enables RSVP routers to detect when neighbor nodes are not reachable, so this mechanism provides a very local and effective failure detection. Thus, the Hello extension is designed in the way that one side can use the mechanism while the other side does not and may be initiated at any configured failure detection interval and at any time; there are two types of *Hello* objects: *Hello Request* and *Hello Ack*. Nodes with no Hello capabilities or not configured for it, can ignore this messages, i.e., reception of Hello messages not alter the common operation of any node. It is intended for use between immediate neighbors, so *Time To Live* (*TTL*) IP field must be 1; however, with a *TTL*>1 it could be used as *keepalive* between non neighbors nodes.

3.1 RSVP-TE Hello Message Operation

A node may periodically (default value is 5 ms) generates a Hello message containing a *Hello Request* object for each neighbor who's status is being tracked. For every *Hello Request,* neighbor must send a *Hello Ack* (see figure 3). If no messages are received within a configured number of *Hello* intervals (default for this is 3.5 intervals), then a node presumes that it cannot communicate with the neighbor. They also compare new received values of *Source* and *Destination Instance* fields with the values most recently received from every neighbor and with last values send to them. This is used to assume that communication with the peer has been lost too.

3.2 GoS Extended Hello Operation

In [7] an extension for *Hello* message is proposed for handling nodal faults, relates to the case where a node losses its control state (e.g., after a restart) but does not loose its data forwarding state, as well as for control channel faults, relates to the case where control communication is lost between two nodes. The format of this extended Hello message is:

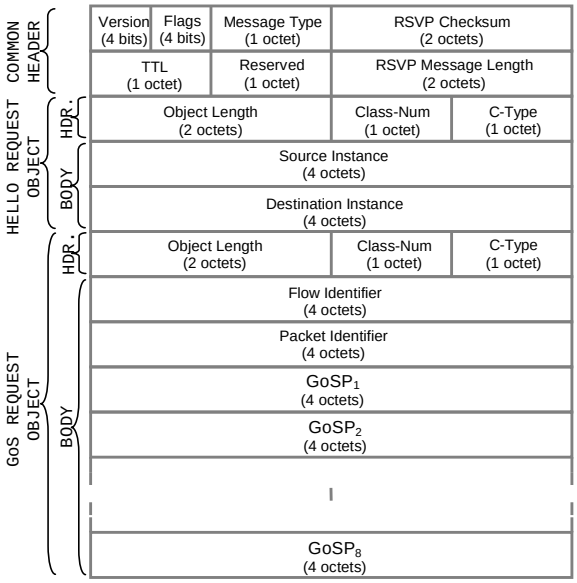

Fig. 3. GoS extended Hello message format, with common Hello Request and *GoS Request* objects.

```
<Hello Message> ::= <Common Header> + [<INTEGRITY>] +
<HELLO> + [<RESTART_CAP>]
```

In this work a GoS Hello message is proposed with following format:

```
<Hello Message> ::= <Common Header> + [<INTEGRITY>] +
<HELLO> + [<GoS>],
```

which, besides a Hello object, also includes an object with a GoS request or a GoS ack. GoS nodes will use information of Source and Destination Instances of common *Hello objects* to test connectivity with neighbors in GoSP as explained above. Formats of *GoS Request* and *GoS Ack* objects are in figures 3 and 4.

The usual state of a GoS MPLS node is *data forwarding* state, switching labels and forwarding data packets to the next node (see figure 3). There are only two events that change this state in the GoS node (see figure 5). One of them is detection of a discarded GoS packet. In this case node is able to capture GoS characterization information of discarded packet (see figure 2) and change its state to *request of local retransmission*, to send a extended Hello message with a GoS Request to the first node of GoSP ($GoSP_1$) (see figure 6). When a *GoS Ack* object is received from $GoSP_1$, it changes to the *forwarding state* again. The other event that make state changing is receiving from any downstream GoS node an extended Hello message with a GoS Request for a local retransmission. Here the node changes its state to *NMGP search*, to accede to its temporal buffer trying to find the requested packet, according to the characterization information received in the GoS request.

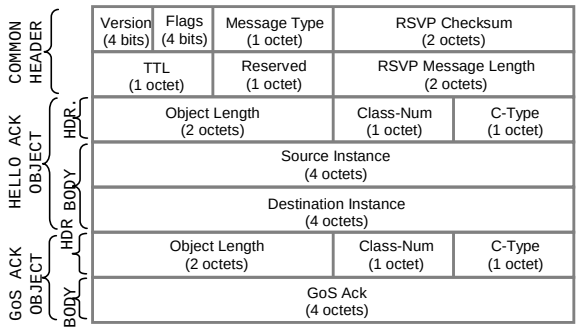

Fig. 4. GoS extended Hello message format, with common Hello Ack and *GoS Ack* objects.

If it finds in NMGP the requested packet, it send a GoS Hello message with a GoS Ack object indicating that packet was found and it will be locally retransmitted. After this, it changes to *local retransmission* state, to get the GoS packet from NMGP and retransmit it. After this it will return to initial *forwarding* state. In case of not find the packet in NMGP buffer, it will send a GoS Ack object indicating that packet was not found, changing to *request of local retransmission* state and sending a GoS Hello message with the GoS request to the next GoSP node, if it is not the last one. This new GoS request message to the next node in GoSP is shorter than previous one, since that if a node does not find the requested GoS packet in the NMGP and it has to request it to next node of GoSP, it first will remove its address of GoS Request object, to simplify the message (see figure 3). So with a bigger used diameter in the plane GoS, the GoS messages to send will be shorter.

4 Analysis and Evaluation of the Proposal

In this section we will show an analysis of GoS benefits in the delay of packets belonging to privileged flows. We consider an MPLS domain $G(U)$ network with a set X of n nodes and a set U of links. Let d_{ij} the delay of link $(x_i, x_j) \in U$ and let $d(x_i, x_j)$ the delay of a path between two nodes x_i and x_j which can be non-neighbors. Our objective is to minimize the delay used by packets when are transmitted between two any nodes of the path $LSP_{i,n}$ of $U(G)$:

$$min\ d(x_i, x_j) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}\ x_{ij}\ , \tag{3}$$

$$\text{subject to: } \sum_{l=2}^{n} x_{1l} = 1\,, \tag{4}$$

$$\sum_{i=1}^{n} x_{il} - \sum_{j=1}^{n} x_{lj} = 0,\ l = 2,3,...,n-1, \qquad (5)$$

$$\sum_{l=1}^{n-1} x_{ln} = 1 \qquad (6)$$

where $d_{i,i} = 0,\ \forall\, i \in N\ ;\quad x_{i,j} = 1,\ \forall (x_i, x_j) \in LSP_{i,n}$ and $x_{i,j} = 0,\ \forall (x_i, x_j) \notin LSP_{i,n}$.

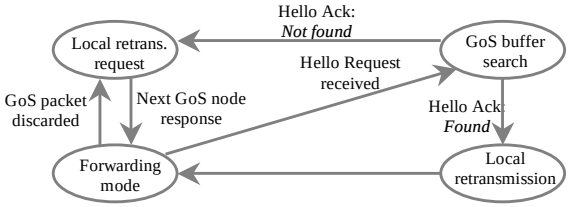

Fig. 5. GoS extended Hello message format, with common Hello Request and *GoS Request* objects.

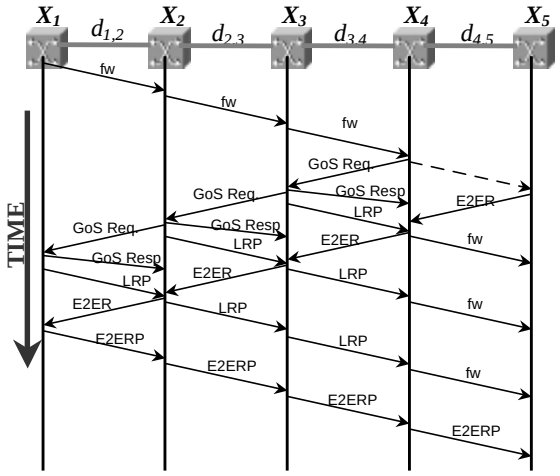

Fig. 6. Operation after a packet discard in intermediate node X_4, using 3 available *GoSP diameters* to get a local retransmission. It is compared with a case of end to end recoveries (*d*: links delay; *Fw*: packet forwarding*; E2ER*: end to end retransmission request time; *LRP*: locally recovered packet; *E2ERP*: end to end recovered packet).

4.1 End to End Retransmissions

Let x_n a non-GoS congested end node. In case of packet discarding by x_n, then function *Discarding Detection Time* (DDT_{e2e}) between two nodes of $LSP_{i,n}$ is:

$$DDT_{e2e}(x_i, x_n) = \sum_{l=i}^{n-1} d_{l,l+1}\, x_{l,l+1} \tag{7}$$

Minimal delay of the end to end (*e2e*) retransmission is:

$$d_{e2e}(x_i, x_n) = 2\sum_{l=i}^{n-1} d_{l,l+1}\, x_{l,l+1} \tag{8}$$

So total delay $\Delta_{e2e}(x_i, x_n)$ to get discarded flow in x_n is got from (7) and (8):

$$\Delta_{e2e}(x_i, x_n) = 3\sum_{l=i}^{n-1} d_{l,l+1}\, x_{l,l+1} \tag{9}$$

4.2 If Congested End Node x_n is GoS Capable

Let x_n a GoS congested end node. In case of packet discarding by x_n, then *Discarding Detection Time* (DDT_d) between source and sink nodes of path $LSP_{i,n}$ is:

$$DDT_d(x_i, x_n) = \sum_{l=i}^{n-1} d_{l,l+1}\, x_{l,l+1} \tag{10}$$

Minimal delay of local retransmission using a *GoSP* with diameter *d* (d_d) is:

$$d_d(x_i, x_n) = 2\sum_{l=n-d}^{n-1} d_{l,l+1}\, x_{l,l+1}\,, \tag{11}$$

$$\text{subject to: } 0 < d < n - i \tag{12}$$

If diameter in Eq. (11) was n-i, then if $l = n–d = n – (n–i) = n – n + i = i$, we get:

$$2\sum_{l=n-d}^{n-1} d_{l,l+1}\, x_{l,l+1} = 2\sum_{l=i}^{n-1} d_{l,l+1}\, x_{l,l+1}\,, \tag{13}$$

i.e., it would be and *e2e* retransmission. Moreover, if in Eq. (11) *GoSP diameter* was bigger than *n-i*, then it would be trying to get a retransmission from a previous node to x_i, but this one is the source of data flow, so it is unfeasible. Thus, total delay $\Delta_d(x_i, x_n)$ to get discarded traffic from start is got from (10) and (11):

$$\Delta_d(x_i, x_n) = \sum_{l=i}^{n-1} d_{l,l+1}\, x_{l,l+1} + 2\sum_{l=n-d}^{n-1} d_{l,l+1}\, x_{l,l+1} \tag{14}$$

At this point we test if (14) < (9):

$$\sum_{l=i}^{n-1} d_{l,l+1}\, x_{l,l+1} + 2\sum_{l=n-d}^{n-1} d_{l,l+1}\, x_{l,l+1} \;<\; 3\sum_{l=i}^{n-1} d_{l,l+1}\, x_{l,l+1} \tag{15}$$

$$2\sum_{l=n-d}^{n-1} d_{l,l+1}\, x_{l,l+1} \;<\; 2\sum_{l=i}^{n-1} d_{l,l+1}\, x_{l,l+1} \tag{16}$$

So according to Eq. (8) and Eq. (11), we only need to verify in Eq. (16) that $d_d(x_i, x_n) < d_{e2e}(x_i, x_n)$. The only condition that distinguishes the members of (16) is the set of values of variable l. We only need to demonstrate that l takes a lesser number of values in $d_d(x_i, x_n)$ than in $d_{e2e}(x_i, x_n)$:

$$\begin{aligned} n - 1 - (n - d) &< n - 1 - i; \\ n - 1 - n + d &< n - 1 - i; \\ -1 + d &< n - 1 - i; \\ -1 + 1 + d < n - i &\Rightarrow d < n - i \end{aligned} \tag{17}$$

We get that the problem is kept in feasibility zone, since Eq. (17) is one of the restrictions of (12). Thus, it has been demonstrated that $\Delta_d(x_i, x_n) < \Delta_{e2e}(x_i, x_n)$. So, Eq. (14) offers delay benefits: Eq. (14) – Eq. (9) > 0, improving (3):

$$\Delta_{e2e}(x_i, x_n) - \Delta_d(x_i, x_n) = 2\sum_{l=i}^{n-d-1} d_{l,l+1}\, x_{l,l+1} \tag{18}$$

4.3 If a Congested Intermediate Node x_{DD} is GoS Capable

Let x_{DD} a GoS congested intermediate node. In case of packet discarding by x_{DD}, then *Discarding Detection Time* (DDT_d) between source and congested node x_{DD} is:

$$DDT_d(x_i, x_{DD}) = \sum_{l=i}^{DD-1} d_{l,l+1}\, x_{l,l+1} \tag{19}$$

Minimal delay of the *e2e* retransmission is:

$$d_d(x_i, x_{DD}) = 2\sum_{l=DD-d}^{DD-1} d_{l,l+1}\, x_{l,l+1}, \tag{20}$$

$$\text{subject to: } 0 < d \leq DD - i, \tag{21}$$

If diameter in Eq. (20) was bigger than $DD - i$, then it would be trying to get a retransmission from a previous node to x_i, but this one is the source of data flow, and this is unfeasible. (In this case, retransmission from source node x_i ($d = DD - i$), brings improvement with respect to *e2e*, because x_{DD} is a previous node to x_n,, i.e.: if $DD < n$ Þ $DD - i < n - i$), so it is a local retransmission. So total delay $\Delta_d(x_i, x_n)$ to get discarded traffic from initial instant of transmission is got from (19) and (20):

$$\Delta_d(x_i, x_n) = DDT_d(x_i, x_{DD}) + d_d(x_i, x_{DD}) + \sum_{l=DD}^{n-1} d_{l,l+1}\, x_{l,l+1} = \\ = DDT_{e2e}(x_i, x_n) + d_d(x_i, x_{DD}) \tag{22}$$

At this point we test if (22) < (9):

$$\sum_{l=i}^{n-1} d_{l,l+1}\, x_{l,l+1} + 2\sum_{l=DD-d}^{DD-1} d_{l,l+1}\, x_{l,l+1} < 3\sum_{l=i}^{n-1} d_{l,l+1}\, x_{l,l+1} \tag{23}$$

Optimizing, we get:

$$2\sum_{l=DD-d}^{DD-1} d_{l,l+1}\, x_{l,l+1} < 2\sum_{l=i}^{n-1} d_{l,l+1}\, x_{l,l+1} \tag{24}$$

So according to Eq. (8) and Eq. (11), again we only need to verify in Eq. (24) that $d_d(x_i, x_n) < d_{e2e}(x_i, x_n)$. As in Eq. (17) we get that the problem is kept in feasibility zone. So Eq. (22) offers delay benefits, i. e., Eq. (22) – Eq. (9) > 0, improving Eq.(3):

$$\Delta_{e2e}(x_i, x_n) - \Delta_d(x_i, x_n) = 2\left(\sum_{l=i}^{DD-d-1} d_{l,l+1}\, x_{l,l+1} + \sum_{l=DD}^{n-1} d_{l,l+1}\, x_{l,l+1}\right) \tag{25}$$

Consider a congested MPLS domain with a path $LSP_{i,n}$. The last n-i nodes are discarding packets. Figure 7 shows a comparative between no congested traffic, *e2e* case (using *e2e* retransmissions) and three cases of local retransmissions (with d=1, d=2 and d=3). In figure 8 a comparative at different time samples is shown. For example, at 3,700 ms only 171 packets have been correctly received in the sink node. In *GoSP diameter*=3 case, sink node has already received 213 packets node, with d=2, 313 packets and in case of using d=1, sink has received 584 packets.

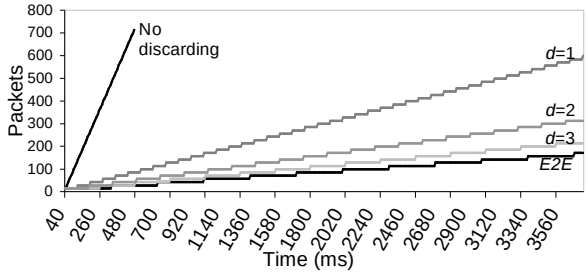

Fig. 7. Throughput comparative between local retransmissions and E2E, with congestion in last n–1 nodes.

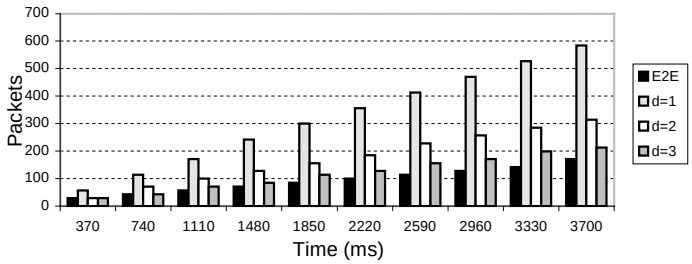

Fig. 8. Comparative between local retransmissions and E2E at different time samples.

5 Conclusion

This work proposes GoS as a traffic local recovery technique in an MPLS domain in order to improve performance of privileged data flows. We have first discussed the requirements for GoS over MPLS. We have then shown that by introducing a limited number of RSVP-TE protocol extensions it is possible GoS signaling to such privileged data flows that require reliability. The proposed technique has been analysed and demonstrated the benefits due to local retransmissions of discarded traffic with respect to end to end retransmissions.

References

1. Taesang Choi.: Design and implementation of an information model for integrated configuration and performance management of MPLS-TE/VPN/QoS, IFIP/IEEE 8th International Symposium on Integrated Network Management (2003) 143-146.
2. Fowler, et al.: QoS path selection exploiting minimum link delays in MPLS-based networks, Proceedings IEEE Systems Communications (2005) 27-32.
3. Suryasaputra, R. Kist, A. A. Harris, R.J.: Verification of MPLS traffic engineering techniques, 13th IEEE International Conference on Networks (2005) 190-195.
4. Dominguez-Dorado, M., et al.: Guarantee of Service (GoS) support over MPLS using Active Techniques, WSEAS Transactions on Communications (2004) 1959-1964.
5. Fowler, S. Zeadally, S.: Priority-based congestion control in MPLS-based networks, Advanced Industrial Conference on Telecommunications. IEEE AICT/SAPIR/ELETE (2005) 332-337.
6. G. Siganos: Powerlaws and the AS-level Internet topology, ACM/IEEE Transactions on Networking (2003), vol. 11, 514-524.
7. RFC3473: GMPLS Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions (2003).

[1] This work is sponsored in part by the Regional Government of Extremadura (Education, Science and Technology Council) under GRANT Nº PDT05A041