# Virtual Private Network To Spanning Tree Mapping

Yannick Brehon, Daniel Kofman[1], and Augusto Casaca[2]

[1] ENST, 46 rue Barrault, 75013 Paris, France,
brehon@ , kofman@enst.fr
[2] INESC-ID/IST, Rua Alves Redol 9, 1000-029 Lisboa, Portugal,
augusto.casaca@inesc.pt

**Abstract.** Metro Ethernet has been widely adopted lately. One of its goals is to provide clients with VPN services, and in this context, load-balancing and protection have become important issues. This paper presents an original formulation of these problems in Metro Ethernet, and introduces an algorithm for load balancing which provides good numerical results.

## 1 Introduction

Ethernet has been a very successful technology and has been widely accepted in local networks. Network operators have been inclined to push this technology further into their network, following a cost-effective desire of unification. Therefore, Ethernet is now becoming the technology of choice when building Metropolitan Area Networks (MANs).

Ethernet MANs will support VPNs (Virtual Private Networks), using the concept of VLANs (Virtual Local Area Networks), such as defined by [1]. VPNs are distributed LANs which are connected using a provider-provided network. Once connected, nodes in VPNs are able to communicate as if they were physically directly connected.

VLANs allow a single network to interconnect several groups of Ethernet nodes - groups which, in a VPN context, will be the VPNs - and maintain a separation between those groups, isolating one from another.

There are many advantages to the Ethernet technology. It has been around long enough and deployed widely enough to benefit from substantial cost reductions. It is now a mature technology which is very cost effective, easy to deploy and easily inter-operable. However it lacks a decent control plane. For the network operator, this means there are no resource reservation schemes available and no load-sharing mechanisms. For the network user, this means there are no QoS (Quality of Service) guarantees and no protection mechanisms for failure resiliency. These are essential drawbacks which need to be addressed when deploying MAN-grade Ethernet. These major issues are therefore a focus of great interest.

For the protection issues, the Spanning Tree Protocol [3] (STP) has evolved to a Rapid Spanning Tree Protocol [4] (RSTP). These protocols are in charge of building the Spanning Tree (ST) which will support the Ethernet traffic. The RSTP version accelerates reconvergence of the tree when links or nodes of the current tree fail, bringing the convergence time from 30 seconds to 3 seconds in most failure scenarios. While this is still no protection mechanism, the recovery times are at least more acceptable.

As far as load balancing is concerned, the introduction of the Multiple Spanning Tree Protocol [5] (MSTP) allows for multiple trees to coexist on a single network. Each tree has an identifier and traffic is mapped to various trees so as to allow some sort of traffic engineering. For instance, using multiple trees, it is possible to achieve load balancing in the network by spreading the load across trees which use different links.

Many papers have studied how to do traffic engineering using MSTP [6–8]. [6] shows the tradeoff between the number of STs deployed, the alternate routing and the load-balancing. It also offers an algorithm to group various VPN clients onto fewer STs. This grouping strategy however does not show how to initially build the STs. [7] proposes a heuristic algorithm to build a spanning tree for each VPN. In the end, there are as many trees deployed as there are VPNs in the network: this raises both management and scaling issues. Finally, [8] proposes a control admission scheme to make the best use of deployed STs. This last paper therefore is adapted for optimal dynamic use of STs.

In this paper, we formulate the problem as an original Mixed Integer Non Linear Program. However, this problem is not computationally tractable. Therefore we propose a heuristic algorithm to help the operator first configure his Ethernet switches, that is, build the STs of MSTP, and then map client VPNs to STs in an optimal way for load sharing considerations.

The rest of the paper is organized as follows. We first introduce the problem of VPNs to ST mapping in Section 2 and show the various considerations a network operator must face. We will then present the formulation of this optimization problem as a Mixed Integer Non Linear Program (MINLP) in Section 3. In Section 4, we introduce our algorithm for heuristic optimal mapping of VPNs onto STs, and results will be shown in Section 5. Finally, in Section 6 conclusions are drawn and future work is presented.

## 2 VPNs and Spanning Trees

The Metro Ethernet architecture (see Figure 1) consists of a core network which relies on Ethernet to transparently connect distant nodes of the network.

A customer may want to interconnect various sites and operate them as a single LAN: he wants to setup a Virtual Private Network (VPN). When this VPN uses a Metro Ethernet provider, it appears as a C-VLAN (Customer VLAN) for the provider. Each of his sites is connected to the core network via a gateway: the access point (AP). The network provider is in charge of transparently interconnecting these APs according to some Service Level Agreement (SLA).

The notion of VLAN identifier (VID) was introduced for this purpose. The use of VLAN IDs enables the logical separation between groups of Ethernet clients (otherwise impossible due to Ethernet's flat addressing). When a frame belonging to a given client enters the provider's network, it is tagged with a VID. Each port of the Ethernet switches in the core are assigned VIDs. The switch will forward the tagged frame to and from a port only if it is part of the corresponding C-VLAN. This mechanism allows the different C-VLANs to use the same provider network and yet, C-VLANs are isolated one from another.

Ethernet uses a spanning tree protocol which prevents loops in the forwarding. This protocol constructs a shortest-paths tree given a root and link weights. The root is automatically selected based on an administrator-set identifer for each node. The link weights are also set by management. Frames are then sent on this tree, which guarantees a single path between any two nodes, and the switches just need to learn which port gives access to which Ethernet destination machine.

However, using a tree structure means many links are not used, since they are left out of the structure. This means the network resources are wasted. One way to overcome this issue is to use the Multiple Spanning Tree Protocol (MSTP). This protocol allows, among other things, to have several Spanning Trees (STs) on the same network. Each tree has a single identifier. The C-VLANs are mapped to STs, such that each C-VLAN uses only one of the available STs. However, one ST can carry multiple C-VLANs. The question is to build the STs and have them carry the C-VLANs in a way which is optimal: in Figure 1, there are many ways to map C-VLANs 1, 2 and 3 to the two STs. The optimality can be defined in many ways.

One may want to minimize total network resources usage, minimize the maximally loaded link (that is improve load sharing) or have distinct trees (or at least, trees which share the minimum amount of links) for failure resiliency. These aspects will be dealt with in the next sections.

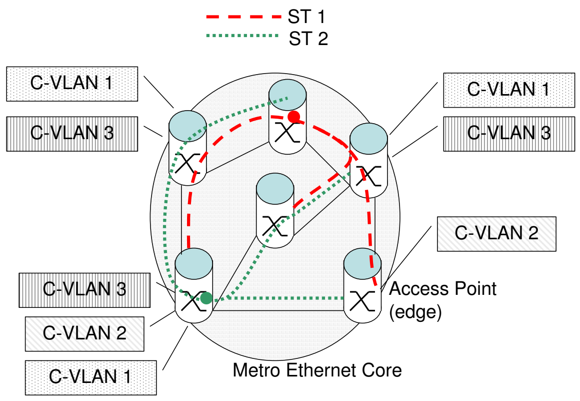

**Fig. 1.** Metro Ethernet Architecture

## 3 Optimization Problem

In this section, we will provide a mathematical formulation of the above described problem in the form of a MINLP (Mixed Integer Non Linear Program). Given a physical topology, which STs and which assignment of VPNs to STs make the optimal use of the network resources? We first make preliminary observations which are used in the MINLP formulation.

### 3.1 Preliminary observations

For simplicity of the formulation in Section 3.2, we will consider here that nodes in a given VPN $i$ are all given the same access bandwidth ($D_i$). We also assume there is a uniform traffic matrix. This implies that the bandwidth is reserved in such a way that along any tree supporting a given VPN, there must be enough bandwidth to satisfy traffic of volume $D_i/(Nb\_Clients(i)-1)$ between all nodes in the VPN, where $Nb\_Clients(i)$ is the number of nodes belonging to VPN $i$. However, simple modifications will allow nodes to have differentiated accesses, by adding weights to the nodes.

The calculation of the bandwidth needed on a single link is very simple. Consider two nodes on a tree such as those depicted in Figure 2. One node gives access to $p$ nodes of the VPN, the other node to $N-p$. Then, for the above bandwidth reservation hypothesis to apply, we need to reserve

$D_i * p * (N-p)/(N-1)$ units of bandwidth on the link. The key element to understanding the formulation is therefore the calculation of how many nodes are accessed through a given node.

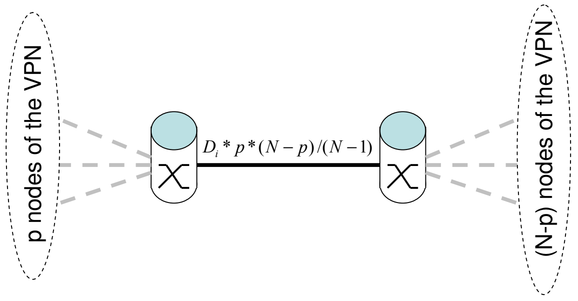

**Fig. 2.** Link Bandwidth Calculation

### 3.2 Optimization Problem Formulation

Let us assume that:

- $v = 1, 2, .., n$ indices network nodes

- $e = 1, 2, .., p$ indices links of the network
- $i = 1, 2, .., Nb\_VPN$ indices the VPNs we are trying to setup where $Nb\_VPN$ is the total number of VPNs to be supported.
- $j = 1, 2, .., Nb\_ST$ indices spanning trees, where $Nb\_ST$ is the maximum amount of Spanning Trees which can be setup in the network.

Additionally we will assume that the following constants are known by the operator:

- $D_i$ is the bandwidth allocated to each node of VPN $i$
- $P_v^i$ is the binary indicating that node $v$ is a part of VPN $i$. A node may only be a part of a single VPN: one node must be created for each VPN connected to a given AP. This is not a restriction, it is simply for clarity of the formulation. A simple way to ponder some nodes in regards of others is therefore to connect more nodes of the same VPN to a given AP.
- $Nb\_Clients(i) = \sum_v P_v^i$ is the number of clients of VPN $i$
- $a_{ev}$ is the binary indicating that link $e$ has an extremity at $v$
- $C_e$ is the capacity of link $e$
- $M_e$ is the module of link $e$ (number of trees which the link can support)

Finally, the optimization problem will manipulate these variables:

- $x_{eij}$ is the flow allocated to link $e$ for VPN $i$ on tree $j$
- $x_{ej}$ is the total flow allocated to link $e$ on tree $j$
- $x_e$ is the total flow allocated to link $e$
- $Map(i, j)$ is the binary variable set to 1 if VPN $i$ is mapped to tree $j$
- $if(v, e, i, j)$ is the integer variable indicating how many nodes of VPN $i$ are available behind node $v$ when coming from link $e$, using tree $j$.
- $dir(e, j, v)$ is the binary variable indicating that link $e$ is being used in VPN $j$ such that $v$ must go through $e$ to get to the root. If this variable is null, then either the link is not being used in the tree's active topology or it is being used in the other direction.
- $u_j$ is the binary variable indicating whether ST $j$ has at least one VPN mapped to it.
- $1/\Lambda$ is the amount by which the bandwidths assigned to each VPN can be multiplied. $1/\Lambda$ should be greater or equal to 1 if the initial bandwidth is to be assigned.
- $1/\Theta$ is the amount by which the number of trees assigned to each VPN can be multiplied. $1/\Theta$ should be greater or equal to 1 if the modules are significant.

We can then easily formulate a Mixed Integer Non-Linear Program with a set of constraints:

- Domain definition constraints:
  - The $dir()$ function is only different from zero for $(e, v)$ couples such that link $e$ is connected to node $v$

$$dir(e, j, v) \leq a_{ev} \quad \forall j, v, e \tag{1}$$

  - The $if()$ function is only different from zero for $(e, v)$ couples such that link $e$ is connected to node $v$. Also, no interface may present an access for a VPN for more nodes than there are clients (this prevents loops to appear, considered as being part of the ST (wrongly) by the program: the graphs generated by the algorithm are connex, and because of the n-1 active links (see constraint (8)), a spanning tree)

$$if(v,e,i,j) \leq Nb\_Clients(i) * a_{ev} \quad \forall e,i,j,v \tag{2}$$

- Mapping constraint:
  - Only one ST can be used for a given VPN:

$$\sum_j Map(i,j) = 1 \quad \forall i \tag{3}$$

  - ST use:

$$u_j \geq \frac{\sum_i Map(i,j)}{Nb\_VPN} \quad \forall j \tag{4}$$

- Spanning Tree (ST) Definition:
  - In a ST, there are exactly (number of nodes - 1) links used (one link going to the root per node, the *uplink*). We use this property to define our ST:

$$\sum_{e,v} dir(e,j,v) = n-1 \quad \forall j \tag{5}$$

  - One uplink per non-Root node and per ST:

$$\sum_e dir(e,j,v) \leq 1 \quad \forall v,j \tag{6}$$

  - A given link may only be oriented once at most:

$$\sum_v dir(e,j,v) \leq 1 \quad \forall j,e \tag{7}$$

- We must define, for each node, the number of VPN nodes it gives access to when coming from a given link. These are the interface definitions constraints:
  - For ST nodes which are part of the VPN (and are thus, by hypothesis, leaves of the network graph), the interface is set to 1.

$$\sum_{e/a_{ev}=1} if(v,e,i,j) = Map(i,j) \quad \forall j,i,v/P_v^i = 1 \tag{8}$$

  - For ST nodes which are not part of the VPN, the interface gives access to the sum of the values of the interfaces its other interfaces are linked to.

$$\begin{aligned} if(v,e,i,j) \geq & \sum_{\substack{e' \neq e/ \\ a_{e'v}=1}} \sum_{\substack{v' \neq v/ \\ a_{e'v'}=1}} if(v',e',i,j) - \\ & (1 - \sum_{v'} dir(e,j,v')) * MaxClients \\ & \forall j,i,v/P_v^i = 0, e/a_{ev} = 1 \end{aligned} \tag{9}$$

where $MaxClients$ is the largest VPN count ($= max_i Nb\_Clients(i)$). The second term insures that the inequality is only applied to links in the active topology (see constraint 10).

- For all ST nodes, interface must only be set to a positive number if the link is part of the active topology:

$$if(v,e,i,j) \leq \sum_{v'} dir(e,j,v') * MaxClients \quad \forall e,i,j,v \tag{10}$$

– Based on the number of clients of a VPN accessed behind every node, it is easy to define a set of constraints dictating the flow assignation and related conditions.

- Flow on links:

$$\sum_i x_{eij} \leq x_{ej} \quad \forall e,j \tag{11}$$

$$\sum_i x_{ej} \leq x_e \quad \forall e \tag{12}$$

- Flow can only be set if the VPN is mapped to a tree, and only if the link is used in the tree's active topology

$$x_{eij} \leq \sum_v dir(e,j,v) * MaxFlow \quad \forall j,e,i \tag{13}$$

$$x_{eij} \leq Map(i,j) * MaxFlow \quad \forall j,e,i \tag{14}$$

where MaxFlow is a constant large enough not to be a constraint.

- Flow assignation: a link must provide enough flow for every node on one side to be able to setup a connection with a node on the other side (this is based on Section 3.1)

$$\begin{aligned} x_{eij} \geq D_i/(Nb\_Clients(i)-1) * \sum_{\substack{v' \neq v/ \\ a_{ev'}=1}} if(v',e,i,j) * \\ (Nb\_Clients(i) - \sum_{\substack{v' \neq v/ \\ a_{ev'}=1}} if(v',e,i,j)) \\ \forall e,i,j,v/a_{ev}=1 \end{aligned} \tag{15}$$

- Capacity constraint: this constraint can be used in two ways. First, in a resource optimization problem for a given network. Second, by setting all link capacities equal, maximizing $1/\Lambda$ in this constraint will minimize the maximally loaded link, i.e. achieve load balancing.

$$x_e \leq \Lambda . C_e \quad \forall e \tag{16}$$

- Number of trees per link constraint: this constraint can also be used in two ways. First, in a resource optimization problem for a given network, one may want to limit the number of trees on a link. Second, by setting all link modules equal, maximizing $1/\Theta$ in this constraint will minimize the maximal number of trees on any single link, which means that any link failure will impact a minimal number of trees.

$$\sum_{j,v} dir(e,j,v) \leq \Theta.M_e \quad \forall e \tag{17}$$

The objective is now:

$$\text{Minimize } \alpha \sum_{j} u_j + \beta.\Lambda + \gamma.\Theta \tag{18}$$

Where $\alpha, \beta, \gamma$ are the weights assigned according to relative importance of respectively, the cost of the number of STs (management cost), load balancing and protection needs.

Note that constraint (15) is quadratic, semi definite negative, which means that the solution space is concave. This means it is not possible to solve exactly without exploring the entire solution space. So as to solve this problem, we tried linearizing this constraint, but calculations were still untractable (using the commercial solver CPLEX [10]), even for instances of the problem implying very small networks; the following heuristic method was thus conceived.

## 4 Heuristic Mapping of VPNs on Trees

In this section we will present an original heuristic algorithm for mapping VPNs to STs: the *Diversified Forest with Greedy Mapping* (**DFGM**). This method is based on two stages. First, we build a set of STs which aim at having few links in common: the Spanning Forest. Second, we assign VPNs to STs in a greedy fashion.

The rationale behind this process is that this way, the network operator only builds a Spanning Forest once and then maps his various VPNs to it. This method is therefore very convenient for operators, while providing good results (see Section 5).

### 4.1 Building the Spanning Forest

For this stage of the algorithm, the weights of each link will be changed so as to represent how often the link appears in spanning trees. After building a ST, the weights of its links are increased by a fixed value $\Delta$. Initially, all links have a same fixed weight of 1.

When building the STs, an important factor is the choice of the root. The root is typically the node of the tree which will have the most traffic running along it. Therefore, it is important that the roots be as varied as possible, which is why we introduce a variable associated to each node which counts how many times a node has been a root in any ST. The root is then chosen as the node which minimizes:

$$\frac{\sum_{\text{all adjacent links}} W_{link}}{\sum_{\text{all adjacent links}} C_{link}} * (\text{number of times root}) \quad (19)$$

Once the root is chosen, the ST is built by using a method such as Dijkstra's shortest path [9]. The weights are then updated and the process is repeated a fixed number of times $Nb\_ST$.

**Algorithm 1:** Spanning Forest Generation

```
Require: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$
Require: $C_l$ the capacity of link $l, \forall l \in \mathcal{A}$
Require: $W_l$ the weight of link $l, \forall l \in \mathcal{A}$
Ensure: A diversified Spanning Forest
1: Set $W_l = 1 \quad \forall l \in \mathcal{A}$
2: for all $i \in [1, ..., Nb\_ST]$ do
3:     Select a root minimizing the quantity of (19)
4:     Build a spanning tree $ST_i$ rooted at the selected node
5:     $W_l \leftarrow W_l + \Delta, \forall l \in \mathcal{S}T_i$
6: end for
```

## 4.2 Mapping the VPNs on the Spanning Forest

Once the spanning forest has been generated, the VPNs have to be mapped to it. In this part, a simple "greedy" method is applied.

**Table 1.** Relative Algorithm Performances

| | | Network | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | VTHD | | | | | Italian | | | | | Dense | | | | |
| | *Algorithm* → | Single ST | MST without traffic update | MST with traffic update | Enhanced MST | **DFGM** | Single ST | MST without traffic update | MST with traffic update | Enhanced MST | **DFGM** | Single ST | MST without traffic update | MST with traffic update | Enhanced MST | **DFGM** |
| 30 VPNs | **Max.load** | 1678 | 1084 | 930 | 862 | **794** | 5915 | 3658 | 2807 | 2872 | **2252** | 1581 | 650 | 403 | 382 | **286** |
| | Total nb of trees | 1 | 30 | 30 | 30 | **10.4** | 1 | 30 | 30 | 30 | **13.8** | 1 | 30 | 30 | 30 | **11.5** |
| 10 VPNs | **Max.load** | 535 | 390 | 354 | 349 | **270** | 1826 | 1267 | 1055 | 1221 | **785** | 522 | 280 | 202 | 219 | **139** |
| | Total nb of trees | 1 | 10 | 10 | 10 | **6.72** | 1 | 10 | 10 | 10 | **7.29** | 1 | 10 | 10 | 10 | **6.71** |

The VPNs are sorted by their volume (ie: by decreasing order of the quantity $D_i * Nb\_Clients(i)$ using the same notations as in section 3.2).

Next, the largest VPN is mapped to the ST of the Forest which minimizes the load of the maximally loaded link. If more than one ST have the same minimum value for the maximally loaded link, the ST which satisfies the VPN while using the total minimum amount of bandwidth in the network is selected.

## 5 Results

In this section, we applied the heuristic method described in the previous section to three different networks. The first one is a 11 node- and 14 (bidirectional) link-network, based on the French research network VTHD [11]. The second network is a 21 node- and 36 link-network, based on an Italian network [12]. The third network is a highly connected 12 node- and 40 link- network, such as the one described in [7].

We compare our heuristic to the 3 algorithms described in [7]. The first one constructs one ST associated to each VPN. This leads to as many STs as there are VPNs. Links have weights which increase with the load carried, and trees are constructed based on these weights. This technique is referred to as "MST with traffic update". They further enhance their tree by mapping some links of some shortest paths to the constructed trees: this will be referred to as the "enhanced MST" technique. When the link weights are not updated after constructing a tree, that is, each ST is constructed for a VPN by simply selecting a root randomly among the VPN's AP nodes, the technique is referred to as "MST without traffic update". We compared the results of our heuristic with those provided by the above algorithms and the single ST strategy, by implementing all algorithms.

For simplicity, networks have links which are all of equal capacity. We generate a fixed number of VPNs, having each a random set of APs in the network. Each VPN asks for a random amount of bandwidth, uniformly chosen betwen 0 and 10 MBps. This amount must be available between any 2 nodes of the VPN. $\Delta$ was set to 1 in this chart. The results presented here are average values of 100 simulations. The results are presented in Table 1, and give the statistics for maximum load on a link and number of trees used.

As the results show, our algorithm (*DFGM*) outperforms the other 4 as far as load balancing is concerned. For the VTHD network, the maximum load on a single link is 7.9% lower on average than the best concurrent algorithm (here, the "enhanced MST"). For the Italian network, the improvement is of 19.8% over the "MST with traffic update". We can also notice that for this large network which is not highly connected, the "enhanced MST" algorithm does not improve on the "MST with traffic update". Finally, for the dense network, our algorithm improves the "enhanced MST" algorithm by 25%.

When there are 30 VPNs (respectively 10), the "enhanced MST" and "MST with/without traffic updates" all use 30 (resp. 10) trees. The single ST uses a single tree to support all VPNs. However, our algorithm makes use of only 30% to 50% that number of trees. This does not have any meaning regarding traffic distribution, but it does mean a considerable label space savings (less VLAN-IDs). It also means there will be less signaling-related traffic, since there are fewer trees to maintain.

## 6 Conclusion

In this paper we tackled the problem of mapping VPNs to Spanning Trees in Metro Ethernet networks. We formulated this problem as a novel MINLP problem. However, due to tractability issues, we conceived a heuristic algorithm to solve the problem of optimally balancing the load of the VPNs accross the network, which is quite simple to implement by an operator. We compared the results of our heuristic to the other known methods of mapping VPNs to Spanning Trees which are available, and very good results were obtained. Further research on the subject is targeting the resiliency of the generated forests, to minimize the impact of a single failure.

## References

1. IEEE 802.1q, *Virtual Bridged Local Area Networks*, IEEE, 2003.
2. L. Berger, *Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description*, Internet RFC 3471, January 2003, IETF.
3. IEEE 802.1d, *Media Access Control Bridges*, IEEE, 1998.
4. IEEE 802.1w, *Rapid Spanning Tree Configuration*, IEEE, 2001.
5. IEEE 802.1s, *Multiple Spanning Trees*, IEEE, 2002.
6. M. Ali, G. Chiruvolu, A. Ge, *Traffic Engineering in Metro Ethernet*, IEEE Network, p.10-17 March/April 2005.
7. M. Padmaraj, S. Nair, M. Marchetti, G. Chiruvolu, M. Ali, A. Ge, *Metro Ethernet Traffic Engineering Based on Optimal Spanning Trees*, WOCN 2005, p.568-572.
8. X. He, M. Zhu, Q. Chu, *Traffic Engineering for Metro Ethernet Based on Multiple Spanning Trees*, ICN/ICONS/MCL 2006, p.97.
9. E. W. Dijkstra, *A note on two problems in connection with graphs* Numerische Mathematik volume 1, p. 8389, 1959.
10. CPLEX and AMPL optimization packages, ILOG, on-line, www.cplex.com
11. Ministère de l'Économie, des Finances et de l'Industrie de France, *Réseau National de Recherche en Télécommunications*, on-line, www.vthd.org
12. R. Sabella, E. Iannone, M. Listanti, M. Berdusco, S. Binetti, *Impact of transmission performance on path routing in all-optical transport networks*, IEEE JSAC vol.6, p. 1617-1622, Dec 1988.