A Bandwidth-Feasibility Algorithm for Reliable Virtual Network Allocation

Rafael L. Gomes, Luiz F. Bittencourt, Edmundo R. M. Madeira
Institute of Computing (IC), University of Campinas (UNICAMP), Brazil
Email: {rafaellgom,bit,edmundo}@ic.unicamp.br

***Abstract*—Currently, the Internet is the main communication medium, however it does not guarantee Quality of Service. To contour this problem, companies establish Service Level Agreements with their Internet Service Providers, including parameters like bandwidth and reliability of the network. This paper presents an algorithm for virtual network allocation in a reliability context, where a feasible bandwidth approach and reliability calculation are used to deploy the virtual network according to the client's requirement. Simulations using a real network topology show the effectiveness of the algorithm to deploy the virtual network in a scenario of multiple cumulative requests.**

I. Introduction

Nowadays, the Internet is the base for many services provided by companies, however the current Internet does not have any guarantee of service level. A popular way to pursue Quality of Service (QoS) between companies/clients and Internet Service Providers (ISPs) is the establishment of a Service Level Agreement (SLA), where companies negotiate and specify parameters to be achieved by the ISP [1].

Different types of traffic flow through the Internet, where each one has different needs (infrastructure requirements), for example real time video transmission, text messages, and others. This occurs because of the increase in popularity of multimedia applications, which are now part of the everyday people's life. Attached to this fact, the users become frustrated when the Internet access fails and the multimedia applications break.

Within this context, one important parameter that can be negotiated is the network reliability. Network reliability refers to the probability of the overall network to provide communication in the event of failure in the network. Network reliability is based on the establishment of link-disjoint path pairs, i.e., a primary path, called the Working Path (WP), and a second one, called the Backup Path (BP). So, when a link/node fails, the WPs affected by the failure switch over the traffic to their respective BP [7].

Currently, the Internet needs to be updated due to the lack of guarantees regarding QoS, which leads to the so-called Future Internet idea. Along with this, the Network Virtualization (NV) approach arises as one of the most prominent technologies for the Future Internet. In a nutshell, NV is a technology that enables the deployment of multiple network environments over the same physical infrastructure [3]. The flexibility of virtualization allows the customization of several virtual services, including the virtual topology and virtual network (VN) resources. In the same context, Software Defined Networks (SDNs) appear as a key approach to be adopted in the Future Internet [2]. SDN is a network design which separates the control and data planes, allowing the programmability of the network. This characteristic changes the control of the network as a whole, increasing the customization capacity of a network environment.

The SDN and NV approaches can be mixed through the network hypervisor (such as Flowvisor [12]), which allows the slicing of the network in layers. Each layer can be configured with particular resources and protocols (deployed in the network controller through apps [2]) in such way, each slice is a customized VN. With that, the ISPs can isolate VNs in the SDN and deploy the functionalities that the client requests.

The ISPs must have an allocation algorithm to allow the deployment of VNs. The allocation algorithm decides which components (links and nodes) will be part of the VN according to some parameters such as: nodes to be interconnected, desired bandwidth (Bw), available Bw in the links, the reliability of the network.

In this context, this paper proposes an algorithm for virtual network allocation that focuses on network reliability negotiation, trying to maximize the number of requests solved, called *Bandwidth-Feasibility* allocation algorithm. We want to adapt the virtual network allocation according to the reliability defined by the client, as well as to minimize the total bandwidth compromised to solve these requests.

This paper is organized as follows. Section II presents some basic concepts. Section III details related work, while Section IV shows the steps to calculate the network reliability. Section V describes the proposed algorithm. Section VI describes some results of the developed algorithm, and Section VII concludes the paper and presents future work.

II. Context

In the context of network virtualization, the capability to customize the parameters of the networks and services provided by the ISP for the clients arises. Many papers [4], [10], [15] use this flexibility of virtualization to bring to the client the desired service, adapting its quality according to the client's definition. Consequently, the client pays a price relative to the desired quality of the service.

Quality can be expressed through diverse parameters, among these parameters the network reliability and the bandwidth are the focus of this work. Reliability represents a factor that measures the capacity of the network to be operational when a component fails. Bandwidth is the amount of data per time unit that can be carried by a link in the network.

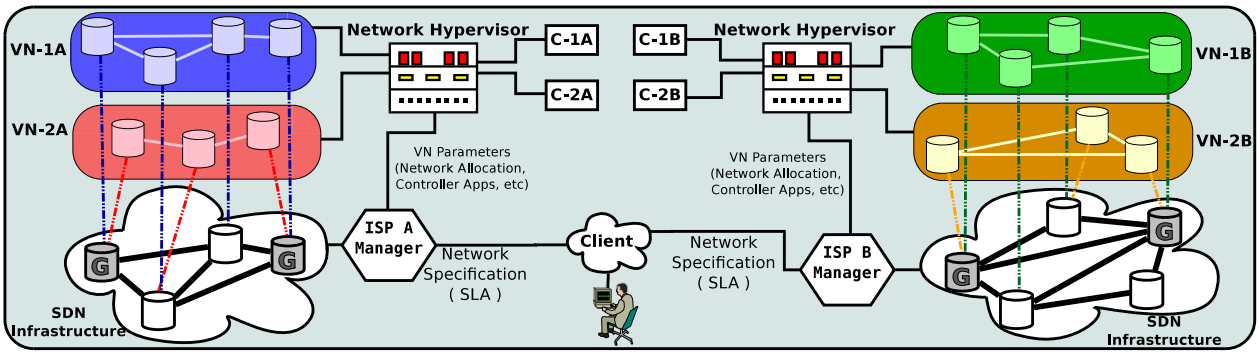

Fig. 1. Scenario representing the context.

Currently, the Internet traffic is composed of many multimedia applications, which need a high bandwidth and are sensitive to packet losses. So, these two parameters (reliability and bandwidth) are related to the QoS experienced by the users. On the other hand, ISPs aim to maximize profit, deploying as much SLAs as possible according to its available infrastructure.

The reliability is directly related to the number of alternative paths of the network, where the higher redundancy of paths, the greater the reliability. However, it is not advantageous for the ISPs to allocate a full redundancy network for a client that does not desire such reliability (and will not pay for it), since it represents waste of resources. The reliability can be achieved through a partial disjoint path approach, so we restrict the redundancy percentage of the network according to the reliability desired by the client: high redundancy for high reliability, and low redundancy for low reliability.

When a client wants a virtual network from an ISP, usually he/she wants to pass through the ISP to reach another domain or specific nodes inside the domain. Thus, the ISP needs to allocate the virtual network between the client's gateway node and the desired destination nodes, which could be the border gateways or the final destination.

Figure 1 illustrates a scenario of the context of this work. The client and the ISP negotiate an SLA (Network Specification), and the VN defined must be deployed by the ISP. To allocate the VN from the existing SDN infrastructure the ISP needs to define the topology, which must have the parameters specified by the client. In this work, the client specifies the desired reliability (detailed in Section IV), the desired bandwidth and the gateways to be interconnected (gray nodes marked with a "G" in Figure 1).

Due to the SDN and VN mixed approach, the allocation of nodes and links are made directly from the infrastructure, i.e., if a component is allocated in two VNs it is directly accessed by the controllers through the network hypervisor [12], where the behavior of the VN is defined through the controller configuration (one controller for each VN). Therefore, the allocation algorithm must define which components of the existing network are linked with the VN deployed at the moment.

For example, if the client negotiates a high reliability VN with *ISP B*, it could allocate the green VN-1B which has two full disjoint paths to link the gateways and C1-B as controller. On the other hand, if the client wants a lower reliability VN, the *ISP A* could deploy the red VN which has just one path between the gateways and C2-A as controller. So, the algorithm proposed in this paper acts in the ISP Manager to allocate the VN according to the client's specification.

Within this context, this paper proposes an algorithm to generate a virtual network topology according to the reliability requested by the client. The algorithm defines the topology according to redundancy restrictions by applying a variation of the method for finding shortest pairs of disjoint paths proposed by Suurballe et al. [14]. The algorithm generates a virtual network topology that encompasses all the desired nodes in the network with the lowest possible Bw, and calculates the network reliability of this topology. The calculation of network reliability is based on the method proposed by Li et al. [8], with few modifications to adjust it to the VN context.

In this paper we evaluate the existing algorithms *Lowest Weighted Path* and *Maximum Available Bandwidth* that will be described in sections V-B1 and V-B2, respectively. The proposed algorithm will be defined in Section V-C.

III. Related Work

In this section we describe some work related to disjoint paths and the virtualization approach in the reliability context.

Gomes et al. [6] aimed to find the most reliable pair of link disjoint paths. To perform this task, the authors used an algorithm to sequentially obtain the k-shortest length constrained paths (with a defined maximum number of arcs per path). The proposal was evaluated through experiments for randomly generated networks with low connectivity to adapt the experiments to the WDM optical networks context.

Lee et al. [7] studied the path protection issue in a network with multiple, possibly correlated, failures. To address this problem, the authors developed an algorithm for finding diverse routes with minimum joint failure probability.

Rahman et al. [11] formulated the Survivable Virtual Network Embedding (SVNE) problem to incorporate substrate failures in the virtual network embedding problem. In general, the authors remove the assumption that the network is operational all the time and develop a hybrid policy heuristic based on a fast re-routing and a pre-reserved quota.

Sun et al. [13] designed a framework for solving the Survivable Virtual Network Mapping (SVNM) problem with resource constraints in a region failure context. The framework addresses two key issues of the SVNM problem: reduce VN mapping cost and minimize the computational complexity.

In our previous work [5], we developed an algorithm to generate a virtual topology according to the desired reliability for the virtual network, however the algorithm does not focus on generating the VN using the lowest possible bandwidth.

None of the papers found in the literature focuses on the development of an algorithm to enable the negotiation of the virtual network reliability, allowing the provider to save resources to maximize the number of requests solved, which is the proposal of this work.

IV. The Network Reliability Calculation

In the context of virtual network allocation, Algorithm 1 initially determines the VN topology, and after that the reliability of the network can be calculated. So, a network reliability method can be applied to evaluate the VN considering that each component (nodes and links) has a reliability value (i.e., a probability of the component to be operational).

In this work, the applied method is based on the reference [8] to model the network states and the most probable states of the network. It considers the reliability of the network as the probability of the network to be operational when L failures occurs. So, the method allows the number of failures customization, i.e., calculates the reliability of the network considering up to L failures. We modified the method in order to the final reliability to include only the communication between the client's node and all destination nodes.

Analyzing the issues related to reliability calculation, in our previous work [5] we identified some tradeoffs between the network reliability and the network size. Initially, we realized that the more paths exist between the root node and the destinations, the greater the network reliability. It occurs due to alternative paths between the source and the destinations. However, the reliability of the path decreases according to the path length increase, since the more components, the greater the cumulative probability of some component of the path to fail. In the same way, our previous study [5] realized another tradeoff that includes the virtual network size and the time to calculate the network reliability, where the time to calculate the reliability grows exponentially according to network size increases (number of components). This encompasses the most probable states definition and the reliability bounds calculation.

V. Proposal

This section describes the proposed algorithm that adapts the virtual network according to the desired reliability, and also minimizes the bandwidth used to solve clients' requests. Moreover, it presents two existing algorithms to define paths in a graph.

The following graph notation is used throughout the paper to model the VN allocation problem. Let $G = (V, E)$ be a weighted directed graph representing the topology of the ISP, where V is a set of n vertices (or nodes) and E is a set of m directed edges (or links). Let D be a set of k designated destination nodes, and $w_{u,v}$ be the cost/weight of link $e_{u,v} \in E$. Let $\wp$ represent a quantity close to infinity, and let ε represent a quantity close to zero. The weight of each link (u, v) is defined according to the algorithm applied in the problem.

A. Disjoint Virtual Network Topology Algorithm

To generate the alternative paths in the virtual network we developed the Algorithm 1 [5], which generates the redundancy inside the virtual network according to desired factor. Therefore, it is the basis for disjoint paths generation using a path definition algorithm, which finds a tree that connects a source node and a set of destination nodes.

Initially, we find the tree T_1 with node s as root by running a Path Definition algorithm ($PathDefinition(Graph,\ Node)$ function). This function calls one of the following algorithms: the proposed *Bandwidth-Feasibility*, the *Lowest Weighted Path* and the *Maximum Available Bandwidth* algorithms. So, the steps performed by this function will change according to the algorithms, described in Sections V-C, V-B1 and V-B2.

A common step in the three algorithms is to count how many times each link is utilized in the redundant paths. The T_1 tree contains, for every node u, a path from s to each node in D. In Algorithm 1, line 2 makes P_1 to contain the links belonging to the paths from s to nodes D in T_1, which are extracted by the $Edges(Graph,\ Set\ of\ Nodes)$ function. Line 3 assigns to e the number of links to be updated to generate the redundancy in the topology. e is calculated as a percentage from the number of links in P_1 according to the desired redundancy, called p ($0 \leqslant p \leqslant 1$).

After running the Path Definition algorithm, Algorithm 1 updates the cost of each link in the graph, creating a new graph $G' = (V', E')$. The algorithm orders the links according to their usage to reach the nodes in D, i.e., the number of times each link is part of all found paths ($OrderedSet(Set\ of\ Edges)$, line 4).

Algorithm 1 Generate a Disjoint Virtual Network Topology

```
1:  Tree T1 = PathDefinition(G, s);
2:  Path P1 = Edges(T1, D);
3:  int e = p * |P1|;
4:  Path P1 = OrderedSet(P1);
5:  for all edge i ∈ G do
6:      for all edge j ∈ P1 do
7:          UpdateLink(j, i)
8:      end for
9:  end for
10: Tree T2 = PathDefinition(G', s);
11: Path P2 = Edges(T2, D);
12: Graph Gf = MergePaths(P1, P2)
```

To create G' the algorithm replaces the cost $w_{u,v}$ of the first e links of P_1 in the function $UpdateLink(Edge)$. This process is performed by lines 5 to 9 of Algorithm 1. As in the $PathDefinition$ function, the $UpdateLink$ function has different steps according to the Path Definition algorithm used, so the steps performed by each one will be described later.

In the next step, the algorithm finds the tree T_2 in the graph with the updated cost of the links by running the same Path Definition algorithm applied before, and it assigns to P_2 the set of links belonging to the paths from s to nodes D in T_2 (lines 10 and 11, respectively).

Finishing that process, the algorithm merges the paths of P_1 and P_2 (line 12), i.e., the algorithm creates a graph adding the links and nodes that exist in both sets. This process is performed by the $Merge(Set\ of\ Edges, Set\ of\ Edges)$ function. The resulting graph is the final topology G_f that contains relative disjoint paths between the node s and the nodes of D.

Basically, the algorithm constructs an initial tree in the first lines, and after that, with the update of link cost, it seeks alternative paths by adding new edges to the initial tree. This addition of links is limited by the redundancy factor defined.

The link update ($UpdateLink$ function in line 7) is used to avoid the usage of edges already being used, but without discarding them as an option, thus forcing the algorithm to seek for alternative paths to reach the desired nodes in the second call of path definition algorithm (line 10). If no alternative path exists, the algorithm uses the path already known due to non existence of a disjoint path. Therefore, there may be cases where the desired redundancy can not be met due to the lack of alternative paths, where different redundancy factors can generate similar topologies.

By running the developed algorithm of relative disjoint paths, a virtual topology in accordance with the redundancy desired by the client is constructed, taking advantage of the flexibility offered by the network virtualization. This flexibility enables the bandwidth usage reduction (since it can use part of the edges in the network) or/and the provision of reliability (since it enhances the routing possibilities) according to the client requirement.

B. Existing Algorithms

In this section we present two existing algorithms to define paths in a graph, which can be applied in the context of virtual network allocation. This section describes the set of steps performed by the *Lowest Weighted Path* and *Maximum Available Bandwidth*.

1) Lowest Weighted Path: In this section we describe our previous algorithm, called *Lowest Weighted Path*, proposed in [5]. The *Lowest Weighted Path* algorithm aims to find the shortest path in the topology which satisfies the desired bandwidth. So, it punishes the links which have lower bandwidth than the value requested, and considers suitable all the links which have at least the bandwidth requested.

To apply this idea, the path definition in this algorithm first defines the weight of each link (u, v) as in Equation (1), where $R_{(u,v)}$ is the current available resource capacity of the link and R_r is the amount of bandwidth requested by the client. For example, if the client requests 10 Mbps and the link has only 8 Mbps available, its weight will be 3.

$$w_{u,v} = \begin{cases} 1 + 10\left(1 - \frac{R_{(u,v)}}{R_r}\right) & , if\ R_{(u,v)} < R_r \\ 1 & , otherwise \end{cases} \quad (1)$$

The path definition in the *Lowest Weighted Path* algorithm proceeds by finding the shortest path between the source node (s) and each destination node (D). To do this task, the algorithm applies the classic Dijkstra's algorithm, using the weights according to Equation (1). It finds the shortest path to each destination, and then it merges the paths.

Following the steps described in Algorithm 1, the P_1, created from the paths defined by Dijkstra's algorithm, contains the links belonging to the paths from s to nodes D. Updating links of *Lowest Weighted Path* occurs according to Algorithm 2, where it aims to avoid the usage of an already used link in the alternative path search, as cited in Section V-A. In the Algorithm 2, j is the link passed as parameter for the function, as shown in Algorithm 1.

Algorithm 2 $UpdateLink(Edge\quad j, Edge\quad i)$ of *Lowest Weighted Path*

```
1: node u = j.from, v = j.to;
2: if (i == j) then
3:     w'_{u,v} = 0;
4:     if (e > 0) then
5:         w'_{u,v} = ℘ ;
6:         e = e − 1 ;
7:     end if
8: else
9:     w'_{u,v} = w_{u,v} − d(s,v) + d(s,u);
10: end if
```

Algorithm 2 replaces the cost $w_{u,v}$ of the first e links of P_1 by $w'_{u,v} = \wp$, and the remaining links by $w'_{u,v} = 0$, where e is the number of links to update according to Algorithm 1. The other links are updated with $w'_{u,v} = w_{u,v} - d(s,v) + d(s,u)$, as presented by Suurballe et al. [14].

To illustrate the behavior of the *Lowest Weighted Path* algorithm, Figure 2 presents a scenario where a network connecting nodes 0, 2 and 4 with 20 Mbps is requested. The values in black next to the links represent the available bandwidth, and the gray circles are the respective weights according to Equation (1). The reliability of each component (link or node) is 0.99. The definition of the reliability factor of a network is expressed between 0 and 1, where the higher this value, the higher the reliability of the network. The calculation of reliability factor described in Section IV.

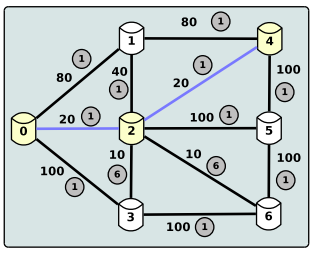

Fig. 2. Example of Lowest Weighted Path Algorithm.

The blue links in Figure 2 represent the links allocated to the desired VN. The *Lowest Weighted Path* algorithm defines a network which gets a reliability factor of 0.95, using 40 Mbps of the available bandwidth in the network. However, the algorithm allocates all the available bandwidth of the two links, preventing future requests with lower bandwidth to use them and reducing the number of access links to the nodes.

2) Maximum Available Bandwidth: The *Maximum Available Bandwidth* is an algorithm based on the widest path algorithm for the context of bandwidth allocation [9]. The *Maximum Available Bandwidth* aims to find the path with highest available bandwidth regardless the number of hops/links used. So, it is not a cumulative algorithm, it just saves the path whose bottleneck (link with lowest bandwidth) has the highest available bandwidth.

The *Maximum Available Bandwidth* does not apply weights to find the path, it uses the real amount of available bandwidth as criterion in the search process. The *Maximum Available Bandwidth* updates the links according to Algorithm 3. The variables of Algorithm 3 follow the same model presented in Algorithm 2.

Algorithm 3 $UpdateLink(Edge\ \ j, Edge\ \ i)$ of *Maximum Available Bandwidth*

```
1: if (i == j) then
2:     node u = j.from, v = j.to;
3:     w'_{u,v} = ℘;
4:     if (e > 0) then
5:         w'_{u,v} = ε ;
6:         e = e − 1 ;
7:     end if
8: end if
```

The Algorithm 3 sets a link weight to ε if it was already used (line 1), and to $\wp$ (represents value close to infinity) otherwise. It encompasses the goal of the $UpdateLink$ function (as described in Section V-A) for the context of *Maximum Available Bandwidth*, which is to force the algorithm to search alternative paths inside the network.

To exemplify the behavior of the *Maximum Available Bandwidth* algorithm, we applied the same example scenario which results in the allocation are presented in Figure 3. The orange links represent the links allocated to deploy the virtual network among the desired nodes (the yellow nodes) without redundancy. Unlike the *Lowest Weighted Path*, the *Maximum Available Bandwidth* aims to find the paths which have the higher lowest available bandwidth, without caring about the number of hops in this path. Due to this fact, the network composed only of the orange links gets only 0.91 of reliability, since it allocates too many components (which can fail) without any redundancy. Details about reliability issues will be presented in Section IV.

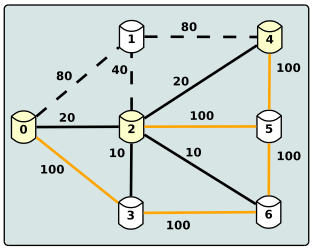

Fig. 3. Example of Maximum Available Bandwidth Algorithm.

If we apply a full redundancy approach in our example scenario, the algorithm will allocate the orange links plus the dashed links, getting a reliability factor of 0.95, as now it has an alternative path for each destination.

The *Maximum Available Bandwidth* algorithm spends more bandwidth than necessary to allocate a network with 0.95 reliability. It tends to saturate the bandwidth of the network more quickly, as it may not be able to solve requests with high reliability between many nodes. For example, in this scenario the algorithm will allocate nine links, while *Lowest Weighted Path* uses just two links (as shown in Figure 2).

C. Bandwidth-Feasibility

The network reliability and its bandwidth are important parameters of a network, which in the context of VNs and SDNs can be negotiated between client and ISP through an SLA, as shown in Figure 1. These parameters are directly related to the charging to deploy the network and the QoS experienced by the applications that will pass through the network, since it represents the resource usage of the ISP.

Currently, there exists a diversity of applications that need internet access, where each of these applications has distinct QoS specifications. For example, if a client needs a virtual network just to use services like email, simple web pages access, instant messenger and others, then this virtual network can have low bandwidth and reliability requirements, since these types of traffic exchange small amounts of data and are not time sensitive to possible fails in the network. So, the allocated virtual network needs to be simple, solving the client's requirement using as low resources of the ISP as possible.

On the other hand, if a client needs a virtual network to run multimedia/real time applications as online games, video/voice calls, healthcare and others, then this virtual network will need higher resources allocation and reliability, as long as any break in the communication, either a network component fail or network congestion, results in major problems related to the QoS experienced by the user.

Therefore, in this context we have both perspectives: from the client and from the ISP. The client wants a virtual network which meets his requirements, paying a proportional price for it. In the same way, the ISP aims to negotiate with as much clients as its infrastructure can bear, since the more clients the ISP has, the higher its profits can be.

The proposed algorithm for virtual network allocation, named (*Bandwidth-Feasibility*), defines a virtual network with the bandwidth and reliability specified by the client directly in the SLA. It focuses on defining a network between the desired nodes with the lowest saturation of the ISP network.

The algorithm should allocate the shortest path with higher available bandwidth. With that it saves bandwidth and adds a lower number of components prone to fail (links and nodes) to the network, directly influencing the reliability of the virtual network (as will be described in Section IV).

In the *Bandwidth-Feasibility* algorithm, the weights used for the links follow Equation (2), where $R_{(u,v)}$ is the available bandwidth of the link and R_r is the bandwidth requested by the client.

$$w_{u,v} = 1 + 10\left(\frac{R_r}{R_{(u,v)}}\right) \tag{2}$$

Equation (2) aims to enhance the usage of links with higher available bandwidth, where links with available bandwidth lower than requested are removed from the algorithm. For example, if the client requests 10 Mbps and the link has 20 Mbps available, its weight will be 6. In the same way, if the client wants 10 Mbps and the link has 100 Mbps available, the weight of the link will be 3. So, the algorithm will choose to use two links with higher available bandwidth instead of just one that has almost no bandwidth left. This strategy represents a possible adjustment in the reserved bandwidth for the VN in cases of resources saturation, i.e., it allows the VN adjustment on the fly without the redefinition of the VN components.

To develop the *Bandwidth-Feasibility* algorithm, we evolved our previous work presented in [5]. To perform this evolution we applied the principles of multiple-attribute search presented in [9]. The overview of the *Bandwidth-Feasibility* algorithm is presented in Algorithm 4.

In Algorithm 4, s is the source node, $b_{i,j}$ represents the available bandwidth between nodes i and j, b is the required bandwidth, and W is a list of best paths to each node from node s, where W_i is the best path between source node and node i. The function $lower$ returns the node whose path presents the lowest value in the list passed as parameter.

Algorithm 4 Bandwidth-Feasibility Path Definition Algorithm

```
1:  S = i;
2:  S' = AllNodes − {i};
3:  for all Node j ∈ S' do
4:      if (w_{s,j} < ∞) then
5:          if (b_{s,j} ≥ b) then
6:              W_j = w_{s,j} ;
7:          else
8:              W_j = ℘;
9:          end if
10:     else
11:         W_j = ∞;
12:         w_{s,j} = ∞ ;
13:     end if
14: end for
15: while S' ≠ ∅ do
16:     Node Min = lower( W );
17:     if (b_{s,Min} < b) then
18:         Finish the algorithm returning fail;
19:     end if
20:     S = S + Min;
21:     S' = S' − {Min};
22:     for all Node j ∈ S' do
23:         if (w_{Min,j} > ∞) then
24:             if (W(j) > W(Min) + w_{Min,j}) then
25:                 if (b_{Min,j} > b) then
26:                     W(j) = W(Min) + w_{Min,j};
27:                 end if
28:             end if
29:         end if
30:     end for
31: end while
```

Lines 1 and 2 initialize the sets of nodes already tested and not tested yet, respectively S and S'. Lines 2 to 14 assign to each link its weight according to Equation (2), if it has at least the requested bandwidth, or assigns ∞ otherwise. From line 15 to 31, the algorithm will travel through the existing nodes verifying if the node with lowest weight path to source node (node Min defined in line 16) can be used in the path to reach other nodes with lower cost (lines 20 to 30).

The *Bandwidth-Feasibility* algorithm aims to find paths with requested bandwidth, so in lines 17 to 19 the algorithm verifies if the node with the lowest weight path (best option at the moment) has the desired bandwidth, and if it does not, then the algorithm ends, since a feasible path will not be found. Using this approach the algorithm saves time to analyze the next existing requests.

Following the steps proposed in Algorithm 1 (described in Section V-A), after defining the initial paths we need to update the weights of the links. Updating links of *Bandwidth-Feasibility* follows the steps presented in Algorithm 5. As in the *Lowest Weighted Path*, it aims to avoid the usage of already used links in the alternative path search.

Algorithm 5 $UpdateLink(Edge\ \ j, Edge\ \ i)$ of *Bandwidth-Feasibility*

```
1: if (i == j) then
2:     node u = j.from, v = j.to;
3:     w'_{u,v} = 0;
4:     if (e > 0) then
5:         w'_{u,v} = ℘ ;
6:         e = e − 1 ;
7:     end if
8: end if
```

The *Bandwidth-Feasibility* algorithm does not apply the update in the reverse link (line 11 in the Algorithm 2), as suggested by Suurballe et al. [14]. This step is unnecessary for the context of this work, since when a link is allocated, it is used in both directions (bidirectional communication).

To illustrate the behavior of the *Bandwidth-Feasibility* algorithm, we applied it in the same scenario used before. The values in black represent the available bandwidth of a link and the gray circles are the weights defined by Equation (2).

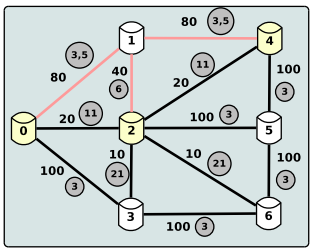

Fig. 4. Example of Bandwidth-Feasibility.

The red links in Figure 4 represent the links allocated to deploy the virtual network between the desired nodes (the yellow nodes) without redundancy. The defined virtual network gets a reliability of 0.94.

Differently from the previously analyzed algorithms, *Bandwidth-Feasibility* allocated the shortest path with highest available bandwidth, i.e., the links $(0, 1)$, $(1, 2)$ and $(1, 3)$. Although, allocating links with less bandwidth than *Maximum*

Available Bandwidth, *Bandwidth-Feasibility* yet uses fewer links (reduces aggregated bandwidth consumption) and gets a higher reliability without redundancy.

When compared to *Lowest Weighted Path*, the *Bandwidth-Feasibility* uses more links, and consequently it gets a little lower reliability factor: 0.94 against 0.95 of *Lowest Weighted Path*. Nevertheless, the *Bandwidth-Feasibility* has two advantages in front of *Lowest Weighted Path*. First, the *Bandwidth-Feasibility* does not exhaust the available bandwidth of used links, preventing further requests with lower bandwidth requirements to be solved. Second, if during the search the *Bandwidth-Feasibility* realizes that the available bandwidth will not be adequate, it aborts the search and considers the request unsolved, saving time mainly due to reliability calculation described in the next section.

VI. Experiments

To evaluate the proposal we applied the algorithm in a real network topology. The experiments compare the performance of the algorithms described in this paper: *Lowest Weighted Path* (Section V-B1), *Maximum Available Bandwidth* (Section V-B2) and *Bandwidth-Feasibility* (Section V-C).

The experiments aim to evaluate the capacity of the proposal to solve a set of virtual topology requests. We work in the context of deploying a virtual network inside an ISP, so we want to generate a topology that connects the border gateway of the client (the source node) to specific border nodes in the ISP (set of destination nodes), specifying the bandwidth (Bw) and the reliability ($Comm$) for the virtual network.

The experiments use the GEANT Network Topology[1], shown in Figure 5. GEANT is the European research backbone that interconnects National Research and Education Networks (NRENs) across Europe and provides worldwide connectivity through links with other regional networks.

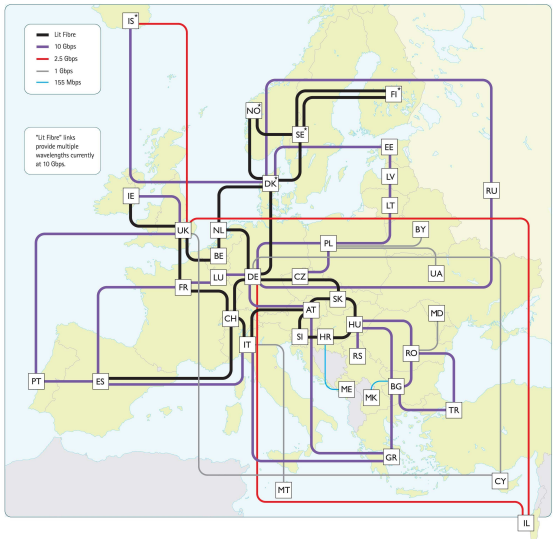

Fig. 5. Topology of the GEANT network.

Each set of requests is generated randomly and is composed of 150 requests. The parameters of a request are: (i) the source node; (ii) the set of destination nodes; (iii) the requested bandwidth (values between 10 and 200 Mbps); and, (iv) the desired reliability (values between 0.85 and 0.95).

In the experiments, the reliability of each component in the network is 99%. Likewise, the results are presented with a 95% confidence interval. The experiments were repeated one hundred times, i.e., one hundred sets of 150 requests are randomly created and the same sets are used as input for each algorithm.

To evaluate the capacity of the algorithms to allocate the virtual networks when the resources of the network became scarce, the requests are cumulative, i.e., the resources and components allocated for a virtual network are not released until all the requests were analyzed. Initially, every link in the network has 1Gbps of available Bw.

We evaluated four aspects: (i) the requests solved over the number of analyzed requests; (ii) the number of links saturated (available bandwidth lower than 10% of the original value); and, (iii) the average available bandwidth in the network.

The status of the requests defines if a request was solved or not by an algorithm. The cases where the request is unsolved, the reason is specified: (i) A request that the algorithm did not find a topology with the specified reliability, called *reliability fail*; and, (ii) the request encompasses nodes where the desired bandwidth can not be allocated, called *bandwidth fail*.

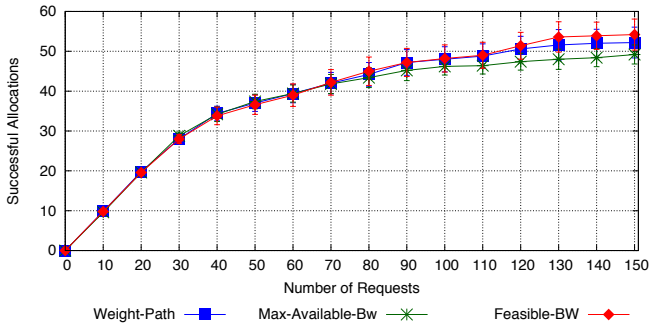

Fig. 6. Successful allocations.

The results regarding the status of the requests are shown in Figure 6. It shows the cumulative number of solved requests. Among the analyzed algorithms, the *Bandwidth-Feasibility* algorithm solved more requests than *Lowest Weighted Path* and *Maximum Available Bandwidth* algorithm, where the mean values for each one were 54.51, 52.06 and 48.69, respectively.

Based on information of Figure 6, we observe that the behavior of the algorithms, mainly *Lowest Weighted Path* and *Bandwidth-Feasibility*, are close until the 110th request. At this point, the resources of the network are limited, due to allocation of the previous requests. This enhances the advantage of the proposed *Bandwidth-Feasibility*, where it can solve more requests in cases of limited resources in the network.

To enhance the difference between the algorithms, the Figures 7 and 8 illustrate the bandwidth information according to the number of requests analyzed. Figure 7 presents the average available bandwidth left in the network after the allocation of requests (i.e., the average bandwidth of all links) and Figure 8 shows the number of links saturated.

Following the same behavior illustrated before, the Figure 7 presents the available bandwidth left after the allocation. It can

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

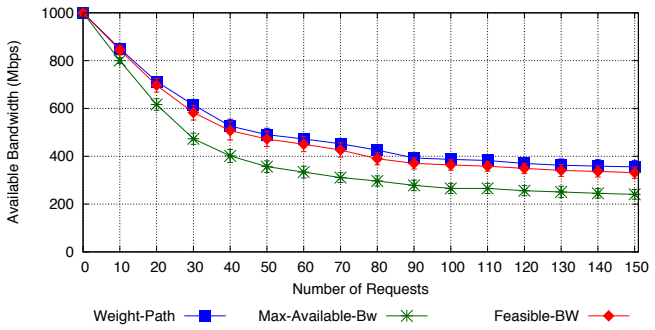

Fig. 7. Available bandwidth after allocation.

be seen that *Maximum Available Bandwidth* algorithm spends more bandwidth when compared with the other two algorithms. This occurs due to the maximization search principle of the algorithm, as described in Section V-B2.

Among the algorithms, the *Lowest Weighted Path* is the one that consumed less bandwidth, where the *Bandwidth-Feasibility* algorithm reached a close value. This happens due to the nature of the *Lowest Weighted Path* to search for shortest paths (lower number of hops), disregarding about the effect of the allocations of these paths on the network as a whole. This effect is observed in Figure 8.

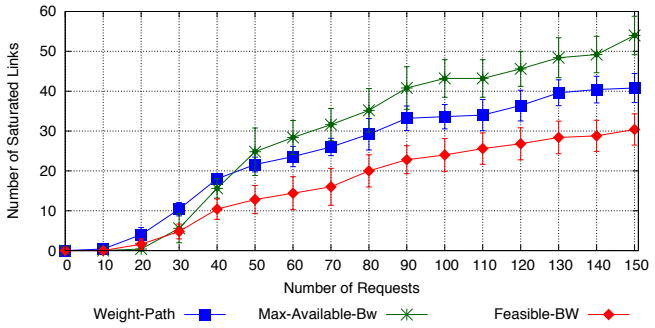

Fig. 8. Number of Saturated Links.

Figure 8 illustrates the saturation behavior of the *Lowest Weighted Path*. It allocates the links until they are almost saturated, preventing the defined VN to increase the allocated bandwidth if necessary. On the other hand, the *Bandwidth-Feasibility* algorithm disperses the allocation of the links when it identifies a high usage of the links, this behavior allows the adaptation of the bandwidth allocated for the VN in cases of increase on traffic demand of the VNs.

The *Maximum Available Bandwidth* tends to defines VN with higher number of components, since it searches for paths with high available bandwidth, regardless of the total bandwidth consumption. This fact can be observed in the information presented in Figures 7 and 8.

The experiments suggest that the proposed algorithm can meet the topology definition according to the reliability specifications set by the client, when feasible. The *Bandwidth-Feasibility* algorithm solved more requests, getting a performance at least 4% higher than the existing algorithms. Moreover, to solve the requests, the proposed algorithm uses less Bw as well as links with higher available Bw, resulting in lower number of saturated links in the network.

VII. Conclusion

This paper presents an algorithm for VN allocation in an SDN and VN mixed context, where the proposed algorithm defines the virtual topology that is better for both the client and the ISP.

The experiments used a real topology, and showed the capacity of the proposal to generate the virtual topology according to the client's desire. The proposed *Bandwidth-Feasibility* algorithm outperforms existing algorithms, serving more customers and enabling the maximization of the provider profits.

As future work, we intend to extend the algorithm for a multicriteria topology definition and add energy issues.

Acknowledgment

The authors would like to thank São Paulo Research Foundation (FAPESP - grants 2014/00873-7 and 2012/04945-7), CAPES, RNP and CNPq for the financial support.

References

[1] H. Carvalho, N. Fernandes, O. Duarte, and G. Pujolle, "Slapv: A service level agreement enforcer for virtual networks," in *International Conference on Computing, Networking and Communications (ICNC)*, 2012, pp. 708–712.

[2] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *IEEE Internet Computing*, vol. 17, no. 2, pp. 20–27, 2013.

[3] R. Esteves, L. Granville, and R. Boutaba, "On the management of virtual networks," *IEEE Communications Magazine*, vol. 51, no. 7, 2013.

[4] R. L. Gomes, L. F. Bittencourt, and E. R. M. Madeira, "A generic sla negotiation protocol for virtualized environments," in *Proceedings of 18th IEEE International Conference On Networks (ICON 2012)*, 2012.

[5] R. L. Gomes, L. F. Bittencourt, and E. R. M. Madeira, "A virtual network allocation algorithm for reliability negotiation," in *22st International Conference on Computer Communications and Networks (ICCCN)*, 2013.

[6] T. Gomes and J. Craveirinha, "Efficient calculation of the most reliable pair of link disjoint paths in telecommunication networks," *European Journal of Operational Research*, vol. 181, no. 3, 2007.

[7] H.-W. Lee, E. Modiano, and K. Lee, "Diverse routing in networks with probabilistic failures," *IEEE/ACM Transactions on Networking*, vol. 18, no. 6, pp. 1895 –1907, 2010.

[8] V. Li and J. Silvester, "Performance analysis of networks with unreliable components," *IEEE Transactions on Communications*, vol. 32, no. 10, pp. 1105 – 1110, oct 1984.

[9] D. Medhi and K. Ramasamy, *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann Publishers Inc., 2007.

[10] H. Pouyllau and R. Douville, "End-to-end qos negotiation in network federations," *3rd IEEE/IFIP International Workshop on Bandwidth on Demand and Federation Economics*, 2010.

[11] M. Rahman, I. Aib, and R. Boutaba, "Survivable virtual network embedding," in *NETWORKING*, 2010, vol. 6091, pp. 40 – 52.

[12] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[13] G. Sun, H. Di, H. Yu, L. Li, and V. Anand, "The framework and algorithms for the survivable mapping of virtual network onto a substrate network," *IETE Technical Review*, vol. 28, no. 5.

[14] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, no. 2, pp. 325–336, 1984.

[15] F. Zaheer, J. Xiao, and R. Boutaba, "Multi-provider service negotiation and contracting in network virtualization," *12th IEEE/IFIP Network Operations and Management Symposium*, 2010.