A Label Space Reduction Algorithm for P2MP LSPs using Asymmetric Tunnels

F. Solano
Institute of Informatics and Applications.
University of Girona.
Girona, Spain.
fsolanod@eia.udg.es

R. Fabregat
Institute of Informatics and Applications.
University of Girona.
Girona, Spain.
ramon@eia.udg.es

Y. Donoso
Department of Computer Science and Systems Eng.
Universidad del Norte.
Barranquilla, Colombia.
ydonoso@uninorte.edu.co

J.L. Marzo
Institute of Informatics and Applications.
University of Girona.
Girona, Spain.
marzo@eia.udg.es

***Abstract* - The aim of Traffic Engineering is to optimise network resource utilization. Although several works on minimizing network resource utilization have been published, few works have focused on LSR label space. This paper proposes an algorithm that uses MPLS label stack features in order to reduce the number of labels used in LSPs forwarding. Some tunnelling methods and their MPLS implementation drawbacks are also discussed. The algorithm described sets up the NHLFE tables in each LSR, creating asymmetric tunnels when possible. Experimental results show that the algorithm achieves a large reduction factor in the label space. The work presented here applies for both types of connections: P2MP and P2P.**

***Keywords* – Asymmetric tunnels, label space, label stack, label space reduction, longest segment first, NHLFE, traffic engineering, MPLS.**

I. INTRODUCTION

Traffic engineering (TE) aims at improving the performance of operational networks usually taking into account quality of service (QoS) requirements. The main objectives are to reduce congestion hot spots, improve resource utilization and provide adequate QoS for final users. These aims can be achieved by setting up explicit routes through the physical network in such a way that the traffic distribution is balanced across several traffic trunks, giving the best possible service [1].

Multi Protocol Label Switching (MPLS) works with these TE schemes by setting up label switched paths (LSPs) when needed to transmit the customer flows of Internet Service Providers (ISP) efficiently and according to their requirements. Customer requirements are flow dependent, i.e. delay, packet loss, jitter, etc. Although this can be achieved in many ways using different algorithms [2], ISPs must be aware of the label switched router's (LSR) internal resource utilization, such as the label space.

Each time an LSP is established, all the LSR that belong to it must use a label in order to identify the LSP transiting it. Therefore, every LSP packet must carry this label encoded inside it when it arrives at the LSR. When a packet is received by an LSR, the LSR must look for the packet label and then search for a Next Hop Label Forwarding Entry (NHLFE) that refers to this label in order to decide which interface to use to reach the next hop in the network [3]. Clearly, the more LSPs an LSR supports, the more NHLFEs exist.

So far, we have identified 4 reasons to reduce the label space of MPLS LSRs: First, each label must be encoded in a 20-bits field [3], which only allows 2^{20} (1.048.576) different possible labels in an LSR: a large, but finite number. Second, despite this being a sufficiently high number for label encoding in a single LSR, large NHLFE could cause long delays while an LSR finds the next hop LSR from its forwarding table each time a packet is received. Therefore, a smaller forwarding table will reduce LSR memory requirements and aid LSR to forward packets faster ([4], [5] and [6]). Third, when we consider Multi Protocol Lambda Switching (MPλS), we see that this problem is on a larger scale. The MPLS label space is comparatively large (one million per port), whereas there is a relatively limited number of lambda channels (tens to hundreds per port today, scaling to thousands in the next few years). The growing interest among several of the largest service providers (such as AT&T) to use MPLS to provide Virtual Private Network (VPN) services led us to a fourth reason. That is, to offer an MPLS-based VPN service to thousands of customers, ISPs would need to handle thousands of MPLS LPS connections at the VPN endpoints (this is especially true for Layer 2 MPLS VPNs and VPN services based on the overlay model) [4].

To support LSP tunnelling and forwarding, MPLS defined a label stack for packets [7] and some stack operations set inside NHLFEs [3]. These operations are: a) replace the label at the top with a new one (label swapping), b) pop the stack, c) replace the label at the top with a new one and then push one or more onto the stack. See figure 1 for an example.

Although IETF have not yet decided how to set up Point-to-MultiPoint (P2MP) LSPs in MPLS, this paper proposes an algorithm that uses the MPLS label stack in a different way to

This work was partially supported by the MCyT in the project TIC2003-05567 and the COST 293 project.

The work of Yezid Donoso was supported by the Universidad del Norte (Colombia) under contract G01 02-MP03-CCBFPD-0001-2001.

The work of Fernando Solano was supported by the Ministry of Universities, Research and Information Society (DURSI) of the Government of Catalonia under contract 2005FIR-00379.

reduce the label space and hence improve the way MPLS uses NHLFE in P2MP LSPs. The terminology used in this paper is the one established in [8].

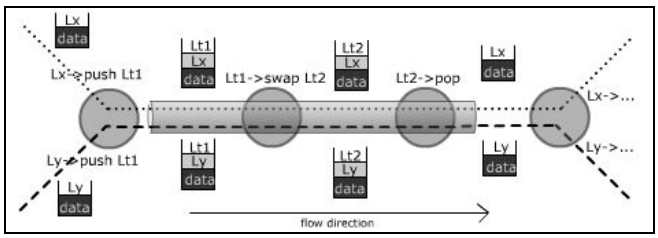

Fig 1. Tunnelling in MPLS networks

This work has been organized as follows: some studies about label space reduction and label stack size are discussed in section II. Several label space reduction techniques are explained in section III together with the asymmetric tunnel concept. An asymmetric tunnelling algorithm for P2MP LSPs using an MPLS label stack is described in section IV. Section V shows some simulation results with different topologies and randomly generated P2MP requests. Finally, conclusions and further studies are presented in section VI.

This work is focused on P2MP connections, as they can be seen as a general version of Point to Point (P2P) connections; thus, the work presented here applies for both types of connections. A previous version of this work appeared in [9].

II. LABEL SPACE REDUCTION METHODS

The MPLS architecture allows *label merging* in P2P LSPs. Label merging reduces the number of labels that are needed to handle a particular set of flows, and may also reduce the amount of label distribution control traffic needed [3].

With regards to label merging, [3] refers to a MultiPoint-to-Point tree (MP2P) created by 'joining' many P2P LSPs, i.e. a tree rooted at an egress LSR with ingress LSRs as leaves. In other words, if two P2P LSPs follow the same path from an intermediate LSR to the egress LSR, this method allocates the same label to both P2P LSPs and thus reduces the number of labels used. In this case, labels assigned to different incoming links are joint into one label assigned to an outgoing link. Fig. 2 shows different P2P connections in which a single MP2P is established between three ingress LSRs {N1, N5, N8} and the egress LSR N11.

In the downstream to upstream label assignation process [3], i.e. the downstream LSR assigns its outgoing label to the previous LSR (upstream LSR), N9 confers the same label L1 to N8 and N6 (see section 3.14 of [3] for more information).

In [5], [6], [10] and [11], algorithms that find P2P LSPs which can be merged into a minimal number of MP2P LSPs are considered. Reference [11] proves an upper bound of N (number of nodes) + M (number of links) for the label space. Note that in these works ([5], [6], [10] and [11]), minimising the label space is a base criterion for finding the LSP's routes. As the ISP's customer's requirements are often measured as QoS requirements (flow dependent), we think that the label space should not be considered as an objective function (at most a model restriction) in any optimisation model that deals with finding the LSP's routes.

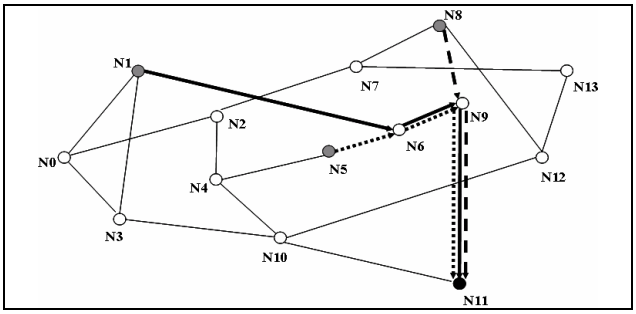

Fig 2. Several P2P connections merged into a single MP2P.

In [4], they make a comprehensive study of label size versus stack depth trade-off for MPLS routing protocols in P2P connections. They show that, in addition to LSP tunnelling, label stacks can also be used to reduce the number of labels needed to set up LSPs in a network using a special coding technique. They also proved some label space upper bounds under certain types of conditions. Gupta, Kumar and Rastogi [4] inferred a lower and upper bound for two basic problems: (1) FIXED STACK ROUTING: minimize the number of labels used when a bound on the stack depth is fixed, and (2) FIXED LABEL ROUTING: minimize the stack depth in P2P connection when a fixed number of labels is set for the network. However, they didn't propose an algorithm to set up the LSR forwarding table using the label stack when a set of LSPs are given.

It should be pointed out that so far we have not found an algorithm that *only* sets up LSR forwarding tables (i.e. no path finding algorithm) in order to minimise the number of labels by using the label stack in P2MP connections. Moreover, to date there is no literature about the novel asymmetric tunnel concept.

III. ASYMMETRIC TUNNELS AS A LABEL SPACE REDUCTION METHOD

To illustrate these label space reduction methods (label merging and label stacking), suppose that $P2MP_1$ and $P2MP_2$ are two trees (see Fig. 3) that can be established in the NSF network.

As $P2MP_1$ and $P2MP_2$ have equal sub-P2MP trees starting at N10 and ending at {N8, N13} through N12, the label merging scheme can be used and therefore a single label is needed in this sub-P2MP tree. Although {N0→N3→N10} is a path used by both P2MP trees, the previous reduction scheme cannot be used here because it will cause either N10 to forward $P2MP_2$ packets to N11 (i.e. packet duplication), or N10 to stop forwarding $P2MP_1$ packets to N11 (i.e. multicast incomplete replication). To reduce the label space the label stacking scheme can also be used. In this case, N0 can push a label into the $P2MP_1$ and $P2MP_2$ packet stacks and this label can be popped when the packets reach N10. Using these two

reduction methods, the total number of labels in the network is reduced from 13 to 9.

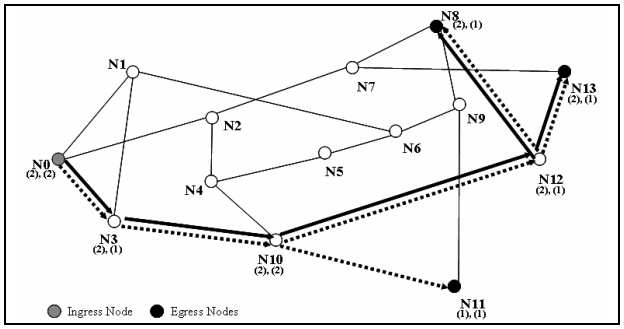

Fig 3. Two P2MP merged at N10-N12-{N8, N13} and "stacked" at N0-N3-N10.

The LSR tables of the example above can be summarised as follows (Table I) when both reduction methods are applied.

Since the labels are assigned upstream, it is the number of incoming labels per LSR that should be minimised, e.g. the number of incoming labels for N3, N11, N12 is 1 and for N10 there are 2 incoming labels. Moreover, note that as the number of incoming labels is reduced, the number of NHLFEs is also reduced.

TABLE I
ACTIVE LSR NHLFES FOR FIGURE 3

LSR	Incoming Label	Outgoing LSR	Operation
N0	$P2MP_1$	N3	Push L1, L0
	$P2MP_2$	N3	Push L2, L0
N3	L0	N10	Pop
N10	L1	N11	Swap L4
		N12	Swap L3
	L2	N12	Swap L3
N11	L4	N11	*Pop*[1]
N12	L3	N8	Swap L5
		N13	Swap L6

We will refer to a P2MP configuration as a set of P2MP LSPs that should be configured in a given topology. The problem of finding a near-optimal label space reduced solution is not trivial since it can be achieved in many ways.

We will first discuss branch nodes since they are discarded as a tunnel member in our solution. Each time a branch LSR needs to forward a packet, in order to assure P2MP LSP consistency, the LSR needs to swap the label of the incoming packet with the incoming labels of the downstream LSRs more than once. Because an LSR cannot swap a label that is not at the top of the stack, branch LSRs cannot be members of any tunnel since they cannot assign the correct label to the stacked LSPs of a tunnel in the replication process.

As an example, consider P2MP configuration in figure 4 with 2 P2MP LSPs and the weak P2MP tunnel in figure 5 which stacks both P2MP LSPs. Without a tunnel, figure 4, LSR N10 will replace an incoming label by two different outgoing labels in order to assure correct packet forwarding to N12 and N11.

In figure 5, LSR N10 forwards both tunnelled P2MP LSPs by swapping the top label but not the stacked label, hence LSR N13 and LSR N9 should receive packets with the same labels, e.g. Lx and Ly in figure 5.

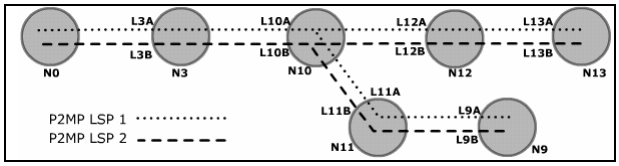

Figure 4. P2MP configuration.

TABLE II
ACTIVE LSR NHLFES FOR FIGURE 4

LSR	Incoming Label	Outgoing LSR	Operation
N0	$P2MP_1$	N3	Swap L3A
	$P2MP_2$	N3	Swap L3B
N3	L3A	N10	Swap L10A
	L3B	N10	Swap L10B
N10	L10A	N11	Swap L11A
		N12	Swap L12A
	L10B	N11	Swap L11B
		N12	Swap L12B
N11	L11A	N9	Swap L9A
	L11B	N9	Swap L9B
N12	L12A	N13	Swap L13A
	L12B	N13	Swap L13B
N9/13	…	…	*Pop*[2]

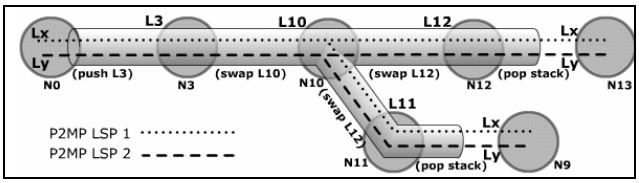

Fig 5. P2MP unfeasible tunnelling solution.

TABLE III
ACTIVE LSR NHLFES FOR FIGURE 5

LSR	Incoming Label	Outgoing LSR	Operation
N0	$P2MP_1$	N3	Push L3,Lx
	$P2MP_2$	N3	Push L3,Ly
N3	L3	N10	Swap L10
N10	L10	N11	Swap L11
		N12	Swap L12
N11	L11	N9	Pop
N12	L12	N13	Pop
N9/13	Lx/Ly	…	*Pop*[2]

[2] These *pop* operations can be omitted for *egress* LSRs, if necessary, but they are included for compatibility with *bud* LSRs in P2MP LSPs.

Unless the architecture is changed so that many downstream LSRs agree about their incoming label, this weakness leads us to consider only P2P branches in the tunnels. To find an easier explanation of other tunnelling techniques, we look at 3 P2MP LSPs (see figure 5). All P2MP LSPs forward packets from N0. In a common MPLS NHLFE set up procedure, each LSR allocates space in the memory for 3 incoming labels, as no label space reduction scheme is considered.

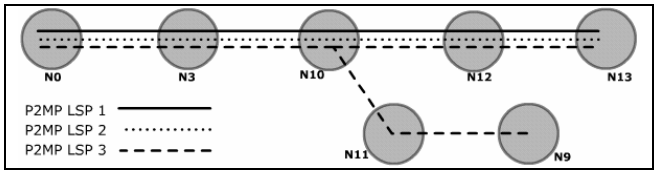

Fig. 6. A P2MP configuration that can be stacked in many ways.

It is clear that the sub-P2MP tree {N10→N12→N13} is the same from here on and therefore can use a single label for the P2MP configuration. The preceding sub-P2MP tree {N0→N3→N10} cannot use the same label because N10 needs to multicast packets for LSP3 through the sub-P2MP tree {N10-N11}. This leads us to an initial solution where the sub-P2MP tree {N0→N3→N10} can be stacked and the sub-P2MP tree {N10→N12→N13} can be merged (figure 7). Despite the fact that N10 and N12 use a single label to forward the P2MP configuration, N10 still uses 3 incoming labels for the configuration.

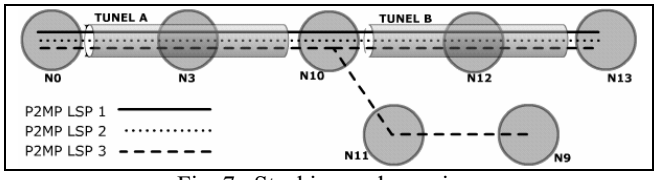

Fig. 7. Stacking and merging.

A more complex and efficient solution can be contemplated if nested tunnels are considered, i.e. tunnels within another tunnel. Figure 8 shows a solution where LSP1 and LSP2 are stacked across the entire topology and LSP3 is stacked in the sub-P2MP trees {N0→N3→N10} and {N10→N12→N13}. Unfortunately this solution can not be set in MPLS because it does not allow *multiple popping* of stacked labels in a NHLFE [3], i.e. popping of more that one label by an LSR, as N3 and N12 need to do.

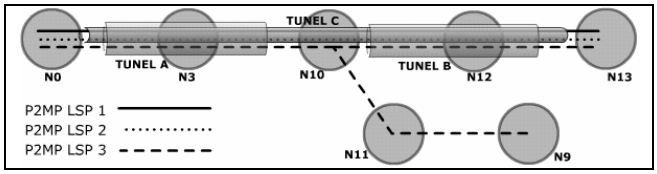

Fig. 8. Solution involving nested tunnels.

A similar MPLS solution that solves this drawback can be regarded as an asymmetric tunnel (in figure 9). The asymmetric tunnel concept comes from the idea that *not all* the stacked LSPs are tunnelled along all LSRs.

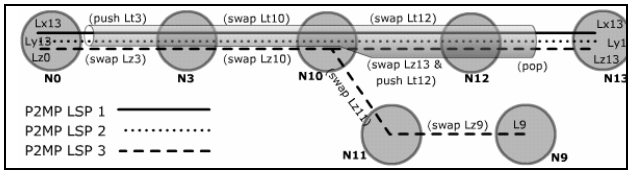

Fig 9. Asymmetric tunnel.

TABLE IV
ACTIVE LSR NHLFES FOR FIGURE 9

LSR	Incoming Label	Outgoing LSR	Operation
N0	$P2MP_1$	N3	Push Lt3
	$P2MP_2$	N3	Push Lt3
	$P2MP_3$	N3	Swap Lz3
N3	Lt3	N10	Swap Lt10
	Lz3	N10	Swap Lz10
N10	Lt10	N12	Swap Lt12
	Lz10	N11	Swap Lz11
		N12	Swap Lz13 & Push Lt12
	L11B	N9	Swap L9B
N12	Lt12	N13	Pop
N13	Lx13, Ly13, Lz13	…	*Pop*†
N9/N11	…	…	*Pop*†

In the example, N10 stacks LSP3 by pushing the same label that N0 pushed before to LSP1 and LSP2. Here, N3 and N10 use 2 incoming labels, but N12 only uses 1. Since there is no way an LSR can look at labels behind the top, all LSPs must be *unstacked* at the same time and therefore, asymmetric tunnels will usually be 'bigger' at the end than at the start.

In the next section we present an algorithm that makes tunnels in a P2MP configuration by selecting the longest P2P branch.

IV. AN HEURISTIC: THE LONGEST SEGMENT FIRST ALGORITHM

Consider a P2MP configuration as a set *P2MP* of P2MP LSPs. For a $m \in P2MP$ consider a P2P decomposition d(m): $(m \in P2MP \rightarrow \{u(i,j) \in P2P\})$ in which each element $u_{(i,j)}$ is a P2P LSP that connects a subset of LSRs of the P2MP LSP starting at LSR i (an ingress LSR, bud LSR or branch LSR) and ending at LSR j (an egress LSR, bud LSR or branch LSR). In Fig 2, the P2MP LSP that connects the ingress node N0 with egress nodes N11, N8 and N13 can be decomposed in 5 P2P LSP: $u_{(0,10)}$, $u_{(10,11)}$, $u_{(10,12)}$, $u_{(12,8)}$ and $u_{(12,13)}$. It is clear that this decomposition is unique and easy to find.

Let $|u(i,j)|$ be the number of LSRs that u(i,j) uses to forward the information. The intersection of two P2P LSPs, $u_{(i1,j1)}$ and $u_{(i2,j2)}$, is the longest $u_{(i,j)}$ contained in both. For example, in figure 1, $u_{(1,11)} \cap u_{(5,11)} = u_{(6,11)}$.

The difference between two P2P LSPs, $u_{(i1,j1)}$ and $u_{(i2,j2)}$ where $u_{(i1,j1)} \subseteq u_{(i2,j2)}$, are two sub-P2P LSPs: one starting at i_2 and ending at i_1 and the second starting at j_1 and ending at j_2.

Table V uses the notation explained before to describe a procedure to find a set of sub-P2P LSPs, $T \subseteq P2P$, that can be tunnelled with a single label.

TABLE V
ALGORITHM TO FIND P2MP LSP TUNNELS.

1	$U = \{u_{(a,b)} \mid \forall m \in P2MP, u_{(i,j)} \in \cup d(m), \lvert u_{(a,b)} \rvert \geq 3\}$, and $W = \phi$
2	Find a P2P tunnel $t_{(i,j)}$ such that $\max \lvert t_{(i,j)} \rvert, t_{(i,j)} = u_{(a_1,b_1)} \cap u_{(a_2,b_2)}, u_{(a_1,b_1)} \in U, u_{(a_2,b_2)} \in U$
3	If this tunnel cannot be found, stop.
4	Let $U' = \phi$
5	For each $u_{(a,b)} \in U$ do
6	Find $u'_{(k,j)} = u_{(a,b)} \cap t_{(i,j)}$
7	If $\lvert u'_{(k,j)} \rvert < 3$ then
8	$U' = U' \cup \{u_{(a,b)}\}$
9	Else
10	$W = W \cup \{(t_{(i,j)}, u_{(a,b)})\}$
11	$U' = U' \cup \{u_{(a,b)} - u'_{(k,j)}\}$
12	End if
13	Repeat
14	Let $U = U'$
15	Repeat from 2

In line 1, a set named U is created which contains all the P2P LSPs that are part of any P2MP LSP decomposition. Because a tunnel cannot be made with less than 3 LSRs, each P2P LSP in U should satisfy this constraint. U is our working set. W is a mapping set of tunnelled P2P LSPs that are initially empty.

Line 2 finds a maximum length P2P LSP, $t_{(i,j)}$, that intersects at least two P2P LSPs in U. The algorithm iterates only if this tunnel is found. In all iterations, a new working set U' is computed because each tunnel found *stacks* several P2P LSPs in U. Line 4 initialises this set as empty.

To compute the new working set, each P2P LSP in U is regarded in order to see whether it can be *stacked* using this tunnel or not. A P2P LSP $u_{(a,b)}$ can be stacked using this P2P LSP tunnel $t_{(i,j)}$ if both LSPs intersect in more than 3 LSRs, $u'_{(k,j)}$. In order to assure asymmetric tunnels, the intersected LSP should include the last LSR of the tunnel $t_{(i,j)}$. Line 8 saves $u_{(a,b)}$ in the new working set if no intersection can be found. Otherwise, in line 11 sub-P2P LSPs that are not intersected are included in the new working set and $u_{(a,b)}$ is included in W as part of the algorithm response.

Note that to build a tunnel, the penultimate LSR, $j - 1$, must do a POP in the stack, the first LSR, i, must do a PUSH of two labels and all intermediates LSRs, from $i + 1$ to $j - 2$, must do a SWAP.

V. EXPERIMENTAL RESULTS

The algorithm in the preceding section was tested in two topologies with several P2MP configurations. The network topologies used are square-like, in which each node is placed at the cross-points of a rectangular grid with X rows and Y columns and each node at position (x,y) in the grid has connecting links to the next column (x,y+1), the upper row (x+1,y), and the lower row (x-1,y) when possible. In one of the tested network topologies X=5 and Y=10 (50 nodes x 125 links). In the other tested network topology X=10 and Y=10 (100 x 270).

For each experiment in both topologies, a set of randomly generated P2MP LSPs was created. Each P2MP LSP connects an ingress LSR with a set of 5 egress LSRs. The ingress LSR is chosen randomly from the first 5 LSRs in the grid. The 5 egress LSRs are selected randomly from the last 10 LSRs in the grid. Once the ingress and egress LSRs are picked, the tree is built by selecting a random path for each egress LSR. Finally, redundant segments are deleted.

To evaluate the performance of the solution presented here, it needs to be noted that the number of labels will increase as the network load increases, which is measured as the number of P2MP LSPs in the network. Each time a simulation was ran, the reduction factor was computed as the relationship between the number of labels using tunnels that were dropped-off and the number of labels not using tunnelling.

Figure 10 shows the reduction factor for both topologies.

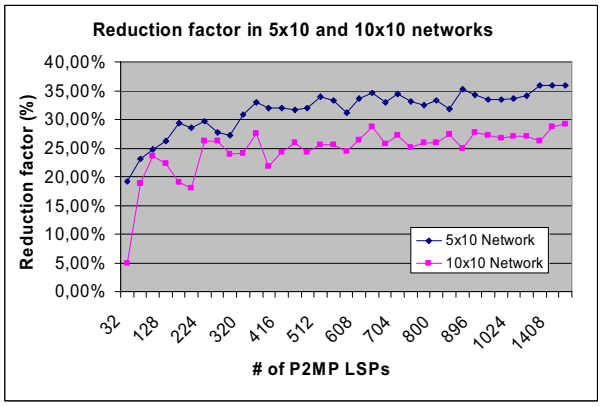

Fig. 10. Reduction factor in two tested networks.

Both topologies reached a stable point when approximately 400 P2MP LSPs were considered. Since the 10x10 network is larger that 5x10 network and there are more ways to reach a destination, the number of intersected LSPs is less and hence each tunnel deals with less P2MP LSPs. In the 5x10 network the reduction factor reached 32.5% when it became stable. In the 10x10 network this factor was about 27.5%. Therefore, the reduction factor depends on the network topology, the P2MP configuration, and the network load.

In addition, the proposed method was simulated over a random generated network generated following M. Faloutsos, P. Faloutsos and C. Faloutsos power laws [12]. The generated network consists of 50 LSRs and 150x2 physical links (rank exponent around -0.75), which could model perfectly a big ISP MPLS core network

In this case, the multicast destination node set is selected randomly from all possible LSRs in the generated network.

A multi-objective optimisation algorithm based on SPEA2 [13] was used to compute the 'best' ways (a set of feasible solutions) to set up each P2MP configuration in the generated network. The SPEA2 based algorithm used, called Generalized Multi-objective Multi-tree Model (GMM-model), can be seen in [14].

Since for each test carried out the GMM algorithm computes a set of feasible solutions to accommodate all flows, the mean of the entire label space reduction factor for each solution in the set is taken.

The following graph illustrates the reduction factor experienced when the network load was varied from 6.25% to 96.88%, with an added tendency line. It is easy to see that the reduction factor follows a logarithmic curvature.

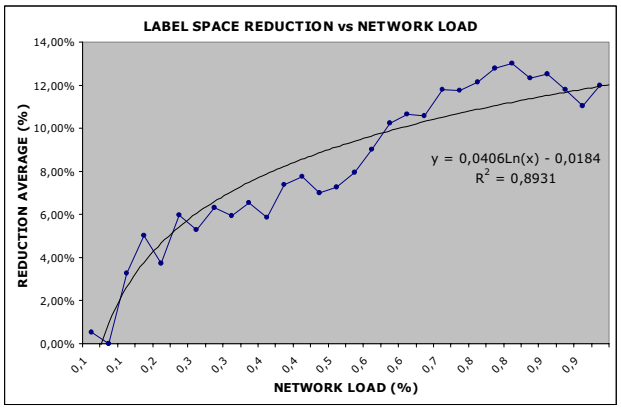

Fig 11. Reduction factor for a real situation

The reduction factor reaches a label space reduction stable state of 11% when the network approaches a load of 60%.

VI. CONCLUSIONS AND FURTHER STUDIES

The work presented here states that the number of labels used can be dropped dramatically using tunnels in a P2MP configuration. It needs to be taken into account that there are many ways to make tunnels in P2MP connections but the best methods are far from being feasible implementations with regards to the current IETF standard. Despite this fact, the novel asymmetric tunnel concept has been discussed and tested in some network topologies with several P2MP configurations using the *longest segment first* algorithm, with satisfactory results. The simulation of this algorithm showed that the reduction factor is dependent on the network topology, the P2MP configuration, and especially on the network load.

Asymmetric tunnel solutions, described here, did not take into account the label merging feature in P2P connections. Moreover, P2MP label merging, which is beyond the scope of this paper, has a good reduction factor. In further work it would be possible to merge these ideas with the one presented here in order to achieve better results. Since there are many ways to create asymmetric tunnels, it would also be beneficial to find an optimisation model which finds the best way to create asymmetric tunnels. This is our next goal. In addition, an algorithm to create tunnels for on-line requests could be considered for future study as an extension for RSVP-TE P2MP [15].

REFERENCES

[1] Z. Wang. Internet QoS: Architectures and Mechanisms for Quality of Service. Morgan Kaufmann Publishers. ISBN 1-55860-608-4.

[2] Y. Donoso, R. Fabregat, F. Solano and J.L. Marzo. "Generalized Multiobjective Multitree model for Dynamic Multicast Groups". IEEE ICC 2005.

[3] E. Rosen, A. Viswanathan, R. Callon. "Multiprotocol Label Switching Architecture". RFC 3031. January 2001.

[4] A. Gupta, A. Kumar, R. Rastogi. "Exploring the trade-off between label size and stack depth in MPLS Routing". IEEE INFOCOM 2003.

[5] S. Bhatnagar, S. Ganguly, B. Nath. "Creating Multipoint-to-Point LSPs for Traffic Engineering". IEEE Communications Magazine, January 2005.

[6] C. Neophytou, C. Phillips. "A Scheme for the Dynamic Formation of Robust Multipoint to Point LSPs". IEEE CCNC 2004. Las Vegas.

[7] E. Rosen, et al. "MPLS Label Stack Encoding". RFC 3032. January 2001.

[8] S. Yasukawa, et al. "Signaling Requirements for Point to Multipoint Traffic Engineered MPLS LSPs". IETF Draft. December 2004.

[9] F. Solano, Y. Donoso, R. Fabregat, J.L. Marzo. "Asymmetric Tunnels in P2MP LSPs as a Label Space Reduction Method". IEEE ICC 2005. Seoul, Korea.

[10] H. Saito, Y. Miyao, M. Yoshida. "Traffic engineering using multiple multipoint-to-point LSPs". IEEE INFOCOM 2000

[11] D. Applegate, M. Thorup. "Load optimal MPLS routing with N+M labels". IEEE INFOCOM 2003.

[12] M. Faloutsos, P. Faloutsos and C. Faloutsos. "On power-law relationships of the Internet Topology". SIGCOMM 1999.

[13] E. Zitzler and L. Thiele. "Multiobjective Evolutionary Algorithm: A comparative case study and the Strength Pareto Approach". IEEE Trans. Evolutionary Computation. Vol. 3, No. 4, 1999.

[14] B. Barán, R. Fabregat, Y. Donoso, F. Solano, J.L. Marzo. "Generalized Multi-objective Multi-tree model", IIiA Research Report, ref. IIiA 04-08-RR, Institute of Informatics and Applications, University of Girona, September 2004.

[15] R. Aggarwal, D. Papadimitriou, S. Yasukawa. "Extensions to RSVP-TE for Point to Multipoint TE LSPs". November 2004.