# Online Virtual Links Resource Allocation in Software-Defined Networks

Mikael Capelle*†, Slim Abdellatif*†, Marie-José Huguet*† and Pascal Berthou*†
* CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
† Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France
‡ Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France

***Abstract*—Network virtualization is seen as a key networking paradigm for building diverse network services and architectures over a shared network infrastructure. Assigning network resources to virtual links and, more generally to virtual network topologies, efficiently and on-demand is one of the most challenging components of any network virtualization solution. This paper addresses the problem of on-line resource allocation of multiple virtual links on a Software Defined Network (SDN) infrastructure. The application context that is targeted is primarily the on-line provisioning of virtual overlay networks even if it can be broadened to address more general virtual networks. Considering an SDN physical infrastructure allows a complete freedom in choosing the optimal physical paths and the associated resources that support the virtual links with no interference from any other network function (such as routing). However, for the time being, forwarding in an SDN network is resource consuming with a noticeable impact on the size of the flow tables. Hence, forwarding (i.e. switching) resources should be carefully considered by the network resource allocation algorithm. This paper proposes a novel Integer-Linear formulation of the above cited problem by taking into account: (1) point-to-point as well as point-to-multipoint virtual links, each with an associated bandwidth requirement and a maximum transfer delay requirement, (2) two types of network resources, namely network links' bandwidth and nodes' switching resources, and (3) optionally, path splitting which allows a virtual link to be established on multiple physical paths. We report preliminary experimental results on a real network topology in an overloaded scenario (bandwidth requests largely exceed network capacity); they show that our algorithm outperforms shortest path heuristics with a gain on the admission rate (of virtual links requests) that ranges from 5 to 15% (compared to the most efficient heuristic) and with a computation time less than a few seconds.**

## I. Introduction

Network Virtualization enables the creation and coexistence of multiple isolated and independent virtual networks over a shared network infrastructure [1]. A virtual network is a logical (opposed to physical) network with some of its elements (nodes and links) being virtual. A virtual node is an abstraction of a network device that is often hosted on a single physical node. It executes network functions such as routing, forwarding, etc. by consuming part of the resources of the hosting node. The resources allocated to a virtual network device are as diverse as CPU, volatile memory, network interfaces, storage, switching, etc. Similarly, a virtual link is an abstraction of a network link that is established on one or multiple physical links or physical paths. It consumes transmission resources (i.e. physical links' bandwidth) as well as switching resources at the traversed physical nodes.

Network virtualisation is claimed to be an essential component of future Internet architecture where elastic, potentially programmable, virtual networks are provisioned on-demand [3]. The virtual network can be a fully virtualised network (with virtual nodes and links) or a simpler construct with virtual links connecting end nodes (which can be virtual) known as overlay networks. In data centres, these latter are considered as the mean to provide network services to applications and are gaining broad acceptance from academia and industry [9], [5], [4].

Virtual link resource allocation is a key component of an overlay network solution (and also of a fully virtualised virtual network solution). It determines the physical paths and the appropriate resources that are used to support a virtual link with predefined characteristics. On legacy networks, it is strongly coupled and dependent on the routing used by the physical network infrastructure which imposes the route(s) used to support the virtual link. With the advent of Software-Defined Networking (SDN) [2], a virtual link resource allocation method can be devised with no interference from any other network function. Indeed, on a physical SDN network infrastructure, flows and their forwarding behaviour can be defined at will allowing the virtual link resource allocation method to freely choose the physical paths to be used.

This paper proposes a Integer Linear Programming (ILP) formulation of the problem of resource allocation of multiple virtual links (that compose a virtual/overlay network) in SDN network infrastructures. The proposed method addresses point-to-point virtual links and, notably, point-to-multipoint virtual links (typically useful when providing network services to applications), each with an associated bandwidth requirement and a maximum transfer delay requirement. Two types of network resources are taken into account: classically, the links' bandwidth but also the switching resources of nodes. This is particularly important since, for the time being, switching in an SDN enabled node is a resource consuming task (because of the match operation (which allows the use of wildcards) and the instructions to execute) and this limits the size of the switching table to a few thousands of entries [6], [7]. A last characteristic of our method is the optional support of path splitting which allows a virtual link to be established on multiple paths. This capability contributes to the objective of the method which is: fairly spreading network load on network links and nodes in order to improve the admissibility of subsequent virtual network requests.

Even if virtual network resource allocation has recently at-

ISBN 978-3-901882-68-5 © 2015 IFIP

tracted lots of attention [10], to the best of our knowledge, the resource allocation problem as described above has not been investigated so far. Indeed, it is in the conjunction of all the above-cited constraints that resides the originality of the addressed problem (point-to-multipoint virtual links, accounting for switching resources, delay requirements and path splitting being the least considered in the literature).
Experimentations on a real network topology brought from the GÉANT network [8] in an overloaded scenario (virtual links requests largely exceed network capacity) show that our method outperforms shortest path heuristics with a gain on the admission rate that ranges from 5% up to 15% compared to the most efficient heuristic (and much more compared to the other heuristics). They also show that the computation time is satisfactory (less than a second for point-to-point links and a few seconds on average for point-to-multipoint virtual links) and that the contribution of path splitting is limited (2 to 3%) for the considered scenario.

This paper is organised as follows. Section II reviews previous work from the literature that are related to virtual network resource allocation. Then, section III describes the mathematical formulation of the problem that we are addressing. Section IV presents the performance analysis of the proposed method and its comparaison to the considered shortest path heuristics. Finally, section V concludes the paper and describes the perspectives to this work.

## II. Related Work

Virtual network resource allocation (or embedding) has attracted a lot of attention from the research community during the last few years [10]. Several optimization approaches were proposed, some follow an integrated strategy considering simultaneously the resource embedding of virtual nodes and links and, most, follow a split approach that considers separately and successively the two resource embedding problems. Whatever the approach, virtual link resource allocation is the core component of any virtual network resource embedding method.
Without being exhaustive, virtual link resource allocation methods can be classified with respect to the following criteria. Some are related to the characteristics of the virtual links being considered, namely: the type which is usually point-to-point but can also be point-to-multipoint, the Quality of Service (QoS) requirements with possibly a bandwidth, a delay and/or a loss rate requirement. The others are related to some features of the methods such as: (1) the allocation process which can be online or offline (the requests are fully known in advance); (2) the general class to which a method belongs which can be heuristic or exact; (3) the followed approach which leads either to a stand-alone method exclusively concerned with virtual link resource embedding or a more general method that integrates and combines virtual node and virtual link resource allocation, (4) the network resources that are concerned by the allocation which can be the bandwidth of physical links and, possibly, the switching resources of nodes[1] (that are needed to forward the packets that belong to a virtual link); and (5) the potential support of techniques that contribute to improve the efficiency of the methods, such as path splitting and/or migration which allows the reallocation of resources to already admitted virtual links. Table I summarises the existing work according to these criteria (where P2P, P2M, BW and ILP respectively stand for point-to-point, point-to-multipoint, bandwidth and integer-linear Programming). It also classifies the method that we are proposing in this paper.

[1]other node resources (typically, CPU and RAM) may be allocated to virtual nodes by the virtual node resource embedding algorithm

TABLE I: Classification of virtual link resource allocation methods

| | VL type | VL QoS | allocation process | method | network resources | supported techniques |
|---|---|---|---|---|---|---|
| [15] | P2P | BW | offline | integrated | link | |
| [14] | P2P | BW delay | online | integrated | link | |
| [16] | P2P | BW | online | stand-alone | link | path-split migration |
| [11] [12] | P2P | BW | online | stand-alone | link | |
| [17] | P2P P2M | BW | offline | stand-alone | link | |
| [18] | P2P | BW | online | stand-alone | link | |
| [19] | P2P P2M | BW | offline | stand-alone | link | |
| [21] | P2P | BW | online offline | integrated | link | path-split |
| [13] | P2P | BW delay | online | integrated | link | |
| [22] | P2P | BW | offline online | integrated | link | |
| our met. | P2P P2M | BW delay | online | stand-alone | link node | path-split |

Among this list of papers, we point at the works of [13], [22], [21] which address an allocation problem similar to the one addressed in this paper while using the same class of methods (based on an ILP or Mixed ILP formulation). In [13], the authors proposed an integer linear model for solving the on-line virtual network embedding problem (virtual nodes and links). Several constraints were considered such as bandwidth, delay, node load (CPU, RAM) and their method aimed at minimizing the resource consumption. The proposed method was compared with several heuristics approaches on theoretical physical network topologies.
In [22], an integer-linear model was proposed to consider the embedding of virtual network requests. The constraints taken into account, and evaluated in their paper, are the capacity of nodes, in terms of CPU, and the bandwidth on links. Moreover, a virtualization request which is composed of a set of subgraphs is considered to be satisfied if at least one subgraph can be embedded in the physical network. The aim is to maximize the revenue (which is dependent from the allocated CPU) while minimizing resource consumption, i.e. bandwith utilization. Their method can be either applied online or offline as it can consider simultaneously a set of virtual requests leading to better solutions.
In [21], the authors consider the embedding of virtual networks on several physical network infrastructures. This problem is solved in two steps. The first one focuses on the selection of the physical network and the second step is dedicated to the embedding of a virtual network (with virtual nodes and links) on the selected physical network. A mixed integer linear model is proposed that aims at minimizing the load on the

physical network. This model is evaluated for on-line virtual requests and also for optimizing the embedding of a set of virtual requests. The considered constraints are the capacity of each node and the bandwidth of each link. To the best of our knowledge, this is the first paper presenting an integrated approach for the virtualization of both nodes and links. A worth noting point is that the flow variables used in their model are assumed to be real numbers leading to a relaxation of the optimization problem.

## III. Problem formulation

This section describes the ILP mathematical formulation that we propose to solve the online virtual links resource allocation problem. Requests arrive and are treated in sequence with no information on future requests (i.e. online). Each request gathers multiple concurrent virtual link sub-requests, each concerning a point-to-point or point-to-multipoint virtual link with bandwidth and delay requirements. The objective is to distribute fairly network traffic and use efficiently network resources (transmission and switching).
Below, the physical network and virtual links request models are described; Then, the variables and problem constraints are listed; Lastly, the considered objective function is defined.

### A. Physical Network Model

The physical network is modelled by a bidirectional graph $G = (V, E)$ where $V$ $(|V|)$ is the set of physical nodes (SDN switches) and $E$ $(|E|, E \subseteq V \times V)$ the set of physical links which operate in full-duplex mode. To each node $i \in V$ is associated a switching capacity $U_i$ which is the maximum number of entries (i.e. size limit) of its flow table. The current size of node $i$ flow table is denoted by $U'_i$. Each link $(i, j), i, j \in V$ is weighted by its bandwidth $B_{ij}$ and its propagation delay $D_{ij}$. Links are assumed to exhibit the same characteristics in both directions, i.e. $B_{ij} = B_{ji}$ and $D_{ij} = D_{ji}$. The bandwidth that is currently assigned at link $(i, j)$ by already admitted virtual links is denoted by $B'_{ij}$.

### B. Virtual Links Requests Model

A virtual links request consists of a set of $K$ virtual links. Each virtual link $k$ is characterised by:

- a source node $s_k \in V$, and a set of destination nodes $T_k \subseteq V - \{s_k\}$ (when $|T_k| = 1$, the virtual link is point-to-point, otherwise it is point-to-multipoint);
- a bandwidth requirement of $b_k \in \mathbb{N}^*$ and a maximum transfer delay of $d_k \in \mathbb{N}^*$;
- a maximum packet size of $p_k$.

### C. Resource-related assignment variables

The output of our assignment problem is the set of routes (with the bandwidth allocations at each supporting physical link and the number of flow table entries at each traversed node) that support each of the virtual links that compose a request. It is worth noting that since virtual links may be point-to-multipoint, it is likely that resource allocations will be mutualised close to the source and as we get closer to destinations, they will tend to be more and more dedicated to specific destinations. As a consequence, basic assignment variables are related to a specific destination of a virtual link. In our model, we distinguish the following variables :

- $f_k^t(i, j)$ is an integer variable that represents the bandwidth allocated at link $(i, j)$ to the packets of virtual link $k$ that are flowing from the origin node $s_k$ to a destination node $t$. More generally, $f_k(i, j)$ refers to the amount of bandwidth used on link $(i, j)$ by the virtual link $k$, whatever the destination. It is set to the maximum of $f_k^t(i, j)$ for all $k \in K$.
- $l_k(i)$ is an integer variable that indicates the switching resources consumed by virtual link $k$ at node $i$. It is expressed as the number of entries that are installed in node $i$ flow table to support virtual link $k$ with the assumption that all entries consume the same amount of resources regardless of the complexity of the *match* operation and the related *instructions* to perform. The number of flow table entries is assumed to be in proportion to the number of node $i$ ports that are receiving traffic from virtual link $k$ (as given by equation 1).

$$\forall k \in K, \forall i \in V : l_k(i) = \sum_{\substack{j \in V \\ (j,i) \in E}} g_k(j, i) \tag{1}$$

  where $g_k(i, j)$ is an intermediate boolean variable that indicates if some bandwidth from link $(i, j)$ is assigned to virtual link $k$ or not. It is derived from another set of more focused intermediate variables $g_k^t(i, j)$ that reflects whether the flow of packets of virtual link $k$ destined to $t$ is supported by the physical link $(i, j)$ (i.e. $g_k^t(i, j) = 0$ if $f_k^t(i, j) = 0$ and $g_k^t(i, j) = 1$ otherwise).
- $d_k^t(i)$ which is an upper bound on the time needed for a packet of virtual link $k$ destined to $t$ to reach node $i$ from its originating node $s_k$.
- $f_{max}$ which refers to maximum link utilization (when considering all network links) after request acceptance (i.e. by taking into account the bandwidth allocations consumed by the virtual links that compose the request).
- $u_{max}$ which similarly refers to maximum flow table utilization (when considering all network nodes) after request acceptance.

### D. Problem Constraints

The constraints on bandwidth allocations are described hereafter in equations 2 to 8. Equation 2 reflects the linearisation of the $Max$ operator applied to variables $f_k^t(i, j)$ to get $f_k(i, j)$. Equations 3 and 4 have a similar purpose and focus respectively on $f_{max}$ and $u_{max}$ which are minimized by the objective function (as explained below).

$$\forall k \in K, \forall (i, j) \in E, \forall t \in T_k : f_k^t(i, j) \leq f_k(i, j) \tag{2}$$

$$\forall (i, j) \in E : \frac{1}{B_{ij}} * \left( B'_{ij} + \sum_{k \in K} f_k(i, j) \right) \leq f_{max} \tag{3}$$

$$\forall i \in V : \frac{1}{U_i} * \left( U'_i + \sum_{k \in K} l_k(i) \right) \leq u_{max} \quad (4)$$

Equation 5 ensures that the bandwidth assigned to each virtual link $k$ at link $(i,j)$ does not exceed the remaining bandwidth. Equation 6 is the usual flow conservation constraints.

$$\forall (i,j) \in E : \sum_{k \in K} f_k(i,j) \leq B_{ij} - B'_{ij} \quad (5)$$

$$\forall k \in K, \forall t \in T_k, \forall i \in V :$$

$$\sum_{j \in \Gamma(i)} \left( f^t_k(i,j) - f^t_k(j,i) \right) = \begin{cases} b_k & if\ i = s_k \\ -b_k & if\ i = t \\ 0 & else \end{cases} \quad (6)$$

Equation 7 is a channeling constraint between integer and boolean variables: $f_k(i,j)$ and $g_k(i,j)$. It also constrains the virtual link $k$'s bandwidth assignment at a physical link to the requested bandwidth $b_k$. Equation 8 constrains the bandwidth that is assigned to the flow of packets destined to a specific virtual link's end-point (or destination) within a range of values, in addition to establishing a channeling constraints between boolean and integer variables. The inequalty on the right side ensures that the bandwidth requirement of the virtual link is never exceeded. The inequalty on the left side directs path-spliting and avoids the multiplication of splits with low bandwidth allocations. Indeed, if active, path-splitting is feasible only if the bandwidth allocated to the splits respects a minimum threshold $b^{min}_k$. In practice, $b^{min}_k$ is a ratio of $b_k$, $b^{min}_k = PS_{ratio} * b_k$ with $PS_{ratio} \in [0,1]$ (then, $PS_{ratio} \leq 0.5$ when the path-splitting is allowed, and $PS_{ratio} = 1.0$ when it is forbidden).

$$\forall k \in K, \forall (i,j) \in E :$$
$$g_k(i,j) \leq f_k(i,j)\ and\ f_k(i,j) \leq b_k * g_k(i,j) \quad (7)$$

$$\forall k \in K, \forall (i,j) \in E :$$
$$b^{min}_k * g^t_k(i,j) \leq f^t_k(i,j)\ and\ f^t_k(i,j) \leq b_k * g^t_k(i,j) \quad (8)$$

The constraint related to switching resource allocations is given by inequalty 9. It simply ensures that with the addition of flow table entries needed by the virtual links composing the request, the size of network nodes' flow tables remains below their maximum size.

$$\forall i \in V : \sum_{k \in K} l_k(i) \leq U_i - U'_i \quad (9)$$

Virtual links have end-to-end delay requirements. For point-to-multipoint links, they hold and must be met for all of their end-points (destinations). These requirements are taken into account by inequalities 10 and 11 which establish a set of constraints on variables $d^t_k(i)$ (defined in section III-C as an upper bound on the time needed for a packet of virtual link $k$ destined to $t$ to reach node $i$ from its originating node $s_k$). The expressions in 10 assure the initialisations at the sources (of virtual links) and that the computed upper bounds at the destinations meet the delay requirements. Inequaltiy 11 show how the upper bounds, computed at two subsequent nodes along the route to a destination, are related to account for packet transmission times and physical link propagation delays. More clearly, let us consider the flow of packets belonging to virtual link $k$ destined to $t$ and let us assume that this flow of packets really goes through link $(i,j)$. Then, the expression in 11 simply states that the increment to the upper bound on the packet transfer delay from $s_k$ to $i$ in order to get the upper bound at $j$ must at least take into account : the transmission time of a packet of maximum size $p_k$ at the lowest possible bandwidth (i.e. $b^{min}_k$) and the propagation delay along link $(i,j)$. The last term of the left-side of inequality 11 is a hint used to neutralise its effect in case link $(i,j)$ is not traversed by the flow of packets ($\Psi$ being large, when $g^t_k(i,j) = 0$, the inequality becomes trivial).

$$\forall k \in K, \forall t \in T_k : d^t_k(s_k) = 0\ and\ d^t_k(t) \leq d_k \quad (10)$$

$$\forall k \in K, \forall t \in T_k, \forall (i,j) \in E :$$
$$d^t_k(i) + \frac{p_k}{b^{min}_k} + D_{ij} - \Psi * (1 - g^t_k(i,j)) \leq d^t_k(j) \quad (11)$$

## E. Objective function

The objective function aims at minimizing link and node (switching) resource consumption but also at distributing the consumed resources among nodes an links in order to reduce the creation of bottlenecks. Both contribute to improve the admissibility of forthcoming requests.
As shown in expression 12, it consists of four components, each weighted with a parameter that controls the impact of the component on the resolution process. The first two concern bandwidth allocations and the last two are their analogue for flow table entries (or switching) allocations. We focus hereafter on the former expressions.

$$\begin{aligned} \text{minimize } & \alpha_1 * \frac{1}{|E|} * \sum_{(i,j) \in E} \left( \frac{1}{B_{ij}} * \left( B'_{ij} + \sum_{k \in K} f_k(i,j) \right) \right) + \\ & \alpha_2 * f_{max} + \\ & \beta_1 * \frac{1}{|V|} * \sum_{i \in V} \left( \frac{1}{U_i} * \left( U'_i + \sum_{k \in K} l_k(i) \right) \right) + \\ & \beta_2 * u_{max} \end{aligned} \quad (12)$$

In the first term, the aspect that is minimized is the average of utilization of network links after allocating bandwidth to the considered request. In the second term, the aspect that is minimized is the maximum of network links utilization. This means that the allocations devoted to the request are distributed over different links in such a way that links load disparity is reduced. This is mainly (but not only) enabled by path-splitting which often has as a side effect the increase of node and link resource consumption.

## IV. Experimental results

The main objectives of the experimental analysis are:

- evaluate the performance of our proposed method from the point of view of network resource utilisation, virtual links requests admissibility and convergence (or computation) time.

- gain comprehension in the behaviour of our method and more precisely, assess the effect of some of its parameters: $\alpha_i, \beta_i, i : 1, 2$ and path splitting ($PS_{ratio}$)
- compare its performance with respect to different variants of shortest path heuristics. Indeed, an objective comparison with the work cited in section II is not obvious because of the disparity of each solution's features (refer to table I).

## A. Simulation model

To conduct our experiments, our algorithm is applied to a real network topology with different randomly generated virtual links requests. The considered experimental set up is described hereafter.

*1) Network model:* We consider in this work a real network topology taken from the European Research Network GEANT[2] (January 2014) with 41 network nodes and 60 links that connect the main European cities. We adopt the link capacities that are given by GEANT. For some links the exact capacity is not given, instead, an interval is provided. For those whose capacity falls within the range of 1 to $10Gbps$, the capacity is set to $5Gbps$. For those whose capacity ranges from 10 to $100Gbps$, the capacity is set to the uppermost value. Link propagation delays are derived from the geographic coordinates of nodes assuming a straight line between the involved nodes and light speed propagation. Finally, we assume that a flow table size of 2000 entries is dedicated at each node to the considered virtual links resource allocations.

*2) Load model:* The load model defines the types of virtual links requests and their characteristics as well as their arrival and lifte-time processes. Two types of virtual link requests are considered :

- Unicast Requests which are made of one single point-to-point virtual link request. The bandwidth (respectively delay) requirement is randomly chosen between 7 and $12Gbps$ (resp. 50 to $80ms$). Unicast Requests target network service providers or virtual network operators to answer the need to connect some of their networking devices;
- Multicast Requests which are made of a set of multiple point-to-multipoint virtual links. The number of virtual links is randomly chosen between 4 and 6. Each point-to-multipoint virtual link has from 4 to 6 endpoints. The bandwidth requirements are fixed on a virtual link basis and is chosen randomly from $200Mbps$ to $3Gbps$. Similarly to unicast requests, the delay requirements range from 50 to $80ms$. This kind of requests target application service providers such as over-the-top players or content distribution networks to connect some of their elements (cache servers, etc.).

For both types of requests, source and destination selection is performed on a random basis, the probability that a node is chosen is proportional to its overall bandwidth capacity (i.e. when considering all its incoming/outgoing links).
As for previous papers [11], [13], the request arrivals follow a Poisson process with an arrival rate $\lambda$ that is varied on $\{0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 1.0\}$, i.e. the average number of requests varies from 4 to 10 requests each 100 units of time ($UT$). The request life-time conforms to an exponential distribution with an average of 1000 UT.
It is important to note that with this load model, our experimental analysis is considering an overloaded scenario where the cumulative bandwidth requests exceed network capacity. For instance, on average, the GÉANT edge nodes which are connected to the backbone via physical links with a bandwidth less than $10Gbps$ are fully saturated with no available bandwidth by two hundreds point-to-multipoint virtual link requests.

*3) simulation settings:* The Integer Linear model was implemented in C++ with CPLEX$-12.6$ solver[3]. The experiments were carried out on an Intel Core i7-4770, 3.4GHz, 8GB of RAM and running Linux Ubuntu-12.04. The resolution time is set to a maximum of 15 seconds. A gap of less than $5\%$ to the optimal solution is considered satisfactory.
The simulation horizon is fixed to $10000UT$ (this time period is sufficient to have our method in the stationary regime). Parameters $\alpha_1$ and $\alpha_2$ of the objective function are set to 1 ($\alpha_1 = \alpha_2 = \alpha = 1$). Parameters $\beta_1$ and $\beta_2$ are merged and are varied to tune the relative importance of minimizing and distributing swichting resources consumption with respect to bandwidth resources in the search of the optimal solution ($\beta_1 = \beta_2 = \beta \in \{1, 50, 100, 150\}$). $PS_{ratio}$ is also varied within $\{10\%, 33\%, 45\%\}$ and $100\%$ (path-splitting is inhibited).

*4) Performance metrics:* The following performance metrics are computed during simulation for performance analysis purposes:

- Acceptance rate: the percentage of successful virtual links requests out of all the requests that arrived during the simulation time;
- Convergence time: the time needed by our method to compute the optimal allocations associated to a virtual links request. The average convergence time and the maximum convergence time are computed over the number of successful requests;
- Instantaneous link utilisation: the percentage of assigned bandwidth at a given link, computed at a time instant $t$, i.e. $\frac{B'_{ij}}{B_{ij}}$ for link $(i, j)$. The instantaneous average link utilisation is averaged over the set of network links as follows

$$\frac{1}{|E|} * \sum_{(i,j)\in E} \frac{B'_{ij}}{B_{ij}} \quad (13)$$

## B. Preliminary results and analysis

*1) Virtual Links assignment method general performance:* The goal is to give an insight into the general performance of our proposed method by describing the evolution of the requests acceptance rate and network resource utilisation when varying the requests arrival rate and the request type (unicast, multicast). Without any loss of generality, $\beta$ and $PS_{ratio}$ are respectively set to 150 and 45%, the presented results are representative of what we observed with different settings.

[2]http://www.geant.net/

[3]http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

Figures 1 and 2 describe the instantaneous acceptance rate as a function of time with different requests arrival rates ($\lambda$). As explained above, we are considering an overloaded scenario with cumulative bandwidth requests that exceed network capacity. The effect is a request acceptance rate that falls sharply away from full acceptance when the arrival rate $\lambda$ is increased. This is particularly noticeable for multicast requests. The reason is that for a given arrival rate, the bandwidth requirement of a multicast request is much more important than its corresponding requirement for a unicast request. Indeed, a multicast request is made of 4 to 6 point-to-multipoint virtual links, each with 4 to 6 end-points while unicast requests are made of one single point-to-point virtual link. Even if the bandwidth requirement of a point-to-point virtual link is more important than its corresponding in a point-to-multipoint virtual link, the overall bandwidth requirement of a multicast request measured at destination nodes is on average four times as important. Another worth noting point is that a multicast request is declined when the network resources needed by at least one of its virtual links can not be afforded. In other words, if one virtual link out of a set of six virtual links that compose a multicast request can not be established, then the whole request is refused. This is the reason why the acceptance rate for multicast request falls down to 40% for some values of $\lambda$.

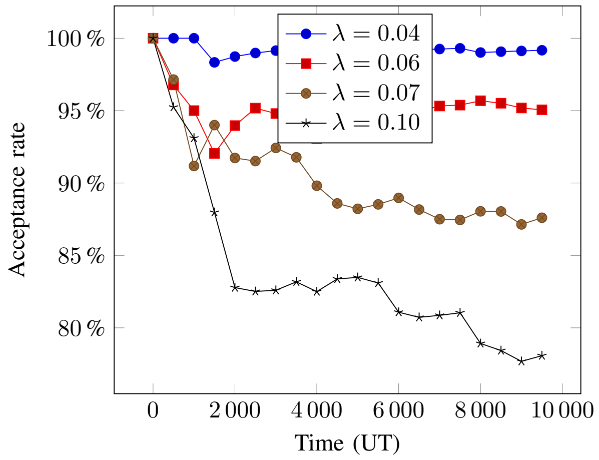

Fig. 1: Acceptance rate - Unicast Requests

Figures 3 and 4 respectively show the associated instantaneous average link utilisation (of expression 13) for unicast and multicast requests. Despite the overload, the average link utilisation remains below 45% which is quite low in view of the refusal rate. A more refined view is presented in figure 5 which shows the distribution function of instantaneous link utilisations measured at the end of the simulation for multicast requests. It clearly shows that most links have less than half of their bandwidth assigned and that around 20% of links have more than 80% of their bandwidth already assigned. A more detailed analysis on the data sets shows that a few backbone links have more than 95% of their bandwidth assigned. They are the major cause of request refusals. These results brings some evidence on the efficiency of the allocations in terms of consumption and distribution although it may seem that our method failed in avoiding the overload of some backbone links.

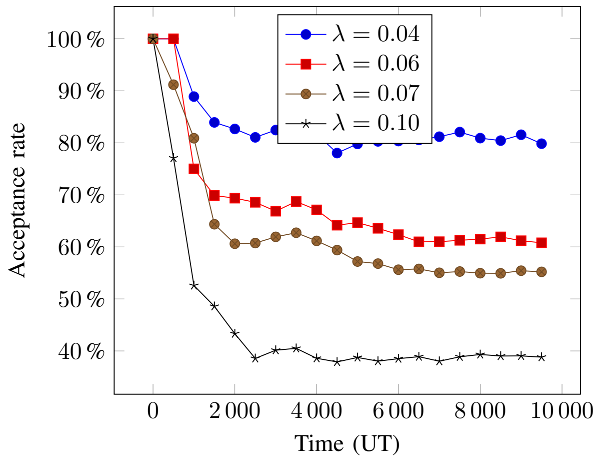

Fig. 2: Acceptance rate - Multicast Requests

In fact, this must be tempered with the request loads scenario (requirements >> network capacity) that we are considering as well as with the network topology of GEANT which binds most allocations to backbone links.

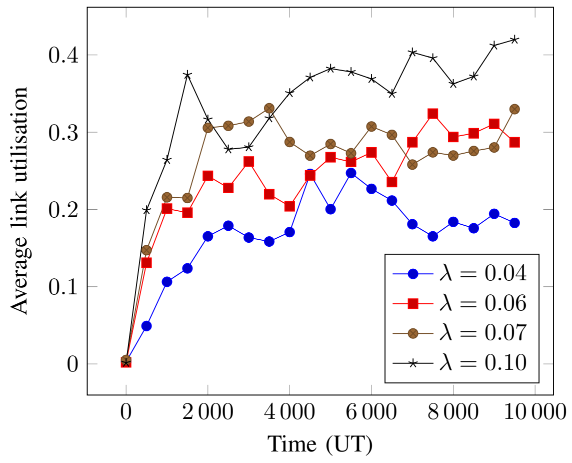

Fig. 3: Average link utilization - Unicast Requests

2) *convergence time:* The goal is to evaluate the convergence time of our method when treating a unicast or a multicast request and contrast it to its application scenario which is the on-demand provisioning of virtual links or overlay networks. We believe that a provisioning time that falls within a few tens of seconds is a reasonable target. So is the target of the convergence time of the resource allocation method that precedes the real deployment of the virtual links.

Figure 6 shows the average convergence time of our method for unicast requests as a function of requests arrival rates with and without the use of path splitting. When path splitting is used, the $PS_{ratio}$ leading to the largest search space and hence to the largest convergence time is chosen (i.e. $PS_{ratio} = 10\%$). Figure 6 also shows the maximum observed convergence time for a successful request. These results are obtained with a value of $\beta = 100$, but again, they are representative of what was observed with other settings. For the considered network and load model, the convergence times associated to unicast

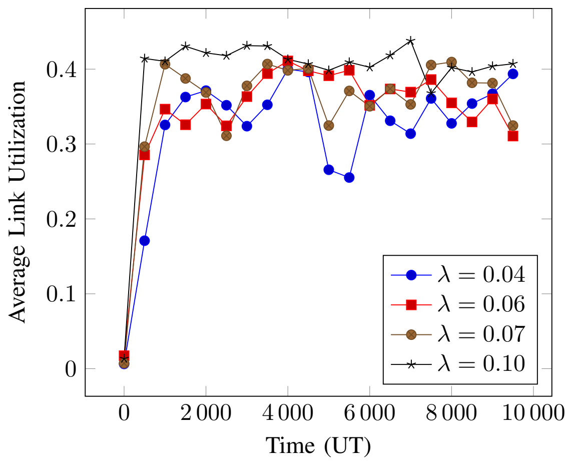

Fig. 4: Average link utilization - Multicast Requests

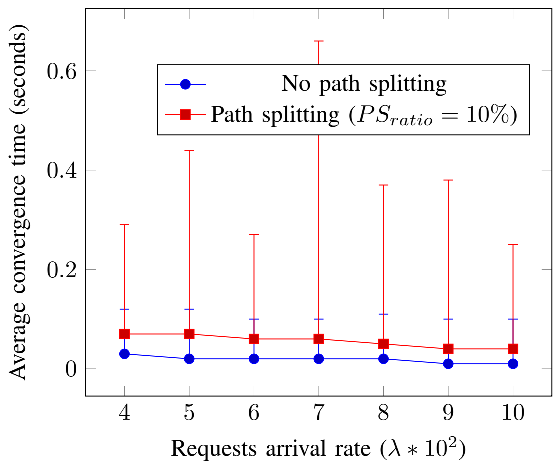

Fig. 6: Convergence time - Unicast requests

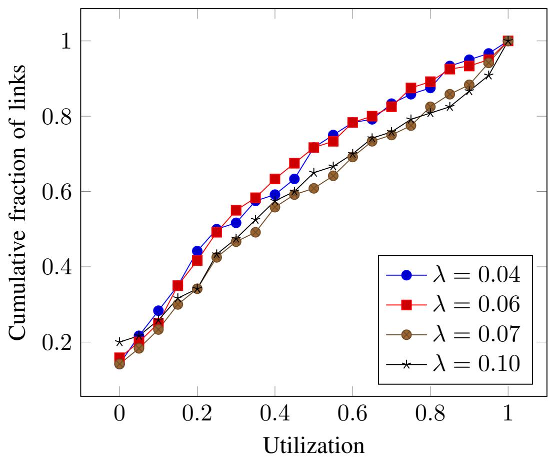

Fig. 5: Link utilisations' distribution function- Mulicast requests

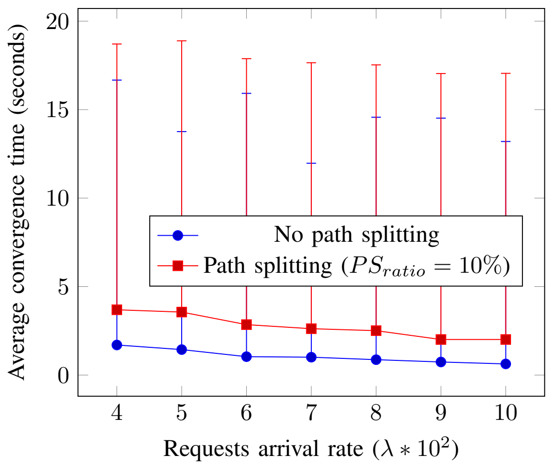

Fig. 7: Convergence time - Multicast requests

requests fully meet the requirements described above. Indeed, the average convergence time is below $100ms$ while the maximum remains below $1s$ (more precisely, $200ms$ without path splitting and $700ms$ with path splitting).

Figure 7 describes the results for multicast requests. The observed convergence times are satisfactory since they remain on average below $1s$ when path splitting is inhibited and below $4s$ when path splitting is active. However, for some very few requests, the convergence time exceeds the maximum resolution time of $15s$ (refer to section IV-A3). At the time of this writing, the network conditions and the request profiles that lead to such convergence times are not identified yet. Remakably, path splitting has a significant impact on the convergence time of our method.

*3) Path splitting and $\beta$ parameter impact:* The goal is to evaluate the impact of parameters $PS_{ratio}$ and $\beta$ on the behaviour of our method. To that end, parameters $PS_{ratio}$ and $\beta$ are respectively varied over $\{10\%, 33\%, 45\%, 100\%\}$ and $\{1, 50, 100, 150\}$ and the request acceptance rates are measured both, for unicast and multicast requests.
No clear impact of parameter $\beta$ is observed from the experiments for unicast but also for multicast requests (as shown in figures 8 and 9 for unicast requests with $\beta = 1$ and $\beta = 150$). For space reasons, the results related to the other values of $\beta$ and to multicast requests are omitted, but the findings are the same. In fact, the load model that is considered for the experiments does not stress the switching resources and the impact of $\beta$ which weights the importance of the switching resource in the selection of the optimal allocation remains marginal. It is clear that additional experiments with different load models are needed to have a clear understanding on the impact of parameter $\beta$.

Path splitting does bring some gain in the admission rate even if, for the considered overloaded scenario, the gains remain within a few percents : 2 to 3% for unicast requests (Refer again to figures 8 and 9) and 1 to 2% for multicast. In a less loaded scenario, its impact on the performance would be stronger. The choice of the most efficient $PS_{ratio}$ is not clear even if a $PS_{ratio} = 10\%$ often exhibit good admission rates. The activation of path splitting is questioned for the considered scenario since the limited gain it brings is balanced with the increase in convergence time of the method as well as in flow tables sizes. Moreover, it introduces desequencing of packet

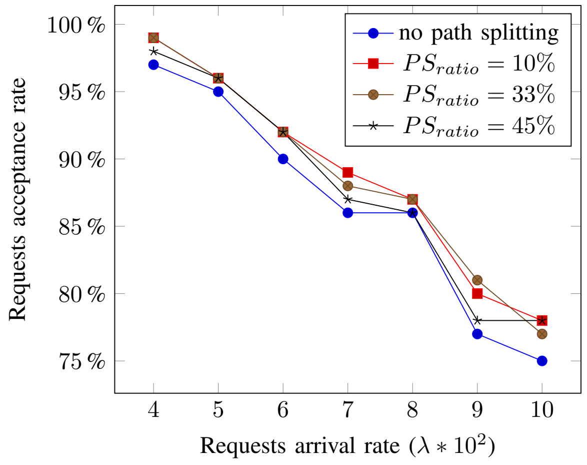

Fig. 8: Unicast requests acceptance rate for $\beta = 1$

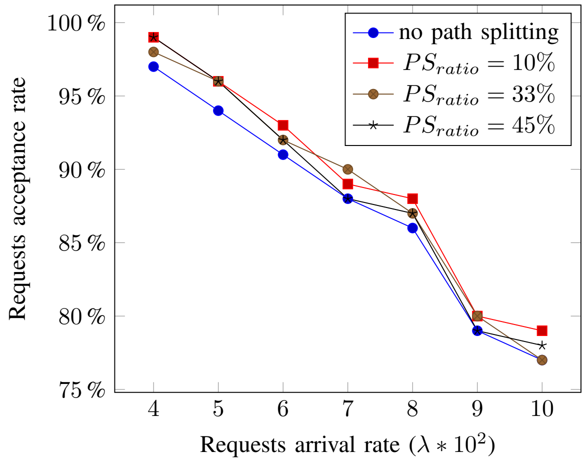

Fig. 9: Unicast requests acceptance rate for $\beta = 150$

transmissions. Additional experiments are needed to specify the situations where path splitting deserves being activated.

*4) Comparison with shortest path heuristics:* The goal is to compare the performance of our method with three Shortest Path (SP) heuristics from a requests admissibility perspective. SP heuristics use a cost function to assign a cost to each physical link. For each couple of end-points that belong to a virtual link, the physical path with the minimum cost (the cost of a path being the sum of the costs of the traversed links) is chosen. If the bandwidth available on the chosen path is below the bandwidth requirement of the virtual link, the corresponding request is not admitted. The three SP heuristics differ in their cost function. The first heuristic $SP_1$ uses a unitary cost for each link meaning that path selection is based on the lowest hop counts. In the second heuristic $SP_2$, the cost of a link is inversely proportional to its bandwidth meaning that the selection favours paths with the highest bandwidth. Finally in heuristic $SP_3$, the link's cost is inversely proportional to its current capacity. Paths with the highest available bandwidth are firstly chosen. It is worth noting that the three heuristics do not care about switching resources. Hence, some of the allocations derived from these heuristics may not be feasible if the induced size of each flow table is taken into account.

Figures 10 and 11, present the acceptance rates respectively for unicast and multicast requests. As expected our method have the best admission rate. When compared to $SP3$, the gain ranges from 3 to 15% (for multicast request when $\lambda = 0,04$). The gains are more important with $SP2$ and significantly more important with $SP1$. We believe that under less loaded scenarios, our method will exhibit much better performance. Indeed, by increasing the load beyond a certain limit, more and more request are declined whatever the allocation method. The performance of these methods in terms of admission will tend to get closer, to each other. This is exactly what we observe in figures 10 and 11 when $\lambda$ increases.

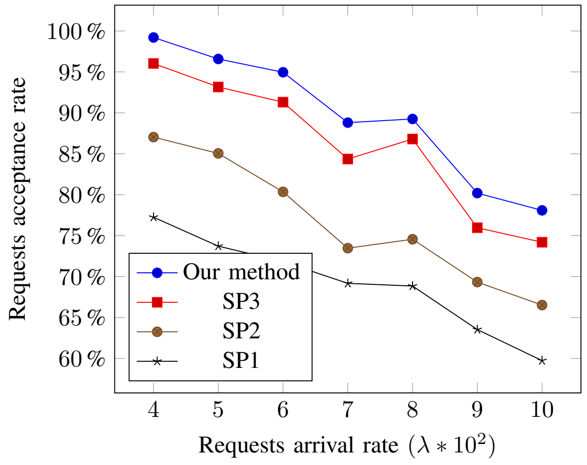

Fig. 10: Comparison with SP heuristics - Unicast requests

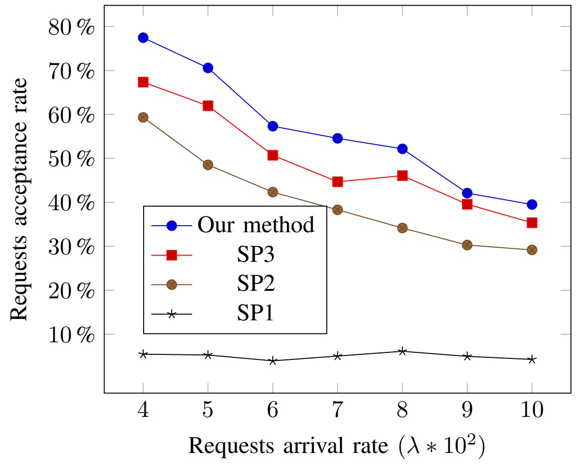

Fig. 11: Comparison with SP heuristics - Multicast requests

## V. Conclusion

In this paper, we developed an Integer-Linear programming method for the resource allocation of multiple virtual links. Compared to existing work from the literature, its contribution lies in the conjunction of the following features: (1) the support of point-to-multipoint virtual links in addition to point-to-point virtual links, (2) the support of virtual links' delay requirements in addition to bandwidth requirements, (3) the consideration of switching resources in the allocation of resources in addition to the bandwidth of links, and (4) the support of path splitting.

Our method was evaluated on a real network topology under high loads (which is not in favour of our method). The analysis showed that our method achieve efficient resource utilisation with a relative fair distribution of the load among nodes and links. The convergence time of our method was also assessed and it proved to be satisfactory. Our method was also compared with three shortest path heuristics and it exhibited better performance in terms of virtual links requests admissibility.

Additional experiments are however needed to cover other load models and other network topologies. This is necessary to get a more exhaustive view on the performance of our method and a more precise understanding on the impact of path splitting and parameter $\beta$ (which sets the relative importance of switching resource allocation in comparison to link resources in the optimization problem) of the objective function. The main goal is to derive guidelines on how to tune these parameters to the situation (i.e. the considered virtual links request, the network topology and current network resource usage).

Other perspectives to this work concern the extension of our ILP model. A first direction is to devise a more precise modelling of the switching resources consumed by each virtual link that takes more precisely into account the resources consumed by the SDN matching operations and the associated instructions. A second direction is the inclusion of resource migration in our model in order to have a more efficient and potentially adaptive method. A last perspective concerns the implementation of our method with an effective deployment of virtual links on a real SDN/Openflow network infrastructure. A technique that, in the one hand, defines how virtual links are classified and identified in SDN/Openflow switches and, on the other hand, ensures the resource allocations of virtual links has to be devised with scalability as the major design goal.

## Acknowledgment

This work was partially funded by the French National Research Agency (ANR) and the French Defense Agency (DGA) under the project ANR DGA ADN (ANR-13-ASTR-0024) and by European Union's Horizon 2020 research and innovation programme under the ENDEAVOUR project (grant agreement 644960).

## References

[1] Chowdhury, N.M.M.K. and Boutaba, R. *Network virtualization: state of the art and research challenges*, IEEE Communications Magazine, vol. 47(7) , pp 20-26, July 2009.

[2] ONF Market Education Committe, *Software-Defined Networking: The New Norm for Networks*, Open Networking Foundation, 2012

[3] Recommendation ITU-T Y.3011, *Framework of network virtualization for future networks*, ITU, 2012

[4] T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, P. Ingram, E. Jackson, A. Lambeth, R. Lenglet, S.-H. Li, A. Padmanabhan, J. Pettit, B. Pfaff, R. Ramanathan, S. Shenker, A. Shieh, J. Stribling, P. Thakkar, D. Wendlandt, A. Yip, and R. Zhang, *Network virtualization in multi-tenant datacenters*, in 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), pp. 203-216, Apr. 2014.

[5] S. Racherla, D. Cain, S. Irwin, P. Ljungstrom, P. Patil, and A. M. Tarenzio, *Implementing IBM Software Defined Network for Virtual Environments*. IBM RedBooks, May 2014.

[6] Bruno Astuto A. Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti *A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks* Communications Surveys & Tutorials, IEEE vol. 16(3) 1617 - 1634, 2014

[7] Yossi Kanizo, David Hay, and Isaac Keslassy, *Palette: Distributing tables in software-defined networks*, IEEE INFOCOM, 545-549, 2013

[8] *GEANT: the pan-European research and education network*, available at : http://www.geant.net/Network/The_Network/Pages/default.aspx. Accessed: 2014-11-22.

[9] Raj Jain and Subharthi Paul, *Network Virtualization and Software Defined Networking for Cloud Computing - A Survey*, IEEE Communications Managzine, pp. 24-31, Nov 2013.

[10] Abdeltouab Belbekkouche, Md. Mahmud Hasan, and Ahmed Karmouch *Resource Discovery and Allocation in Network Virtualization*, IEEE Communications Surveys and Tutorials 14(4):1114-1128, 2012

[11] Chowdhury, N.M.M.K. and Rahman, M.R. and Boutaba, R., *Virtual Network Embedding with Coordinated Node and Link Mapping*, IEEE-INFOCOM'09, pp. 783–791, April 2009.

[12] Chowdhury, N.M.M.K. and Rahman, M.R. and Boutaba, R. *ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping* IEEE/ACM Transactions on Networking, vol 20, pp. , 206-219, 2012.

[13] Melo, M. and Sargento, S. and Killat, U. and Timm-Giel, A and Carapinha, J., *Optimal Virtual Network Embedding: Node-Link Formulation*, IEEE Transactions on Network and Service Management, vol. 10(4), pp. , 356–368, **(2013)**.

[14] Lischka, Jens and Karl, Holger *A Virtual Network Mapping Algorithm Based on Subgraph Isomorphism Detection*, ACM VISA'09, pp. 81–88, 2009

[15] J. Nogueira, M. Melo, J. Carapinha, and S. Sargento. *Virtual network mapping into heterogeneous substrate networks*. In IEEE Symposium on Computers and Communications (ISCC), pp. 438-444, June 2011

[16] Wu-Hsiao Hsu, Yuh-Pyng Shieh, Chia-Hui Wang, and Sheng-Cheng Yeh. *Virtual network mapping through path splitting and migration*. In 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp.1095-1100, March 2012.

[17] Christian Frei and Boi Faltings. *Resource allocation in networks using abstraction and constraint satisfaction techniques*. In Principles and Practice of Constraint Programming (CP'99), volume 1713 of Lecture Notes in Computer Science, pp. 205-218, 1999

[18] Yongtao Wei, Jinkuan Wang, Cuirong Wang, and Xi Hu. *Bandwidth allocation in virtual network based on traffic prediction*. In 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM), 2010

[19] Yufang Xi and E.M. Yeh. *Distributed algorithms for minimum cost multicast with network coding*.IEEE/ACM Transactions on Networking, 18(2), pp. 379-392, April 2010.

[20] R. Trivisonno, I. Vaishnavi, R. Guerzoni, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani *Virtual links mapping in future sdn-enabled networks. In IEEE SDN for Future Networks and Services (SDN4FNS)* pp. 1-5, 2013.

[21] Ines Houidi, Wajdi Louati, Walid Ben-Ameur and Djamal Zeghlache. *Virtual network provisioning across multiple substrate networks*, Computer Networks, 55(4), pp. 1011-1023, 2011

[22] Guerzoni, R. Trivisonno, R., Vaishnavi, I., Despotovic, Z. *A novel approach to virtual networks embedding for SDN management and orchestration* IEEE Network Operations and Management Symposium (NOMS), 2014