This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2008 proceedings.

All-Optical Label Stacking: Easing the Trade-offs Between Routing and Architecture Cost in All-Optical Packet Switching

Fernando Solano*, Ruth Van Caenegem†, Didier Colle†, José L. Marzo*,
Mario Pickavet†, Ramón Fabregat* and Piet Demeester†
*Broadband Communication & Distributed Systems - Department of Electronics, Informatics and Automatics (EIA)
University of Girona - IIiA, Girona, Spain.
{fsolanod,ramon,marzo}@eia.udg.es
†INTEC Broadband Communication Networks - Department of Information Technologies (INTEC)
University of Ghent - IBBT - IMEC, Ghent, Belgium.
{ruth.vancaenegem,didier.colle,mario.pickavet,piet.demeester}@intec.ugent.be

***Abstract*—All-Optical Label Swapping (AOLS) forms a key technology towards the implementation of All-Optical Packet Switching nodes (AOPS) for the future optical Internet. The capital expenditures of the deployment of AOLS increases with the size of the label spaces (i.e. the number of used labels), since a special optical device is needed for each recognized label on every node. Label space sizes are affected by the way in which demands are routed. For instance, while shortest-path routing leads to the usage of fewer labels but high link utilization, minimum interference routing leads to the opposite.**

This paper studies All-Optical Label Stacking (AOLStack), which is an extension of the AOLS architecture. AOLStack aims at reducing label spaces while easing the compromise with link utilization. In this paper, an Integer Lineal Program is proposed with the objective of analyzing the softening of the aforementioned trade-off due to AOLStack. Furthermore, a heuristic aiming at finding good solutions in polynomial-time is proposed as well. Simulation results show that AOLStack either a) reduces the label spaces with a low increase in the link utilization or, similarly, b) uses better the residual bandwidth to decrease the number of labels even more.

***Index Terms*—All-Optical Label Switching, All-Optical Packet Switching, All-Optical Label Swapping, All-Optical Label Stacking, label stacking, label merging, GMPLS, MPLS, label space reduction.**

I. Introduction

The current economic expansion drives telecommunication networks that are at the heart of our information-based society to grow steadily and become more and more flexible. Nowadays these networks are mostly based on optical fiber for information transmission. Wavelength Division Multiplexing (WDM) offers enormous bandwidth through parallel high bit-rate wavelength-channels onto the same fiber link. Information is sent over one of these wavelength-channels and reaches its destination following successive fiber links on its itinerary. The term *lightpath* is used to denote this fully optical route followed by the traffic. Although lightpaths offer big capacities (wavelength granularity), they remain relatively static and do not offer always the flexibility that is asked for by network operators (i.e. different types of traffic resulting from a wide range of protocols and services) [1]. On the other hand, packet switching networks are best suitable considering these aspects since they provide finer bandwidth granularity [2].

The first packet switched networks employing fiber links needed to entirely bring optical packets back to the electronic domain at every hop to achieve routing. Despite the flexibility (and backwards compatibility) of this approach, the enormous number of packet "handlings" makes it badly scalable, demanding costly Optical Electronic Optical (OEO) conversions at high-speed. More recent technologies, namely Optical Packet Switched (OPS) networks, overcome these problems partially by keeping the packet's payload optically during the whole itinerary. Only the packet's header is brought back to the electronic domain for analysis. But, even if OEO conversions can be performed at high speed, there is still the gap between fast fiber transmission speeds and slow router decision-making-speeds (due to table look-up procedures that are time consuming) [1].

Therefore, research has started focussing nowadays on All-Optical Packet Switching (AOPS) nodes that intend to route packets at wavelength-speed without converting them to the electronic domain. Providing the AOPS functionality avoids electronics in the data-forwarding plane (at the core nodes), making the usage of electronics in the control plane solely.

Future networks are believed to be user-centric and allow on-demand and user-defined broadband provisioning [3]. This implies a shift in the broadband bandwidth demand from leased wavelengths over lightpaths to wavelength routed bursts and packets. In addition to the aforementioned technological advantages, this fact motivates the research community to dig into a robust, flexible and fast AOPS architecture.

Even though packet switching granularity is needed, a connection-oriented control plane is desirable in order to provide service guarantees. The most promising control plane technology for the future optical Internet is the Generalized MultiProtocol Label Switching (GMPLS) [4]. Considering an underlying packet switch technology, GMPLS tags packets with labels. In this way, packets belonging to the same demand can be identified and forwarded accordingly, creating a Label

978-1-4244-2026-1/08/$25.00

Switched Path (LSP).

In order to achieve all-optical packet switching, GMPLS forwarding functionality must be implemented in optical hardware, *viz.* burned-in-chip [5]. An implementation of GMPLS forwarding functionalities using optical hardware is known as All-Optical Label Swapping (AOLS). However, AOLS designs do not scale well [6]. This is because each label the node recognizes needs an individual hardware [7]. Solutions to this are: a) the reduction of the number of labels needed to route all packets in the network or, b) architectures and mechanisms that share the label recognition hardware among different labels. This paper concerns solution a) by considering the principles of *label merging* and *label stacking* of GMPLS. To the best of our knowledge, this is the first time that both label merging and label stacking are applied to AOLS networks.

The total number of labels used in a network depends on the routes taken by the demands. On the one hand, shortest-path algorithms would aim at short routes, but also to a high link utilization. On the other hand, minimum interference algorithms would aim at low link utilization, but longer routes [8]. In the context of AOPS, long routes lead to the usage of more labels (increasing capital expenditures) and high link utilization leads to not only call-blockings but also to packets loss. This trade-off between the number of used labels (or label space) and the Maximum Link Utilization (MLU) can be eased by using GMPLS label merging and/or label stacking. This paper concerns the analysis of this softness in the trade-off considering stacks of two labels at most.

In order to perform our analysis, an Integer Lineal Program (ILP) is proposed, which finds the optimal label assignation (considering stacking) given a set of demands and a limit on the MLU. Furthermore, an heuristic tackling the same problem is proposed as well. Considering the label space reduction problem in GMPLS networks, our paper contributes with the novel *MERLIN group* concept; aiming at simplifying the way in which label reduction through label merging is perceived.

The remaining of this paper is organized as follows. In §II, the AOLS architecture is summarized including an extension to implement All-Optical Label Stacking (AOLStack). In §III, the state of the art in label space reduction in GMPLS is summarized. Furthermore, the routing trade-offs are briefly shown. In §IV and §V, we propose an ILP and an heuristic, respectively, to tackle the problem. Simulation results and conclusions are described in the last two sections of this paper, *viz.* §VI and §VII respectively.

II. All-Optical Label Switching

We initially describe the Lasagne implementation of AOLS [7]. At the end of this section, we describe the enhancement that allows Lasagne perform label stacking, AOLStack.

A. The All-Optical Label Swapping Block

We must take into account that the wavelengths entering an AOPS node must be first demultiplexed and for each wavelength an AOLS-block is required.

The basic all-optical functionalities of AOLS are: a) label identification, b) label insertion and, c) packet routing based on the header's label. The original AOLS-block proposed by the Lasagne project [6], [7] can be seen in Fig. 1.

Each AOLS-block comprehends the true forwarding functionality of incoming packets. Entering the AOLS-module, the packet payload (40 Gbit/s) is separated from its label [9]. The extracted label is fed to a bank of All-Optical Logical XOR Gates (AOLXG) [7], or simply *correlators*, where the comparison between the incoming label and a set of local labels is performed. These local labels are generated using a network of optical delay lines (ODLs). An ODL is comprised of a set of interconnected Fibre Delay Lines (FDL), couplers and splitters which generate a bit sequence out of one pulse. *Thus, comparing the incoming label to the local labels implies that for each possible incoming label a separate ODL and a correlator need to be installed in the AOLS-block*. After comparison, a high intensity pulse will appear at the output of the correlator with the matching label. This pulse feeds a control-block that drives a wavelength converter, setting the packets to the appropriate wavelength.

Meanwhile a new label is generated in the New Label Generation (NLG) sub-block using the appropriate ODL. The new label is inserted in front of the payload and both the payload and the new label are now converted to the wavelength defined by the AOFF [7]. The packet is then sent through an Arrayed Waveguide Grating (AWG): thus the wavelength on which the packet leaves the AOLS-block determines the outgoing port on which the packet leaves the node. Because of lack of space, we address the reader to [7] for more details.

In [7], a slight modification of the architecture allowing *label stripping* is proposed as well. With label stripping, the ingress node codes a stack of labels in the header and, at every hop the top label is stripped off (extracted, but not replaced). With label stripping, the overall number of recognized labels equals the number of links in the network, but the depth of

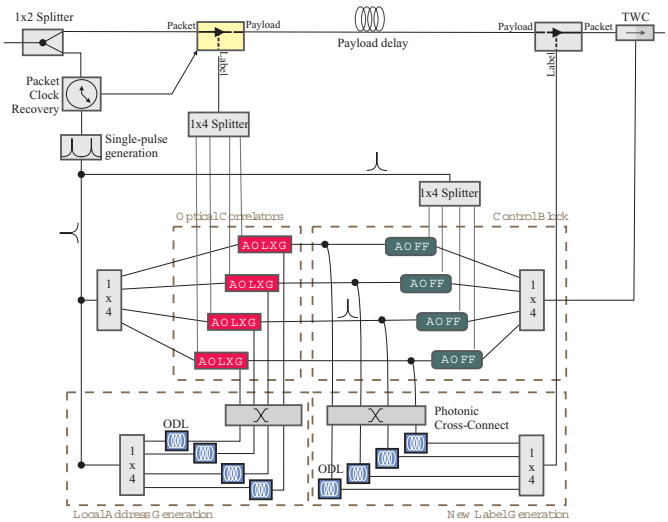


Fig. 1: AOLS implementation according to the Lasagne project

the stack equals the maximum length of the paths, increasing the overhead in the traffic [6].

B. Enhancements for All-Optical Label Stacking

Allowing all-optical label stacking imposes two challenges: a) considering only the top label of the header and, b) push and/or pop the top label (including the AOLS swapping ability). In previous work, the authors proposed AOLStack, which enhances AOLS by:

- Allocating always space in the header for two labels (or the maximum needed by the stack), even though just one label is coded at some hops,
- Setting the payload/label separation circuit timer of AOLS to extract just one label, the first detected in the stream, and,
- Modifying the NLG sub-block as seen in Fig. 2.

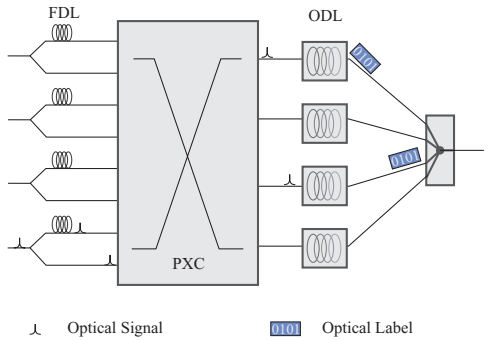


Fig. 2: Enhancement of the *New Label Generation* block for AOLStack.

When a new label needs to be pushed on the stack, the NLG sub-block works as follows. Initially, the matching pulse is split in two. One of the pulses is delayed using an FDL. As a consequence, both pulses are switched by the PXC at different times. After being switched, each one generates a different label by passing through its corresponding ODL. The delayed pulse becomes the bottom label ("swapping" the incoming one) and the other pulse becomes the top label.

The behavior of the NLG sub-block is the same as in AOLS when a label needs to be swapped. In this way, the delayed pulse is discarded. Similarly, popping the stack implies discarding both pulses, or simply the incoming matching pulse.

In this work we consider the distribution of labels in a *per fiber* basis, decreasing the chances of packet contentions [6].

The term *label* and *optical correlator* are going to be used interchangeably, since to process one label we need a fixed number of optical correlators[1].

III. The Label Space Reduction Problem

In this section we review the state of the art of the label space reduction problem in GMPLS. At the end of this section, we contribute with a new concept (the MERLIN groups) which is a stepping stone towards the formulation of our ILP model for AOLS.

We employ the term *segment* to denote a sequence of two or more consecutive links denoting a route in a network between two nodes. In addition, the term *path* is used to mean a segment in the network from an ingress node to an egress node routing a given demand.

A. The Stacking Problem

By stacking a new label on a set of packets belonging to different paths, downstream nodes need to consider just the top label (the same for all paths) to route the traffic. In this sense, we say that: a) there is a label space reduction since one label is used to forward a set of paths and, b) a *tunnel* is set up to forward the packets from all paths. Clearly, the pushed label needs to be popped before any of the paths diverges [11].

When a tunnel ϕ is used to forward the packets of a path α, we say that the tunnel ϕ *covers* path α (at a given link). The norm of a path (or tunnel), i.e. $|x|$, is the number of traversed nodes along its route. The intersection of two paths (or two tunnels, or a path and a tunnel), i.e. $x \cap y$, is the segment that both have in common.

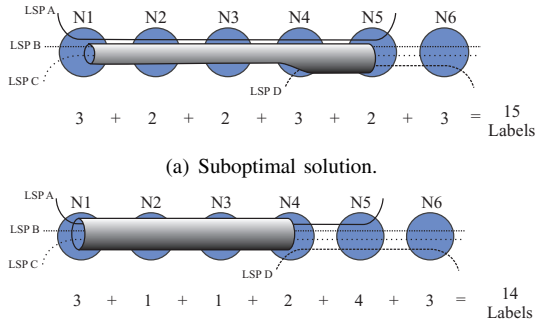
Finding the minimum number of needed labels (or correlators) to route a set of paths is not an easy task. For example, Fig. 3 shows two different solutions for the same problem. In this example, the first solution (Fig. 3(a)) builds a tunnel that makes nodes in the network use a total of 15 labels, while the second solution (Fig. 3(b)) makes them use just 14 labels.

The Stacking Problem becomes harder if we consider that: a) one path can be covered by only one tunnel in a link and, b) there could be more than one tunnel capable of covering a path on a given link.

B. Where are the Tunnels?

Even though solving the Stacking Problem is hard, in previous work, we demonstrated a lemma that aids us in its solution [11]:

Lemma 1 (Stacking Problem Space): Given a set of paths in the network, a set of tunnels *containing* the optimal solution



(a) Suboptimal solution.

(b) Optimal solution.

Fig. 3: Stacking problem example.

[1]Indeed, we need W correlators per label in a per fiber basis, where W is the number of wavelengths per fiber. In this sense, "reducing the label spaces" actually means "reducing the number of installed correlators in an AOLS-block".

to the label space reduction problem can be computed in polynomial time, as follows:

- Let $P = \{\alpha_0, \alpha_1, \ldots, \alpha_{|P|}\}$ be the set of given paths. For every different pair of paths, α and β, the segments resulting from intersecting α with β conform the *primary tunnels*.
- Let $T' = \{\phi'_0, \phi'_1, \ldots, \phi'_{|T'|}\}$ be the set of primary tunnels. For every different pair of primary tunnels, ϕ' and θ', the segments resulting from the difference between them conform the *secondary tunnels*.
- The set of tunnels to be considered for the problem solution space are both, the primary tunnels and the secondary tunnels.

The lemma above allows us to reduce our search space in the optimization problem to only the computed set, and not all the possible paths in the network.

In [12], the Longest Segment First algorithm (LSF) and the Most Congested Space First algorithm (MCSF) are described. While the LSF creates tunnels as long as possible, the MCSF creates them as wide as possible.

C. Label Merging

As commented in [4], two Label Switched Paths (LSPs) in MPLS can be merged at certain Label Switched Router (LSR) if, and only if, they follow exactly the same route downstream that LSR. When two LSPs are merged, the same label is assigned to the segment they have in common. Therefore, one label is allocated for both (or many more) LSPs, decreasing the overall number of labels that are needed.

The contributions on label merging are more extensive than in label stacking. We briefly highlight two of them. Saito *et al.* proposed an ILP formulation in [13] that aims at reducing the number of labels using label merging, but not stacking. Similarly, Applegate and Thorup in [14] gave a bound on the label space under especial assumptions.

We claimed in previous work that: given a set of paths, determining how these paths can be merged in order to obtain the minimum number of labels is a deterministic process, as inferred from [4]:

1) Let i be a node and P_i a set of paths (input parameters)
2) Let $n(i)$ be the set of neighboring nodes of i
3) For every node $j \in n(i)$,
   a) Let $P_{i/j}$ be the set of paths that are forwarded through j to i
   b) All the paths in $P_{i/j}$ use the same incoming label at i (hence the same outgoing label at j).
   c) Call recursively with parameters j and $P_{i/j}$

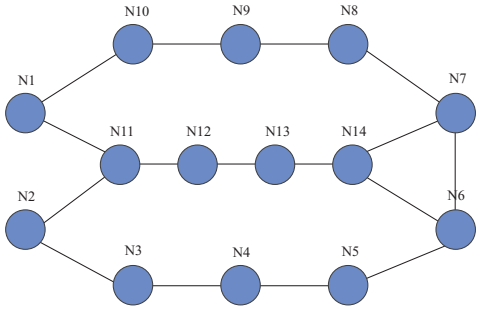
The procedure is initially called with parameters e and P_e, where P_e represents the set of paths that have as egress e.

Considering a particular link in the network, there are disjoint groups of paths that can be merged together. We call this group a MERgeable LINk group of paths, or *MERLIN group* for short. A link can have more than one MERLIN group because: a) the paths traversing the link have different egress, b) even though the set of paths have the egress in common, they do not follow the same route (diverging) downstream the link. The concept applies as well to tunnels. Note that the number of used labels (if label merging is considered solely) is equal to the number of MERLIN groups needed to route the traffic, by definition.

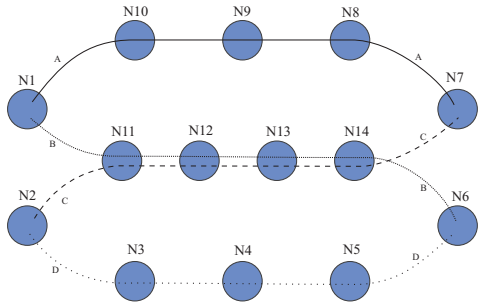
D. Label Space Size vs. MLU: An Example

In this subsection we present an example showing how the MLU affects the label space, and how this repercussion is eased when label merging and stacking are considered.

Let us consider the physical fiber topology of Fig. 4(a) with 14 nodes, in which nodes N1 and N2 are ingress and nodes N6 and N7 are egress. Let us assume that we need to route one unit of traffic from both ingresses to both egresses. More precisely, we denote by A the connection needed from N1 to N7; B the connection from N1 to N6; C the connection from N2 to N7 and; D the connection from N2 to N6. The solutions shown in this subsection do not contemplate the possibility of splitting the demanded bandwidth across several paths.



(a) Fiber Topology.



(b) 22 labels and MLU of 2 units of traffic along 3 links.

Fig. 4: Routing and Traffic Engineering.

A classical Traffic Engineering (TE) routing algorithm could aim at routing traffic such that the MLU is minimized while bounding the delay[2]. A typical TE solution for this example can be seen in Fig. 4(b). The solution has the minimum delay (or hop count) and the minimum MLU. In this case, the number of used labels equals the hop count, i.e. 22, and the MLU does not exceed two units of traffic along three links (N11-N12, N12-N13 and N13-N14).

Considering the *label merging* feature, a different routing solution leads to the usage of fewer labels. For instance,

[2]For simplification, we assume that hop count is equal to the delay

in Fig. 5(a) connection A is using a different route, so its labels can be merged with connection C from node N11 to N7. Similarly, connection B is merged with connection D. Even though the number of labels is reduced in this case down to 16, the MLU has been increased up to four units of traffic along the same three links.

Label stacking, together with label merging, gives much more options to play with. In Fig. 5(b) another solution is given to the problem reducing the label space to its minimum. In this case, 12 labels are needed while the maximum link utilization is preserved to four units of traffic along the same segment.

In case the MLU is limited to (or links capacities are fixed to) two units of traffic, the solutions in Fig. 5(a) and Fig. 5(b) are not feasible. However, the solution in Fig. 4(b) can use 19 labels if one tunnel is created, as seen in Fig. 5(c). In addition, in Fig. 5(d), a different routing solution that reduces the label space down to 16 can be seen as well. While the solution in Fig. 5(c) preserves the TE routing (achieving the best link utilization and delay), the solution in Fig. 5(d) increases the number of links reaching the MLU to eight but reduces even more the label space.

Discussion Summary. Considering label merging and stacking, the more labels we save, the more link congestion is achieved (because of traffic aggregation).

IV. Modeling Routing & Label Space Reduction

The ILP model proposed is path-based. Therefore, all feasible paths in the network are initially generated using an exponential algorithm. Considering the computed paths, all the feasible tunnels are computed as mentioned in Lemma 1.

The following is a list of all the indexes used in the model.

- $i, j \in \mathcal{N}$ represent optical nodes in the network.
- $\alpha \in \mathcal{P}$ represents a generated path in the network.
- $\phi \in \mathcal{T}$ represents a tunnel in the network.
- m, n represents a MERLIN group identifier.

The parameters used in the model are the following.

- S_j^α is set to 1 if node j is the source of path α.
- D_j^α is set to 1 if node j is the destination of path α.
- $C_{(i,j)}$ is set to the total bandwidth demand between nodes i and j.
- κ is set to the minimum of the desired MLU and the link capacity.

In addition, two parameters can be seen as functions in the model.

- $f_m(i,j)$ Given a path MERLIN group m and a link (i,j), the function evaluates to the set of paths belonging to the group.
- $f_n(i,j)$ Given a tunnel MERLIN group n and a link (i,j), the function evaluates to the set of tunnels belonging to the group.

The variables used in the model are the following.

- $\bar{x}^\alpha$ is set to 1 when path α is used to route any demand.
- x^α is set to the bandwidth allocated on path α.
- $y^{\phi,\alpha}$ is set to 1 when path α is covered by ϕ.
- $z_{(i,j)}^m$ is set to 1 when the path MERLIN group m uses a label on link (i,j).
- $\hat{z}_{(i,j)}^n$ is set to 1 when the tunnel MERLIN group n uses a label on link (i,j).

The objective function is minimizing the overall number of used labels (or MERLIN groups) in the network (for both paths and tunnels):

$$\min \sum_{(i,j)} \left(\sum_m z_{(i,j)}^m + \sum_n \hat{z}_{(i,j)}^n \right) \tag{1}$$

Subject to:

Routing Constraints

$$\sum_{\alpha: S_i^\alpha = D_j^\alpha = 1} x^\alpha \geq C_{(i,j)}, \qquad \forall i,j \tag{2}$$

$$\sum_{\alpha | \exists m, \alpha \in f_m(i,j)} x^\alpha \leq \kappa, \qquad \forall i,j \tag{3}$$

$$x^\alpha - \kappa \cdot \bar{x}^\alpha \leq 0, \qquad \forall \alpha \tag{4}$$

(2) assures that all traffic is routed. (3) limits the capacity that can be used in every link to κ. (4) sets a path as requiring labels if it is been used by some demand.

Tunneling Constraint

$$\bar{x}^\alpha - \sum_{(i,j) \in \phi \cap \alpha} y^{\phi,\alpha} \geq 0, \forall i,j,\alpha \tag{5}$$

(5) relates the variable $\bar{x}^\alpha$ with $y^{\phi,\alpha}$. It states that at most one tunnel can be used for a path at every link.

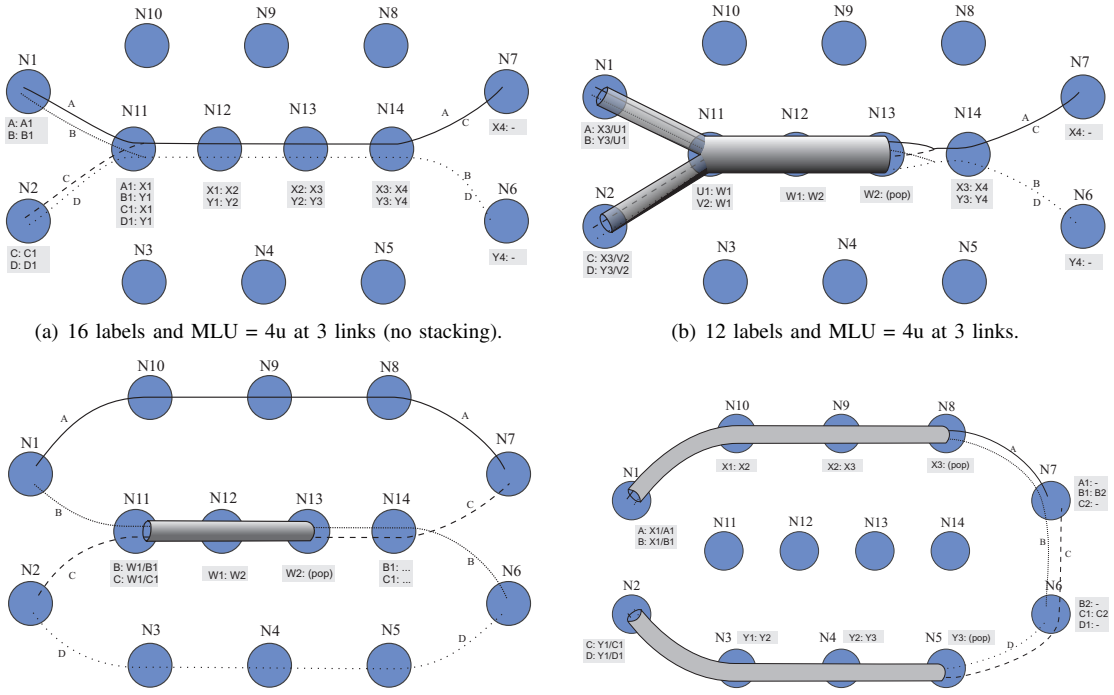
Merging Constraints

$$K \cdot \hat{z}_{(i,j)}^n - \sum_{\phi \in \hat{f}_n(i,j), (i,j) \in \alpha \cap \phi} y^{\phi,\alpha} \geq 0, \qquad \forall i,j,n \in N_{(i,j)} \tag{6}$$

$$K \cdot z_{(i,j)}^m - \sum_{\alpha \in f_m(i,j)} \left(\bar{x}^\alpha - \sum_{(i,j) \in \alpha \cap \phi, D_j^\phi = 0} y^{\phi,\alpha} \right) \geq 0, \qquad \forall i,j,m \in M_{(i,j)} \tag{7}$$

The constant K is set to the minimum number such that $K > \sum_m |f_m(i,j)|, \quad \forall i,j$.

In (6), the variable $\hat{z}_{(i,j)}^n$ pays (i.e. is set to 1) for the use of one label for all the active tunnels ϕ intersecting any active path α. (7) works similarly, but it concerns with paths merging. It differs from (6) in that the term $\sum_\phi y^{\phi,\alpha}$ is added in order to avoid paying for those paths that have been covered.



(a) 16 labels and MLU = 4u at 3 links (no stacking).

(b) 12 labels and MLU = 4u at 3 links.

(c) 19 labels and MLU = 2u at 3 links.

(d) 16 labels and MLU = 2u at 8 links.

Fig. 5: Link Utilization vs. Label Space Size.

V. Solving the Routing and Label Space Reduction Problem using Heuristics

The problem is solved in two-steps by two separate algorithms proposed in this section. The first algorithm routes a traffic demand matrix aiming at setting up paths such that they share the maximum number of links. Once all demands (if possible) are routed, the second algorithm creates a set of tunnels reducing the label space.

A. The Path-Interfering Routing Algorithm (PIRA)

As routing solution, we propose a modification of the Constraint Shortest-Path algorithm (CSPF). The modification consists in how the weight of links is computed. Traditionally, the weight of a physical link is set to its (propagation) delay or proportional to its available bandwidth. In this paper, we propose a new dynamic weight that aims at favoring links using more labels. Our intention is that, after many iterations of the routing algorithm, there are going to be segments in the network that are highly loaded with paths. These segments will later cause a high label space reduction when tunnels are placed to cover them.

Given a network $G^*(V, E^*)$ and a set of paths P, we extend G^* to a *directed multigraph* $G(V, E)$ as follows.

Every node and link in G^* are also considered in G. We name these links: *physical links*. The bandwidth of a physical link in G is set to the available bandwidth of its corresponding link in G^*. In addition, its weight is fixed to one.

Given a pair of non-adjacent nodes $i, j \in V$ and the set of paths P, it is worth noting that:

- There could be several paths that are forwarded through the same segment from i to j.
- Considering the set of paths P, there could be more than one segment that forwards information between i and j.

Regardless of which paths they belong to, we denote $s^{(k)}_{i\to j}$ the k-th different segment from i to j considering P. We create one different link from i to j in G for every $s^{(k)}_{i\to j}$. We name these links: *induced links*. The bandwidth of an induced link in G is set to the minimal available bandwidth of the links in G^* conforming its corresponding segment. In addition, its weight is set to

$$\frac{|s^{(k)}_{i\to j}| - 1}{|P(s^{(k)}_{i\to j})| + 1}, \quad \text{where}$$

$|s^{(k)}_{i\to j}|$ is the length of the segment and $|P(s^{(k)}_{i\to j})|$ is the number of paths in P that traverse the segment. The idea behind the weight is that a new path "pays", at every hop, only for the share of the label that it uses. One (1) is subtracted from the numerator because the last hop of $s^{(k)}_{i\to j}$ never incurs in labels reduction [11]. One (1) is added to the denominator in order to consider the new path as well.

When CSPF takes an induced link to route a demand, the link should be interpreted instead as the set of physical links in G^* representing it. If a cycle is created in the physical links, it is broken. It is worth noting that, even though the number of links contemplated increases, the complexity of the CSPF-procedure should not be worse than that run with a full mesh graph.

B. The Most-Profitable Tunnel First Algorithm (MPTF)

We consider that path routes were already computed. Taking into account these path routes, we compute the set of tunnels to consider as described by Lemma 1.

It is worth noting that the number of labels that ϕ can reduce by covering α is $|\alpha \cap \phi| - 1$, however ϕ needs $|\phi| - 1$ labels despite the number of paths it covers. This gives us an idea of the *profit* gained by setting up a tunnel. More concretely, in this paper, the profit of a tunnel is the number of labels that it reduces by covering all the non-covered segments of the paths.

Note that our metric (i.e. the profit) considers the "savings" due both covered length and covered broadness, gathering in one metric the best of both LSF and MCSF.

At each iteration, a tunnel - name it ϕ - with the *best profit* is always selected. Once a tunnel is selected, all the non-covered segments of paths that ϕ can cover - name them $P(\phi)$ - are marked as covered. These covered segments will not be considered in further iterations. The profit of all remaining tunnels must be updated. The algorithm iterates until no tunnel remains.

At the beginning of the algorithm, since all paths are completely uncovered, the profit of a tunnel ϕ is set to:

$$\Pi(\phi) \leftarrow 1 - |\phi| + \sum_{\alpha \in P(\phi)} (|\alpha \cap \phi| - 1)$$

Before the next iteration takes place, the profit of all the others tunnels θ overlapping with ϕ is updated accordingly to:

$$\Pi(\theta) \leftarrow \Pi(\theta) + |\phi \cap \theta| \cdot \delta_{\phi,\theta} - \sum_{\alpha \in P(\phi)} (|\alpha \cap \theta| - 1), \quad \text{where}$$

$\delta_{\phi,\theta}$ is set to one (1) if tunnels ϕ and θ end at the same node, zero (0) otherwise.

Neither ϕ nor any other tunnel $\phi' \subset \phi$ are considered any more in further iterations. The complexity of the algorithm is bounded by $O(t \cdot \log t)$, where t is the number of feasible tunnels.

VI. Simulation Experiments

In this section, we present a set of simulations that shows the discussed trade-off in this paper.

A. Heuristic Performance

The European network consisting of 37 nodes is used in our simulation experiments, see Fig. 6. The link capacity is varied from 100 units to 3000 units of traffic, creating different simulation scenarios limiting the MLU. We use the term link capacity and MLU to denote the same throughout our analysis.

The first 30% of the nodes with lowest connectivity degree are selected as edge (ingress/egress) routers. For each pair of edge routers, a number of X, Y and Z demands of 1, 3 and 12 units of traffic is randomly generated. X, Y and Z follow an uniform distribution with parameters [0-20], [0-12] and [0-4] respectively. As a result, we generate an average of 200

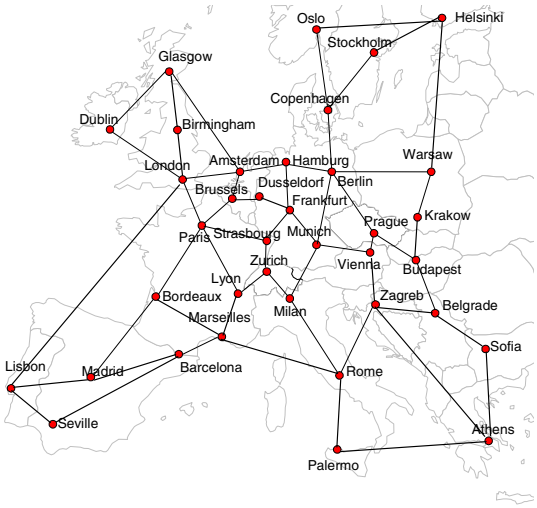


Fig. 6: European network with 37 nodes.

demands with average demanded bandwidth of 6864 units of traffic.

In this subsection we tested several combinations of routing heuristics with label space reduction heuristics. Namely, we considered as routing heuristics: CSPF and PIRA (proposed in this paper). As for label space reduction heuristics, we considered: MultiPoint-to-Point [13], [14] (MP2P, i.e. label merging without stacking), LSF, MCSF and, MPTF (proposed in this paper).

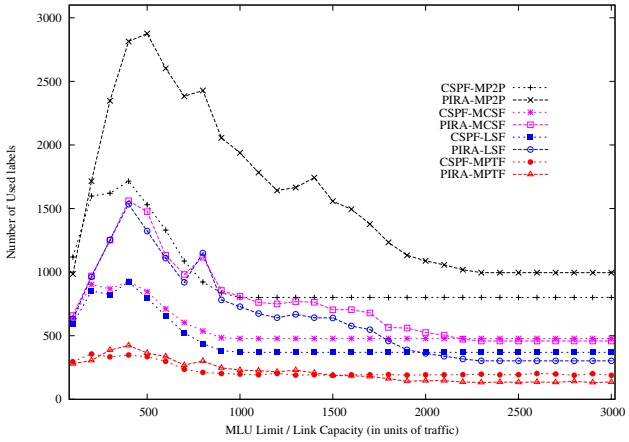
In total, eight routing-tunneling solutions are contemplated. In Fig. 7(a), simulation results show that the best label space reduction is achieved by PIRA-MPTF when the MLU limit is high, and CSPF-MPTF when the MLU is low. In general, PIRA obtains better reductions when used with high MLUs and with any stacking heuristic (i.e. LSF, MCSF and MPTF). In the same way, MPTF obtains the best reduction regardless of the routing heuristic.

The use of the stack reduces the label space four times (CSPF-MP2P vs. CSPF-MPTF) in average when the capacity of the links is just enough to route traffic (around 1000 units of traffic), and almost six times if the capacity is doubled (CSPF-MP2P vs. PIRA-MPTF).

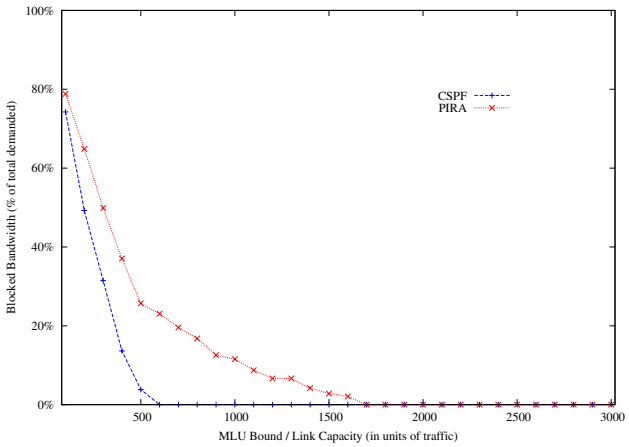
It is worth noticing that the number of labels increases at the beginning, when the MLU limit is low. This behavior is explained by the fact that the routing heuristics (either CSPF or PIRA) employ more, and longer, paths to accommodate the traffic in these scenarios, in order to avoid the violation of the MLU limit. In case of PIRA, this behavior is stronger and particulary emphasized (see peaks in figure) when the MLU limit is set to 800 and 1400 units of traffic.

As expected, PIRA creates more link bottlenecks than CSPF in order to provide more tunneling possibilities, see Fig. 7(b). Therefore, the usage of PIRA is advisable when the minimum spare capacity of the links doubles the MLU. In our simulations, we noticed that the usage of PIRA in this circumstances profits with a 30% in the label space reduction.

We proceed to focus now in the particular routing solutions



(a) Label Space Size vs. Link Capacity.



(b) Blocked Bandwidth.

Fig. 7: Heuristics Overall Performance.

when the capacity of the links is high. While the MLU utilization of CSPF is 934 units of traffic, PIRA's is 2236; this is 2.5 times more. However, we notice that this case occurs in few links. In Fig. 8 we show the number of links in PIRA whose used capacity is above (or below) a given percentage of the MLU of CSPF (934 units of traffic). For instance, there are 6 links in PIRA that are using between 50% and 75% *more* capacity than the minimum MLU considering CSPF routing. It turns out that while 25 links (out of 114) requires a higher capacity, 74 are not used by PIRA (16 of them are not used by CSPF either). This suggests the idea of either: a) rewiring the network (instead of increasing the existing links capacities) or, b) employing the unused links to create lightpaths providing the extended capacity to overused links. These ideas will be studied in further contributions.

We compare now the best solution without stacking (CSPF-MP2P) with the best solution using the stack (PIRA-MPTF). The maximum number of labels that CSPF-MP2P uses is 20. This makes all AOLS-blocks to be sized for coding labels of five bits long. By using stacking (PIRA-MPTF), the maximum number of labels becomes 11, making the AOLS-blocks to be

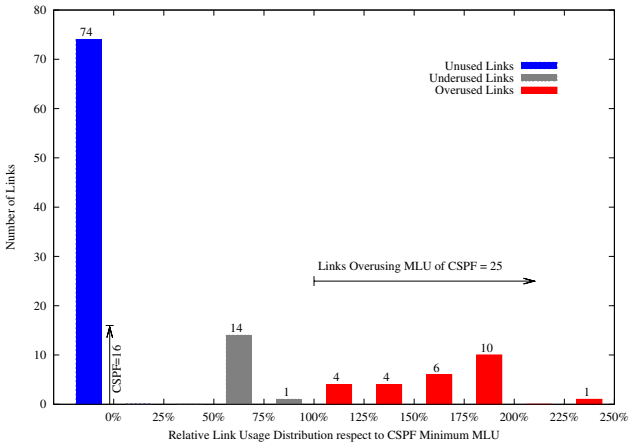


Fig. 8: Distribution of overused link capacities of PIRA respect to CSPF MLU.

sized for coding labels of only four bits long. In Fig. 9 we show the number of links that are using a number of labels within a given range. It shows that 11 links are causing the five bits long labels without stacking. However, using the stack, only four links are forbidding us from using three bits long labels. Even though we did not optimize the maximum number of labels per link, it is not difficult to see that rerouting the traffic in the three links of PIRA-MPTF would be easier than rerouting the traffic of the 11 links in CSPF-MP2P in order to reduce one bit the label encoding size.

Considering the header in [7], we compute the overhead due the coding of the extra label. In the case of no label stacking, the five bits long label yields to a 17 bits header. In the case of label stacking, the four bits long labels yield to a header of 23 bits. Assuming a classical packet distribution (as in [6]), the average overhead caused in the traffic of the network is just 0.18% more due stacking.

It is worth noting that if *label stripping* using CSPF is considered, packet header must code eight labels (minimum

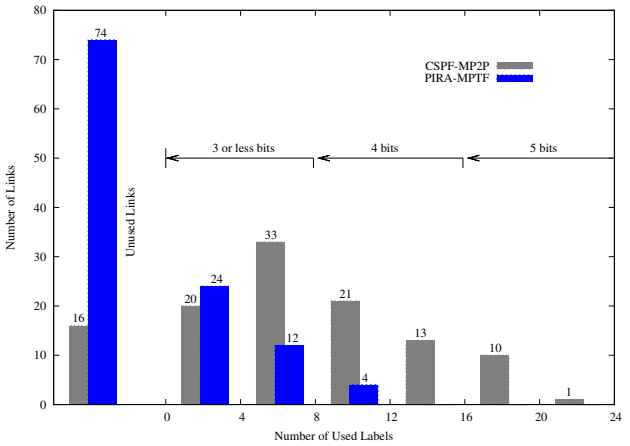


Fig. 9: Distribution of the label space link-by-link using PIRA-MPTF.

distance in CSPF paths) in the header and the network must be able to handle 102 labels (the number of used links by CSPF). If PIRA is considered, packet header must code 17 labels, and the network must be able to handle 40 labels. However, the traffic overhead is 2.02% and 4.5% more, respectively. Therefore, label stacking offers a better trade-off.

B. How Far From Optimum?

Each one of the two heuristics presented in this paper affect the overall optimality of the solutions. Therefore, we compare our results with the optimal in two steps:

1) Minimize Labels considering CSPF/PIRA Routes: We route the traffic using CSPF and PIRA and then we relax the proposed ILP so it computes the best label space reduction using CSPF/PIRA paths. We consider the network scenario in which the MLU limit is the highest. By solving the relaxed ILP using the CSPF routes, the minimum number of used labels considering stacking is 180. Considering PIRA, the minimum number of used labels decreases down to 109. In this way, we can claim that: a) PIRA-routing leads to paths aiming at better label space reduction in general and, b) MPTF performs 21.11% far from its optimal value considering PIRA. Henceforth, we only consider PIRA-MPTF.

2) Minimize Labels with Variable Routes: We solve the complete ILP model proposed in §IV with a smaller network shown in Fig. 10. Edge nodes and demands are generated following the same parameters of the European network. The MLU is set to the double of the minimum needed by CSPF. We found that PIRA-MPTF label space size is 26.7% more than the optimal. A similar test to the previous one exposes that 68% of the error is caused by the selection of the routes. The reason is the tendency of PIRA for using more paths than the optimal. This behavior is mainly explained by the link congestions caused by PIRA and by the usage of long routes.

VII. Conclusion

We considered AOLStack as an enhancement of AOLS that allows us to perform label stacking completely optically. The benefits of using AOLStack are the employment of either fewer labels when the MLU is fixed or, which is the same, a smaller MLU using the same number of labels.

We proposed an ILP and a two-steps heuristic. We performed our simulations based on the two-steps heuristic and later corroborated its results with the ILP.

In brief, simulation experiments showed that the label space can be reduced six times using the stack, if the MLU limit (or link capacity) is increased at least twice. If the MLU is kept to the minimum needed, the use of the stack yields to a four times label space reduction. As a consequence, the usage of the stack reduces in one bit the length of the labels needed; reducing the AOLS-block not only in number, but in size as well. However, considering a typical packet size distribution, a new label in the packet header incurs 0.18% of traffic overhead in the network, which is much more less than that caused by label stripping.

We also observed that the creation of lightpaths may reduce the congestion in some links. This is going to be tackled in future research.

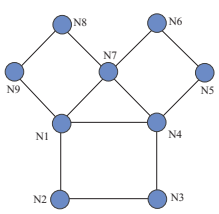


Fig. 10: Network simulated for ILP

Acknowledgements

The work was partly funded by[3]: a) the European Commission through project(s) IST-LASAGNE, b) IWT-Vlaanderen, c) the Department of Universities, Research and Information Society (DURSI) of the Government of Catalonia and the European Social Funds, d) the Spanish Science and Technology Ministry and, e) the European COST 293 project.

References

[1] Sudhir Dixit. *IP over WDM: Building the Next Generation Optical Internet*. Wiley-Interscience, March 2003.

[2] Achille Pattavina. Architectures and performance of optical packet switching nodes for IP networks. *J. Lightw. Technol.*, 23(4):1601–1609, April 2005.

[3] Gigi Karmous-Edwards and Admela Jukan. An optical control plane for the grid community: Opportunities, challenges, and vision. *IEEE Commun. Mag.*, 44(3):62–63, March 2006. Guest editorial.

[4] E. Mannie. *Generalized Multi-Protocol Label Switching (GMPLS) Architecture*. IETF, October 2004. RFC 3945.

[5] Daniel Blumenthal et al. All-optical label swapping networks and technologies. *J. Lightw. Technol.*, 18(12):2058–2075, December 2000.

[6] Ruth Van Caenegem, Didier Colle, Mario Pickavet, and Piet Demeester. Benefits of label stripping compared to label swapping from the point of node dimensioning. *Photonic Network Communications Journal*, 12(3):227–244, December 2006.

[7] Francisco Ramos et al. IST-LASAGNE: Towards all-optical label swapping employing optical logic gates and optical flip-flops. *IEEE J. Sel. Areas Commun.*, 23(10):2993–3011, October 2005.

[8] M. Kodialam and T.V. Lakshman. Minimum interference routing with applications to MPLS traffic engineering. In *Proc. IEEE INFOCOM*, volume 2, pages 884–893, 2000.

[9] C. Bintjas et al. All-optical packet address and payload separation. *IEEE Photon. Technol. Lett.*, 14:1728–1730, 2002.

[10] J.M. Martinez, F. Ramos, J. Marti, J. Herrera, and R. Llorente. All-optical N-bit XOR gate with feedback for optical packet header processing. In *Eur. Conf. Optical Communication (ECOC)*, volume 3, 2002.

[11] Fernando Solano, Thomas Stidsen, Ramon Fabregat, and Jose Marzo. Label space reduction in MPLS networks: How much can one label do? *IEEE/ACM Trans. Netw.*, 2007. In 2nd review. http://eia.udg.es/~fsolanod/papers/solano-ton2007-r1.pdf.

[12] Fernando Solano, Ramon Fabregat, and Jose Marzo. A fast algorithm based on the MPLS label stack for the label space reduction problem. In *Proc. IEEE IP Operations and Management (IPOM 2005)*, October 2005.

[13] Hiroyuki Saito, Yasuhiro Miyao, and Makiko Yoshida. Traffic engineering using multiple MultiPoint-to-Point LSPs. In *Proc. IEEE INFOCOM 2000*, pages 894–901, 2000.

[14] David Applegate and Michel Thorup. Load optimal MPLS routing with N+M labels. In *Proc. IEEE INFOCOM 2003*, pages 555–565, 2003.

[3]Grant references 2005-SGR-00296, 2006-BE-00272, 2006-FIR-00109 and M2R3 - TEC2006-03883/TCM