Controllable virtual network service over a multiadministrative multi-domain network

Stanislas Pedebearn, Slim Abdellatif, Pascal Berthou, Dariusz Nogalski, Dallal Belabed

▶ **To cite this version:**

Stanislas Pedebearn, Slim Abdellatif, Pascal Berthou, Dariusz Nogalski, Dallal Belabed. Controllable virtual network service over a multiadministrative multi-domain network. 27th IEEE International Symposium On Real-Time Distributed Computing (ISORC 2024), May 2024, Carthage, Tunis, Tunisia. à paraître. hal-04502334v2

HAL Id: hal-04502334
https://hal.science/hal-04502334v2

Submitted on 15 Apr 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Controllable virtual network service over a multi-administrative multi-domain network

Stanislas Pedebearn
LAAS-CNRS, Universite de Toulouse
CNRS, UPS, Toulouse, France
spedebearn@laas.fr

Slim Abdellatif
LAAS-CNRS, Universite de Toulouse
CNRS, UPS, Toulouse, France
slim@laas.fr

Pascal Berthou
LAAS-CNRS, Universite de Toulouse
CNRS, UPS, Toulouse, France
berthou@laas.fr

Dariusz Nogalski
Military Communication Institute
Zegrze,Poland
dariusz.nogalski@wil.waw.pl

Dallal Belabed
Airbus Defence and Space
Issy-les-Moulineaux, France
dallal.belabed@airbus.com

***Abstract*—In this paper, we propose a controllable virtual network service that can be provided on a multi-administrative multi-domain network, and whose behaviour can be programmed and customized according to user needs. A resource allocation algorithm is proposed to compute the resources, from different domains, that are needed to support the virtual network with the required QoS (Quality of Service) and capabilities. An implementation of the service on an OpenFlow-enabled multi-domain network is described. The service is then applied in the context of coalition military network to show its benefits and potential.**

***Index Terms*—virtual network embedding, SDN, slicing, multi-administrative multi-domain networks, QoS**

I. Introduction

Among the benefits brought by SDN, network softwarization and network virtualization is the ability to enable a variety of services that better meet current and forthcoming user needs. Since the advent of SDN and thanks to the flow-based forwarding promoted by SDN and OpenFlow, many connectivity services with various QoS guarantees, possibly set up on-demand and adjusted dynamically, were investigated and implemented in operational networks (e.g., [1] [2] [3]). In addition, services that go beyond providing connectivity by including network capabilities/functions were also extensively investigated (i.e. service chaining/VNF chaining, e.g., [4]).

Another kind of service that is not intended for a predefined packet flow is providing a virtual network to support all the traffic exchanged between connecting user networks, with some predefined performance. Such virtual network is the interconnection of virtual nodes, hosted on physical network nodes, via virtual links established on top of the network infrastructure. It is not new and was considered for some time now, especially in a single domain context (in datacenter and cloud environements with the concept of Network as A Service (NAAS) [5]). Also, it is a way of implementing network slicing [6]. This work proposes an End-to-End (E2E) multi-domain service, which provides to the user a fully controllable (programmable) virtual network that spans multiple domains. This network is composed of virtual nodes optimally scattered throughout the multi-domain network and connected with virtual links that, in turn, can span multiple domains. A network controller is also part of the service as well as its connecting links to the virtual nodes (see Fig.1). More precisely, the idea is to provide a virtual network, whose behavior can be programmed and adjusted at will. One typical example of the service is an OpenFlow based multi-domain virtual network with its OpenFlow controller and OpenFlow virtual switches. Depending on the network control applications running on the controller, different behaviors or traffic management policies can be programmed on the virtual network. They can even change over time to meet evolving needs. Also, thanks to OpenFlow, virtual nodes support a finer-grained packet forwarding than what conventional network nodes can do (which typically apply destination-based forwarding).

The proposed service may be useful in the context of coalition networks. For instance, in case of federated military networks where ally nations connect and share some of their network infrastructure to build a multi-domain Federated Mission Network (FMN) [7]. We assume that nations share the same mission goals and trust each other. On top of such multi-domain federated network, setting up specific E2E multi-domain virtual networks (with comprehensive control on how traffic should flow via the virtual network) is very beneficial for a nation (or a group a nations) involved in a mission. The nation(s) can precisely control the assigned resources (from multiple nations) to effectively support and adapt to the changing and diverse network traffic exchanged during the military mission. All nations advertise their resources which can be used for the E2E purpose. The requesting nation (i.e., service owner) solicits the advertised resources and then coordinates setup, modification and decommission of the service (life-cycle on the federated

level). This coordination entity asks participating nations to provide the resources (e.g., link with required bandwidth and delay), but it is up to each nation (responsible for the segment) to manage the physical resources of the segment on the national level. From the mission goals perspective, it might be reasonable to split the role of service owner (service lifecycle) from the role of service user that operates the virtual network. The former guarantees the lifecycle efficiency (e.g., time to setup/modify the service) as entity being topologically close to each nation. The latter one is the entity ultimately responsible for command and control of the network, e.g. the operational HQ of the international troops (or one of its subordinate entities, not necessarily the same as service owner). Such entity would be responsible for programming the service usage (steer the set of traffic paths within the virtual network) applying different policies e.g. re-program already embedded paths when priority traffic comes into play (e.g., urgent medical evacuation, important intelligence data from a drone etc.) drop or reschedule low priority traffic.

In this paper, we investigate the advantages of such a service, we propose an ILP formulation for the embedding of our proposed service in a multi-domain context. We also propose a prototype implementation of the service on a heterogeneous platform that combines network emulators and physical SDN/OpenFlow switches and describe its operation by enforcing a basic traffic management policy.

This paper is organized as follows. Section II describes the proposed service and the considered multi-domain substrate network. Section III proceeds with the mathematical formulation of the embedding of the proposed service on the considered substrate network model. Section IV describes the implementation of the service and demonstrates its operation. Section V highlights its benefits and some of its applications and positions our work with respect to the state of the art. Section VI concludes the paper and draws up some perspectives of this work.

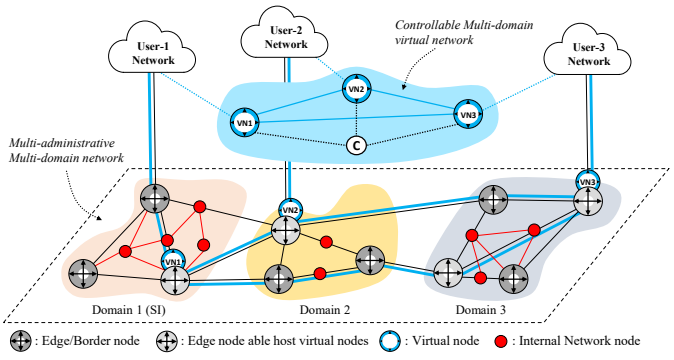

Fig. 1. Illustration of the proposed service.

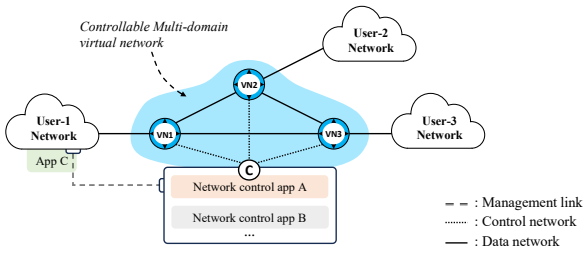

Fig. 2. Service components.

II. Considered service and Muti-domain substrate network

A. Proposed service

The programmable virtual network service that we are proposing is classically composed of three different parts (depicted in Fig.2):

- The data network in charge of conveying user traffic that is exchanged between connecting user hosts. It is composed of a set of virtual nodes that can be hosted on different physical nodes located at different domains, and a set of virtual links connecting in one hand virtual nodes to each other and user hosts to the virtual network. Each virtual link is characterized by a bandwidth and delay requirement. Virtual nodes are characterized with a maximum packet switching rate and a forwarding table size.
- The control network, which is composed of the virtual network controller (which can be hosted on any server/datacentre from any domain) and the virtual links that connect the controller to all virtual nodes. It is in charge of conveying control and monitoring information that enable virtual network programmability. If we consider again the case of an OpenFlow based virtual network, this control network supports all OpenFlow traffic. Virtual control links are characterized with a bandwidth and delay requirements. In turn, the virtual network controller has processing (CPU) resource requirements. The needed resources (control link bandwidth and processing) are derived from the network control applications that state the virtual network behaviour.
- Optionally, a management network/links may also be part of the service in order to allow interactions with the controller from remote locations. In other words, the virtual network controller and some network control applications can be executed on separate servers (in user premisses). Again, some bandwidth and processing resource requirements are typically expressed.

B. Considered multi-domain substrate network

When embedding an end-to-end multi-domain service with some Quality of Service (QoS) guarantees, the initiating domain must solicit multiple domains to support portions of the service. In order to identify the appropriate portions (exhibiting the appropriate performance guarantees), combine

them to compose the requested service, the initiating domain relies on the aggregated topologies exposed by other domains (i.e., compact portray of the network topology with the available resources of each domain). In a multi-administrative multi-domain context, domains restrict the topology information that they disclose to other domains. For instance, when dealing with multi-domain connectivity services with QoS on a multi-administrative multi-domain network, the aggregated topology that domains expose to the others is usually limited to border nodes (Edge nodes connected to other domains) fully or partially connected to each other with abstract links. Some QoS/performance information are attached to these latter, typically maximum and available capacity, maximum transfer delay, maximum packet loss rate, etc. Some cost information can also be used to enforce some policy-based decisions.

To embed the proposed service, domains need to expose more than border nodes and abstract links. Indeed, they need to reveal some nodes that are able to host virtual nodes, the type of virtual nodes that they can host and the available capacity, which is used to assess whether sufficient resources are available to support new virtual nodes instances. Also, some compute nodes (with the available computing resources) need to be disclosed in order to be considered as a potential host for the virtual network controller and eventually some network control applications.

As network slicing is gaining momentum, some domains may also support multiple domain-level slices (each providing predefined types of domain-level services) that can be exposed and made available to other domains in order to compose their end-to-end multi-domain services. In such cases, slices can also to be disclosed as part of the aggregated topology of a domain with the characterization of the type of services provided by the slice and the available capacity. In this work, we focus on transport slices, i.e., slices providing connectivity with some predefined QoS between exposed nodes. One example of such a slice is a Low-latency slice, which interconnects a set of nodes exposed by the domain with a transfer delay of a couple of tenths of milliseconds. Another example is a slice providing efficient point-to-multipoint transmission services between a set of nodes exposed by the domain.

Lastly, in this work, domains can also expose non-border nodes, which allow them to abstract/compact some of their infrastructure topology constructs. As shown in [8], the inclusion of these abstracted non-edge nodes allows more precise multi-domain abstractions, leading to significant improvement of service demand admissibility and decent decrease of service demand admission delays, and network and computing overhead.

Fig.3 sums up the different constructs that a domain can use to build its aggregated topology that it wants to disclose to other domains. Obviously, domains have sovereignty on which

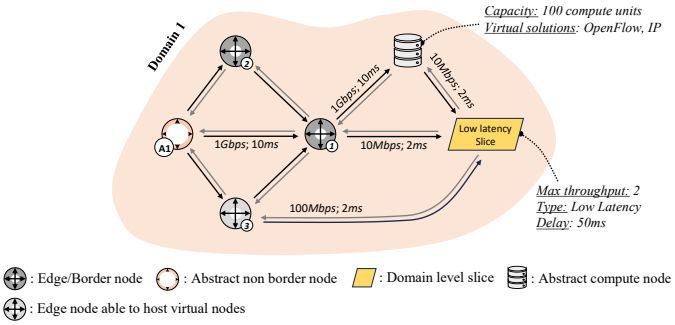

Fig. 3. Illustration of topology abstraction components.

constructs they want to use and on the aggregated topology they want to expose, and also, on how they map the resources attached to the exposed topology to the physical resources at the domain level. They even have sovereignty on the multi-domain substrate network that they derive from the aggregated topologies exposed by domains (by applying some filtering on advertised topologies), which is used for embedding the virtual network service demands.

III. Mathematical formulation

This section describes the mathematical formulation that we propose to solve the online resource allocation problem of the controllable multi-domain virtual networks on top of the substrate multi-domain network described in the previous section. The output of the allocation is the assignment of the physical nodes that will host each virtual node and each controller, as well as the set of data paths that will support each node-to-node logical link and each controller-to-node logical link. We assume that path-splitting is not allowed to support all above-cited logical links. Without any loss of generality, we assume that one single controller is in use, and virtual nodes cannot be hosted on the same physical node. Also, the objective of the resource allocation is to distribute fairly network traffic and use efficiently network resources exposed by all domains. The proposed resource allocation method can be easily adjusted and extended to consider different options and integrate additional constraints regarding the placement of nodes and the controller(s).

Below, the substrate network and virtual network demand models are described. Then, the variables and problem constraints are listed. Lastly, the considered objective function is defined.

A. Multi-domain substrate Network model

The multi-domain substrate model is the interconnection of the aggregated topologies exposed by all domains. It is modeled as a unidirectional graph $G = (N, A)$ where N is the set of vertices, which correspond to exposed nodes. $A \subseteq N \times N$ is the set of edges (i.e. the set of exposed abstract links) connecting exposed nodes to each other. $N = N_t \cup N_c \cup N_s$, where N_t, N_c, N_s are the set of transport nodes (border network nodes and also abstracted non-border

nodes), compute nodes and slice nodes. Indeed, domain level slices are also modeled as nodes; they are connected to the nodes that either send to or receive packets from the slice. The performance of the service (delay, etc.) provided by the slice is reported on the corresponding input edges.

To each node $i \in N_t$, is associated available switching capacity TE_i, which is the current available number of entries of its forwarding table. The maximum size of node i flow table is denoted by $\text{TE}_i^{\max}$. In addition, node i is characterized by a maximum forwarding rate $\text{FR}_i^{\max}$, which in fact reflects the amount of node i switching capacity (resources) that a domain reserves to other domains (expressed in terms of the maximum overall (whatever the incoming interface) number of packets that the node i forwards, on behalf of the federation, per unit of time). The remaining switching capacity at time of arrival of a new demand is FR_i. Similarly, for each slice $i \in N_s$, we respectively denote as $\text{FR}_i^{\max}$ and FR_i the maximum and remaining capacity (in terms of packet rate) that the slice can handle. Last, for each link $(i,j) \in A$, we respectively denote by $\gamma_{ij}^{\max}$, γ_{ij}, and l_{ij} as the maximum capacity of the link as initially made available by the corresponding domain to the other domains (i.e. before any resource assignment), the available capacity of (i,j) at time of service demand arrival, and the max packet transfer delay along the link.

We denote as $N_v \subseteq N_t \cup N_c$ the set of substrate nodes that are able to host virtual nodes. A substrate node $i \in N_v$ can either be a transport node or also, a compute node since many network nodes (OpenFlow switches, etc.) can be implemented in software running on servers. Without losing generality, we assume that network controllers can only be run on compute nodes, and that all compute nodes (i.e., in N_c) are able to host any network controller. It is worth noting that, without loss of generality, in this formulation, we are considering that network controllers are supported by the resources available at the multi-domain network (and not user premises).

Any compute node and any transport node $i \in N_v$ that is capable of hosting a virtual node is characterized by an available and maximum node capacity, respectively denoted as γ_i and $\gamma_i^{\max}$ expressed in compute units. Last, the set of types of supported virtual nodes/networks (i.e., OpenFlow, IP router, Ethernet bridges, etc.) is denoted as F_{vnet}. Each type $tp \in F_{\text{vnet}}$ is characterized by a unitary resource cost $\gamma^{(tp,\text{unit})}$, which specifies the amount of resources (in compute units) needed per units of kbps of traffic. The set of substrate nodes that support virtual nodes of type tp is denoted as N_{tp}.

B. Demand model

Each demand is composed of a set of multi-domain virtual networks D. Each virtual network $k \in D$ of type tp^k is a unidirectional graph $G^k = (V^k, E^k)$ where:

- The type tp^k of virtual network k can be OpenFlow, IP, Ethernet bridge, etc. Without losing generality, we assume here that virtual network k is composed of homogeneous nodes, i.e., all of the same type.
- V^k is the set of virtual nodes composing virtual network k as well as the network controller (denoted as ctl^k). $\forall v \in V^k - \{ctl^k\}$, virtual node v is of type tp^k.
- ctl^k's computing resources requirements $\gamma^{(k,clt)}$ expressed in compute units. The user can even express in the demand, the set of compute nodes that are allowed to host the network controller, denoted as $N_{ctl}^k \subseteq N_c$. $\gamma^{(k,clt)}$ is specific to each virtual network and typically depends on the number of nodes under the control of the network controller and from the network Operating System (Network OS) services that the controller implements and to a lesser extent from the network control applications that control the virtual network.
- $E^k \subseteq V^k \times V^k$ the set of virtual links connecting virtual nodes to each other and the network controller to virtual nodes. Each virtual link $(v,v') \in E^k$ has a bandwidth requirement of $b^{k,(v,v')}$, and a maximum transfer delay of $L^{k,(v,v')}$.
- A maximum packet size of p^k.

C. Resource-related assignment variables

The output of the resource allocation algorithm is the set of substrate nodes that will host each virtual node of each virtual network request $k \in D$ (with the required resources) and, also, the set of data paths (routes with the bandwidth allocations at each supporting substrate link and the number of forwarding table entries at each crossed node) that will support each virtual link composing virtual network k (i.e., those connecting virtual nodes but also those that will transport control traffic between the network controller and the nodes). Hence, we distinguish the following variables:

- $z_i^{k,v}$: is a Boolean variable, which specifies whether virtual node $v \in V^k$ is embedded in the substrate node $i \in N_v$, i.e., $z_i^{k,v} = 1$, if node i hosts virtual node v, 0 otherwise. Also, we denote as $z_i^{k,ctl}$ the binary variable that indicates whether node $i \in N_c$ hosts the controller;
- $\phi_{(i,j)}^{k,(v,v')}$: represents the bandwidth assigned to the packets of virtual link $(v,v') \in E^k$ that are flowing from node v to node v' at link $(i,j) \in A$;
- $x_{(i,j)}^{k,(v,v')}$: is a Boolean variable which reflects whether the flow of packets of virtual link $(v,v') \in E^k$ is supported by the substrate link $(i,j) \in A$ (i.e., $x_{(i,j)}^{k,(v,v')} = 0$ if $\phi_{(i,j)}^{k,(v,v')} = 0$; 1 otherwise. These variables will be constrained to prevent any flow-splitting, i.e., one single data-path is used to support virtual link (v,v');
- $\text{TE}_i^{k,(v,v')}$: specifies the number of entries that are installed in node $i \in N_t$ forwarding table to support virtual link (v,v'). Many options can be considered to express switching resource consumption and integrated in the mathematical formulation. Without losing generality, we consider that the number of table entries needed by virtual link (v,v') equals one entry if at least one node i port is either receiving or sending traffic from (v,v').

Thus, $\text{TE}_i^{k,(v,v')}$ is a Boolean variable derived as follows:

$$\forall k \in D, \forall i \in N_t, \forall (v,v') \in V_k, \forall (j,i) \in A:$$

$$x_{(j,i)}^{k,(v,v')} \leq \text{TE}_i^{k,(v,v')} \tag{1a}$$

$$x_{(i,j)}^{k,(v,v')} \leq \text{TE}_i^{k,(v,v')} \tag{1b}$$

$$\text{TE}_i^{k,(v,v')} \leq \sum_{(j,i)\in A} x_{(j,i)}^{k,(v,v')} + \sum_{(i,j)\in A} x_{(i,j)}^{k,(v,v')} \tag{1c}$$

D. Problem Constraints

The constraints related to the placement of virtual nodes $v \in V^k$ are described in inequalities 2 to 6b. Equations 2 force nodes from virtual network k to be only placed on substrate nodes that support virtual node type tp^k. Same logic holds for the network controller in equations 3. Equations 4 ensure the placement of each virtual node on one substrate node. Equation 5 does the same for network controllers.

$$\forall k \in D, \forall v \in V^k - \{ctl^k\}, \forall i \in N_v - N_{tp^k} : z_i^{k,v} = 0 \tag{2}$$

$$\forall k \in D, \forall i \in N_c - N_{ctl}^k : z_i^{k,ctl} = 0 \tag{3}$$

$$\forall k \in D, \forall v \in V^k - \{ctl^k\} : \sum_{i\in N_v} z_i^{k,v} = 1 \tag{4}$$

$$\forall k \in D : \sum_{i\in N_c} z_i^{k,ctl} = 1 \tag{5}$$

Inequalities 6a and 6b constrain the placement of the vertices of a virtual link. Different options can be specified and easily expressed. One option is to force their placement on different domains, or even to constrain their placement on a specific domain. The constraint that we consider below is to force their placement on different substrate nodes.

$$\forall k \in D, \forall (v,v') \in E^k \text{ with } v \text{ and } v' \neq ctl^k, \forall i \in N_v:$$

$$z_i^{k,v} + z_i^{k,v'} \leq 1 \tag{6a}$$

$$\forall k \in D, \forall i \in N:$$

$$\sum_{v\in V_k} z_i^{k,v} \leq 1 \tag{6b}$$

Note that the previous constraints allow the embedding of the network controller and a virtual node on the same compute node. Of course, this can be easily prevented by including ctl^k to inequalities 6a. Another option is to force the placement of all virtual nodes on different substrate nodes (6b). The constraints related to bandwidth allocations are described in equations 7a to 10b. Equations 7a represent the usual flow conservation constraints related to virtual link $(v,v') \in E^k$ that connects two virtual nodes. Equation 7b states that no bandwidth allocation is possible on links connected to compute nodes that do not support virtual nodes. Equations 8a and 8b are the flow conservation constraints that concern the virtual links between the controller and associated virtual nodes.

Equations 9 and inequalities 10a, 10b ensure that path splitting is not used to support virtual links. Equation 9 connects $\phi_{(i,j)}^{k,(v,v')}$ and $x_{(i,j)}^{k,(v,v')}$. Clearly, since path splitting is disabled, the two variables are equivalent and one of them can be removed. This said, both variables are kept for the sake of clarity and generality and, more specifically, to easily derive the variant of this method that allows path splitting.

$$\forall k \in D, \forall (v,v') \in E^k \text{ with } v \neq ctl^k, v' \neq ctl^k, \forall i \in N_s \cup N_t \cup N_v:$$

$$\sum_{(i,j)\in A} \phi_{(i,j)}^{k,(v,v')} - \sum_{(j,i)\in A} \phi_{(j,i)}^{k,(v,v')} = \begin{cases} b^{k,(v,v')} \cdot (z_i^{k,v} - z_i^{k,v'}) & \text{for } i \in N_v \\ 0 & \text{for } i \in N_s \cup N_T - N_v \end{cases} \tag{7a}$$

$$\forall k \in D, \forall (v,v') \in E^k \text{ with } v \neq ctl^k, v' \neq ctl^k, \forall i \in N_c - N_v', \forall (i,j) \in A, \forall (j,i) \in A : \phi_{(i,j)}^{k,(v,v')} = \phi_{(j,i)}^{k,(v,v')} = 0 \tag{7b}$$

$$\forall k \in D, \forall (ctl^k, v') \in E^k, \forall i \in N:$$

$$\sum_{(i,j)\in A} \phi_{(i,j)}^{k,(ctl,v')} - \sum_{(j,i)\in A} \phi_{(j,i)}^{k,(ctl,v')} = \begin{cases} b^{k,(ctl,v')} \cdot (z_i^{k,ctl} - z_i^{k,v'}) & \forall i \in N_v \cap N_c \\ -b^{k,(ctl,v')} \cdot z_i^{k,v'} & \forall i \in N_v - N_c \\ b^{k,(ctl,v')} \cdot z_i^{k,ctl} & \forall i \in N_c - N_v \\ 0, & \forall i \in N_s \cup N_t - N_v - N_c \end{cases} \tag{8a}$$

$$\sum_{(i,j)\in A} \phi_{(i,j)}^{k,(v,ctl)} - \sum_{(j,i)\in A} \phi_{(j,i)}^{k,(v,ctl)} = \begin{cases} b^{k,(v,ctl)} \cdot (z_i^{k,v} - z_i^{k,ctl}) & \forall i \in N_v \cap N_c \\ -b^{k,(v,ctl)} \cdot z_i^{k,ctl}, & \forall i \in N_c - N_v \\ b^{k,(v,ctl)} \cdot z_i^{k,v} & \forall i \in N_v - N_c \\ 0 & \forall i \in N_s \cup N_t - N_v - N_c \end{cases} \tag{8b}$$

$$\forall k \in D, \forall (v,v') \in E^k, \forall (i,j) \in A :$$

$$\phi^{k,(v,v')}_{(i,j)} = b^{k,(v,v')} \cdot x^{k,(v,v')}_{(i,j)} \tag{9}$$

$$\forall k \in D, \forall (v,v') \in E^k$$

$$\sum_{(i,j)\in A} x^{k,(v,v')}_{(i,j)} \leq 1 \tag{10a}$$

$$\sum_{(j,i)\in A} x^{k,(v,v')}_{(j,i)} \leq 1 \tag{10b}$$

In addition to inequalities 7, inequality 11 explicitly connects the placement of a virtual network element with the placement of the virtual links to which it belongs.

$$\forall k \in D, \forall v \in V^k, \forall i \in N_v:$$

$$z^{k,v}_i \leq \sum_{(v,v')\in E^k} \sum_{(i,j)\in A} x^{k,(v,v')}_{(i,j)} + \sum_{(v',v)\in E^k} \sum_{(j,i)\in A} x^{k,(v',v)}_{(j,i)} \tag{11}$$

The following inequalities ensure that the resource assigned to demand D does not exceed the remaining resources at the substrate network, namely substrate links, nodes, slices, and compute nodes. Inequalities 12 ensure that the bandwidth assigned to all virtual links from all virtual networks of D does not exceed the remaining bandwidth on each substrate network's link.

$$\forall (i,j) \in A : \quad \sum_{k\in D} \sum_{(v,v')\in E^k} \phi^{k,(v,v')}_{(i,j)} \leq \gamma_{i,j} \tag{12}$$

Inequalities 13 ensure that bandwidth allocations assigned to all virtual links respect substrate nodes' and exposed slice's remaining forwarding capacities.

$$\forall i \in N_t \cup N_s : \quad \sum_{k\in D} \sum_{(v,v')\in E^k} \sum_{(j,i)\in A} \phi^{k,(v,v')}_{(j,i)} \leq FR_i \tag{13}$$

The constraints 14 ensure that the flow table entries needed by the virtual links composing the demand D remain below the remaining forwarding table size. As a variant to this formulation, if virtual nodes have a requirement of, let's say, $TE^{k,v}$, then such a requirement can be easily integrated into inequalities 14 by adding for nodes $i \in N_v$ the required table size $\sum_{k\in D} \sum_{i\in N_v} z^{k,v}_i \cdot TE^{k,v}$.

$$\forall i \in N_t : \quad \sum_{k\in D} \sum_{(v,v')\in E^k} TE^{k,(v,v')}_i \leq TE_i \tag{14}$$

Inequality 15 ensures that the remaining computing capacity in compute nodes and transport nodes suffices to support assigned virtual nodes and eventually network controllers. The first sum computes the resources consumed by the virtual node v. It adds the bandwidth from all virtual links that end at v weighted by the unitary resource cost (i.e. $\gamma^{tp^k,unit}$). Indeed, as stated above, the resources consumed by a virtual node are proportional to the rate of incoming traffic. Referring to inequalities 2, if a node $\forall i \in N_c - N_v$, $z^{(k,v)}_i = 0$, this first term is zeroed. The second term concerns compute nodes that can host network controllers (refer to equality (3)) and adds the amount of resources that controllers consume.

$$\forall i \in N_v \cup N_c :$$

$$\sum_{k\in D} \sum_{v\in V^k - \{ctl^k\}} \sum_{\substack{(v',v)\in E^k \\ v'\in V^k - \{ctl^k\}}} z^{k,v}_i \cdot b^{k,(v',v)} \cdot \gamma^{tp^k,unit} + \sum_{k\in D} z^{k,ctl}_i \cdot \gamma^{k,clt}_i \leq \gamma_i \tag{15}$$

Inequality 16 ensures that the maximum latency associated with each virtual link is met. As path-splitting is disabled, the transfer delay that a packet experiences is the accumulation of the delays experienced at each hop. Each hop, through either a transport node or a slice node, induces a packet transmission time bounded by $\frac{p^k}{b^{(k,(v,v'))}}$ for packets belonging to virtual link (v,v'), and switching and propagation delays bounded by latency $l_{(i,j)}$ for any link $(i,j) \in A_t \cup A_s$, where A_t (resp. A_s) are the sets of edges in A whose one end is either a transport node or a slice node. The latency constraints are as follows.

$$\forall k \in D, \forall (v,v') \in E^k :$$

$$\sum_{(i,j)\in A_t \cup A_s} x^{k,(v,v')}_{(i,j)} \left(\frac{p^k}{b^{k,(v,v')}} + l_{(i,j)} \right) \leq L^{k,(v,v')} \tag{16}$$

E. Objective function

Different objective functions can be adopted with this formulation. Basically, the set of variables used in this formulation allows expressing different ways of efficiently using the substrate network resources and also effectively spreading the assigned resources to improve the admissibility/acceptance of forthcoming requests. One option is to minimize the highest link/node utilization rate of the substrate network. Another option is to minimize a utilization threshold that applies to all substrate network elements. Another criterion is to minimize the number of active elements (compute nodes, transport nodes and even active links). Of course, all these objectives can be expressed with this formulation. Below, the objective function of equation (17) is only focused on minimizing the node and link resources that are needed to support the request D. Four terms compose the objective function. The first aim at minimizing the average utilization of the network substrate's links while the second aims to minimize the average utilization of substrate nodes' forwarding table size. The objective of the last two terms is to minimize the average computing resource utilization that is used to support virtual nodes and network controllers on the substrate network's nodes. Other similar sub-objectives can also be added, especially those related to the packet forwarding capacity of transport and slice substrate nodes.

$$
\begin{aligned}
&\alpha_1 \cdot \frac{1}{|A''|} \cdot \sum_{(i,j)\in A} \left(\frac{1}{\gamma^{max}_{(i,j)}} \cdot \left(\gamma^{max}_{(i,j)} - \gamma_{(i,j)} + \sum_{k\in D} \sum_{(v,v')\in E^k} \phi^{k,(v,v')}_{(i,j)} \right) \right) \\
&+ \alpha_2 \cdot \frac{1}{|N_t|} \cdot \sum_{i\in N_t} \left(\frac{1}{\mathrm{TE}^{max}_i} \cdot \left(\mathrm{TE}^{max}_i - \mathrm{TE}_i + \sum_{k\in D} \sum_{(v,v')\in E^k} \mathrm{TE}^{k,(v,v')}_i \right) \right) \\
&+ \alpha_3 \cdot \frac{1}{|N_v|} \cdot \sum_{i\in N_v} \left(\frac{1}{\gamma^{max}_i} \cdot \left(\gamma^{max}_i - \gamma_i + \sum_{k\in D} \sum_{v\in V^k - \{ctrl^k\}} z^{k,v}_i \cdot b^{k,(v',v)} \cdot \gamma^{tp^k,unit} \right) \right) \\
&+ \alpha_4 \cdot \frac{1}{|N_c|} \cdot \sum_{i\in N_c} \left(\frac{1}{\gamma^{max}_i} \cdot \left(\gamma^{max}_i - \gamma_i + \sum_{k\in D} z^{k,ctl}_i \cdot \gamma^{k,clt} \right) \right)
\end{aligned}
\tag{17}
$$

IV. Service implementation and results

A. Considered use case and objectives

We consider federated military networks to showcase our proposed service. In these networks, ally nations connect and share some of their network infrastructure to build a multi-domain Federated Mission Network (FMN) that spans a large region with points of presence at the theater of operation but also at very remote headquarters, command posts or data centers. The goal is to leverage on this multi-domain FMN to provide the end-to-end multi-domain communication services needed during the coalition military missions. The end-to-end service provided on a such network is standard IP connectivity between end-points. Also, a crucial aspect is to efficiently use the resources provided by nations and maximize the support of the end-to-end services needed for the mission. In this context, the service that we are proposing allows a nation to deploy and operate wide-area virtual networks whose nodes can be located at very remote locations and provided by different nations. Since the requesting nation is granted full control to its virtual networks, it can enforce its own policies (QoS, security, etc.) regardless of the policies used by the other nations that contribute to the provision of the virtual network. Moreover, these policies can be changed over time depending on the mission's needs. Also, each virtual network can be programmed to provide customized and novel services (not only IP). This is particularly the case when provisioning an OpenFlow-based Virtual network, which allows a flow-based forwarding at the level of the virtual network (as opposed to standard IP destination-based forwarding). The objectives of the implementation are twofold. First, to show the feasibility of provisioning an OpenFlow-based multi-domain virtual network on an OpenFlow-enabled multi-domain substrate network. Second, demonstrate the operation of the proposed service by enforcing an ad-hoc QoS policy and highlight some of its benefits.

B. Implemented prototype

The network platform is composed of three domains belonging to three different nations. Fig.4 describes the considered multi-domain substrate network combining the aggregated topologies exposed by the three domains. For simplicity, we assume that the exposed topologies resume to domain-level topologies. This avoids dealing with the topology aggregation policy of each domain. Hence, it avoids the need of having a domain-level resource allocation since the embedding that we propose at the multi-domain (i.e., aggregated) level embeds the virtual network by considering all domain-level resources (and not an abstraction of these). Each domain is an OpenFlow-based network with one domain-level OpenFlow controller and a couple of OpenFlow switches. OpenFlow traffic between the controller and the switches are exchanged in "in-band" mode by assigning a dedicated VLAN and installing the appropriate OpenFlow rules at each switch. This is part of the domain network initialisation. Most switches are emulated OVS (OpenVswitch) switches with the Containernet emulation tool [9]. These are depicted in green in the figure. Some others are PICA8 p3295 physical switches that, in turn, implement a hardware-optimized version of OVS. We assume that only the physical switches can host OpenFlow virtual switches. A Compute node (server or datacentre abstraction) is also exposed by domain 3. From a network management perspective, each domain has a "Service Orchestrator" (SO) in charge of orchestrating the provisioning of the multi-domain services initiated by the domain. When solicited by other

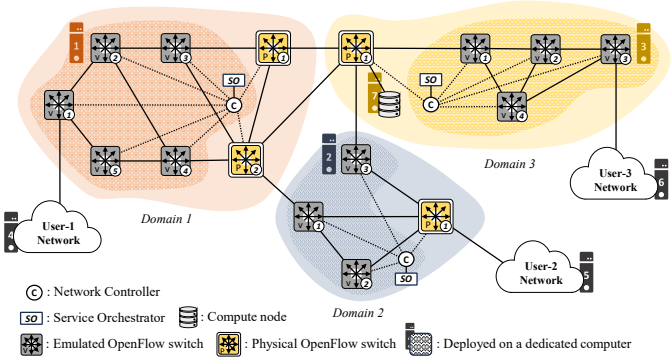

Fig. 4. Network set-up.

domains, it also contributes to the provisioning of some portions of their accepted services [10]. For simplicity, the management traffic exchanged between the SOs is supported "out-of-band" via a dedicated management network. Each domain has a dedicated computer to run its Containernet network emulation tool, the Ryu controller and the Service Orchestrator.

The resource allocation algorithm was implemented on the Cplex solver. It is executed by the "Service Orchestrator" of the initiating domain upon the arrival of the service demand. Under the above-cited assumption, when the demand is accepted, the algorithm outputs all the required resources (related links, switches, compute nodes from all crossed domains) for each virtual network. The SO launches the service deployment stage by requesting the deployment of virtual nodes, virtual links, and controllers. On its domain, it instructs its physical switches to instantiate one or multiple virtual nodes, its compute nodes to instantiate a controller, and its domain's OpenFlow controller to generate the appropriate OpenFlow rules to set up some portions of virtual links. Also, it instructs other SOs to provision the needed resources (virtual nodes, portions of virtual links, etc.) on the domain they belong to.

For simplicity, each virtual data link is assigned a unique VLAN identifier within a domain. To provision a virtual link, OpenFlow rules that match this VLAN identifier are installed on the switches along its path. If the virtual link crosses another domain, the VLAN identifier is rewritten accordingly. Virtual control links are assigned one VLAN identifier, their deployment on the substrate network relies on OpenFlow rules that jointly match the VLAN-ID and the IP addresses of the controller and virtual nodes and transport layer information, i.e., TCP protocol and port numbers. Clearly, this solution is not scalable with the number of virtual links and can only be considered for prototyping. A more effective solution is to rely on MPLS (Multi-protocol label Switching) labels or even on appropriately formatted MAC addresses. Regarding virtual OpenFlow node instantiation, they take place at the physical OpenFlow (OF) switches. A first OF bridge is instantiated, and all the physical links to which the physical switch is connected to are attached to it. We denote this switch as OF-sw. The instantiation of a virtual OF node is realized by instantiating a new OF bridge. We denote this virtual node

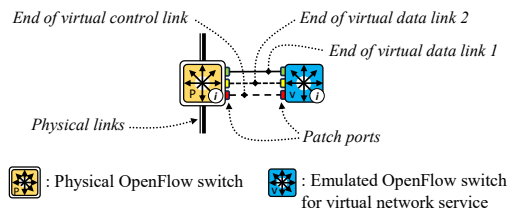

Fig. 5. Virtual node implementation.

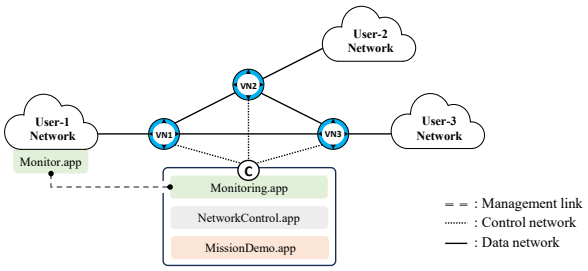

Fig. 6. Service request with control application.

as OF-vnode. As shown in Fig.5, OF-vnode is connected to OF-sw via multiple "patch" ports: One "patch" port per virtual data link and one "patch" port for the control link that conveys OpenFlow traffic to its controller. As each virtual link is identified with a specific VLAN, "patch" ports are attached to the corresponding VLAN, and OpenFlow rules based on the appropriate VLAN-ID are installed on OF-sw to route the data and control traffic to the right virtual node.

C. Results Overview

1) Service request provisioning: We consider below a service request composed of one OF-based virtual network described in Fig.6. This latter specifies a topology that represents the expected virtual network with some QoS requirements (bandwidth, latency, etc.). In our case, we are deploying a virtual network composed of 3 OpenFlow virtual switches and one network controller. Three user networks are reachable from this virtual network. A management link connects the user 1 network to the controller. The controller runs two control applications. The first, described below, programs the forwarding of the virtual network according to user needs. The second is a monitoring application reporting the state of the virtual network to a control center located in user1 network. Fig.7 describes the resources computed by the resource allocation algorithm and assigned to the requested service across the multi-domain network. The data network component of the virtual network is as follows. The virtual nodes were placed on three different physical switches (right-hand side) as enforced by the embedding algorithm. The embedding virtual data links are depicted in blue. The controller of the virtual network is placed on the unique compute node belonging to domain 3. The data path assigned to the three virtual control links, which also compose the control component of the virtual network, are depicted in green. Last, the embedding of the management link that connects the controller to user1's network is depicted in red.

2) Virtual network operation: To show the operation of this service, the requested service is deployed with a dedicated control application. The control application demonstrates the ability to enforce any ad-hoc QoS policy on the virtual network. This policy is a basic representation of the policies used in military networks. Two levels of priority are considered: one for video streams and one for

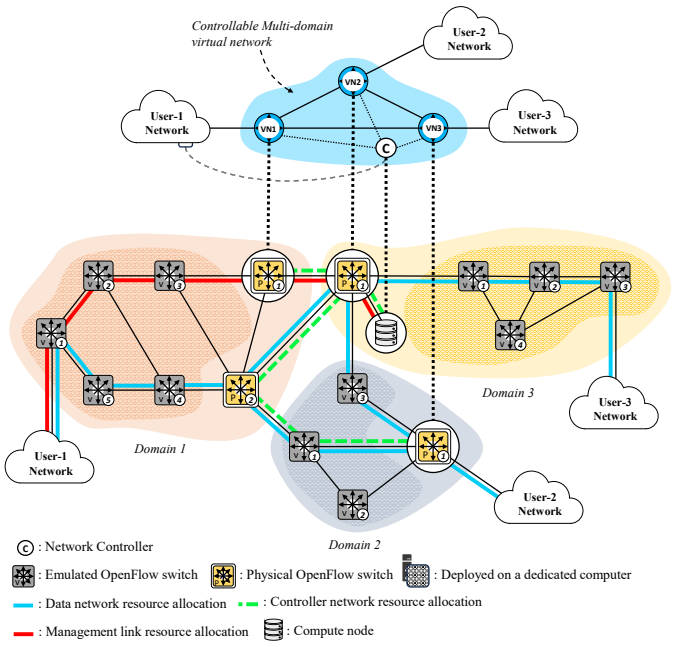

Fig. 7. Provisioned resources.

low-priority data streams. The QoS policy implemented in the control application ensures that at any moment a high-priority flow (video stream) has precedence on data streams on any network resource. When a new video stream is launched, if resources are available (not used by already admitted video flows), the control application programs the virtual switches to route the video stream. In order to protect the resources assigned to the new video stream, when possible, some existing low-priority streams may be rerouted along longer and less efficient paths. If such paths are not available, they are interrupted, i.e., traffic is filtered/dropped at the network entrance. Fig.8 highlights the control application workflow.

The following scenario is played to illustrate the enforcement of the QoS policy on the deployed virtual network. First, a data stream is detected from user2 to user1 and reported to the network controller, which assigns to the flow the shortest path (Fig.9, step 1). A video stream is set from user2 to user1, the controller detects a resource constraint violation on the shortest path. Then it assigns the shortest path to the newly arrived video flow and reroutes the data stream on the longer path via the upper part of the network via virtual switch 2 (Fig.9, step 2). A video stream and a low-priority data stream are then set from user3 to user1, no more free paths are available, the controller decides to assign the shortest path to the video stream and to block low-priority streams (Fig.9, step 3). When the first video stream stops, the network controller detects it and re-assigns the data paths to the data streams. This first data stream restarts on the shortest path and the second gets assigned the long path (Fig.9, step 4). When the second video stops, the controller detects it and the second data stream is redirected to the shortest path. All of the logic is coded in the network control application run at the controller.

V. Discussion and Related Work

The service proposed in this paper offers to the user the opportunity to have a dedicated network that span multiple administrative domains that it can fully manage and fully control in order to provide customized multi-domain services. This network is composed of virtual nodes optimally scattered throughout the multi-domain network. When using Openflow, virtual nodes support a finer-grained packet forwarding than what an IP router can do. Sophisticated, homogeneous and dynamic traffic management policies can be enforced on the virtual network even if virtual nodes are hosted in different domains. Also, different control applications can be executed by the controller over time, allowing the user to adjust the forwarding policy according to its current needs. Naturally, many such virtual networks can be provisioned at the same time through the multi-domain network. They can be requested on demand and, possibly, deployed in the time scale of minutes rather than days.We have shown that our proposal can be useful in a FMN context. We also think similar interest exists in the civilian context since service providers usually have privileged partners; their respective domains may enable such service for their own use or for their clients.

This service assumes that, based on network/device virtualization, domains will “logically” share their network devices with other domains and provide them with some decent programming capabilities on these virtualized devices.

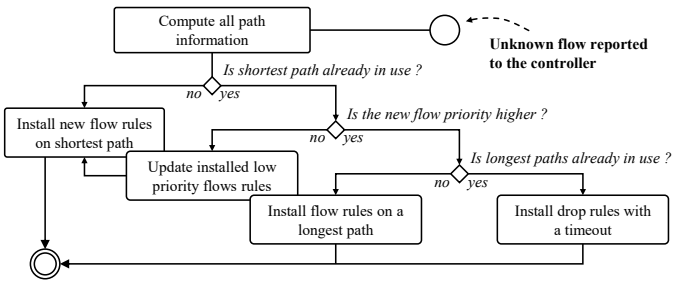

Fig. 8. Control application generic behavior.

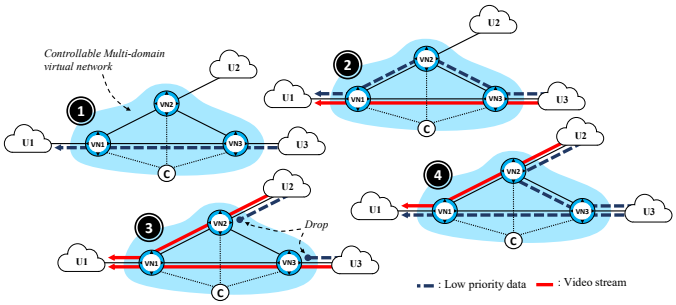

Fig. 9. Main steps of the considered scenario.

This is much more demanding and committing than providing "best-effort" IP transit, as in the current multi-domain networks, and puts a lot of security and performance commitments/requirements on the hypervisor in charge of device virtualization.

As mentioned above virtual link and virtual network embedding have been extensively researched for more than a decade as shown in [11] [12]. Different variants of the algorithms were proposed by considering various resources, resource cost, optimization criteria and resolution methods. With the advent of network slicing and VNF service chaining, some recent research has also considered the context of multi-administrative multi-domain networks. Toumi et al. [13] adopt centralized framework, solve multi-objective (latency, bandwidth, cost) placement of multi-domain SFCs with use of Physical Programming optimization. Such service however promotes a fixed packet flow behavior (defined via SFC order) as opposed to controllable service proposed by us. The complex multi-domain multi-stratum architecture (Taleb et al. [14]) concentrates only on the coarse-grained slice setup and modification (scale up/down VNF resources, use VNF in another domain etc.) and does not address fined-grained slice programmability by the slice tenant the way as we propose. To the best of our knowledge, the peculiarities of our proposed method are as follows. First, the consideration of the three components of the virtual network: data, control and management components. Also, our algorithm considers the switching resources: forwarding table size as well as the forwarding rate. Last, it takes into account domain-level network slices.

In conclusion, in this work we go beyond current works (e.g., [11] [12]) and discuss the architecture of fully controllable multi-domain virtual network. We do not only focus on embedding the virtual links/networks to satisfy QoS (with the proposed LP formulation) but assume programmability of such embedded service via dedicated controller. The placement of such SDN controller is a decision parameter in our model. Our approach promotes more flexibility and fined-grained policies compared to previous works.

VI. Conclusion

A controllable virtual network deployed on a multi-administrative multi-domain network offers interesting opportunities. The ability to enforce homogeneous and dynamic traffic management policies on the virtual network even if the virtual nodes are hosted by different domains. The ability to leverage the virtual network to provide customized and novel end-to-end service on top of a multi-domain network. In this paper, we have proposed a resource allocation algorithm for such service. We have also described a prototype implementation on top of an SDN enabled multi-domain network and elaborated on its benefits. Some perspectives to this work are as follows. To consider the case of dynamic virtual network with a changing topology or QoS requirements. Another point is how to derive the virtual network topology and performance requirements from high-level user intents.

Acknowledgment

This work was supported by the European Defense Agency (EDA) project No B 1520 IAP4 GP" Software Defined Tactical and Theatre Network (Softanet)".

References

[1] R. Bruschi, F. Davoli, P. Lago, A. Lombardo, C. Lombardo, C. Rametta, and G. Schembra, "An SDN/NFV Platform for Personal Cloud Services," *IEEE Transactions on Network and Service Management*, vol. PP, pp. 1–1, Oct. 2017.

[2] R. Alvizu, G. Maier, S. Troia, V. M. Nguyen, and A. Pattavina, "SDN-based network orchestration for new dynamic Enterprise Networking services," in *2017 19th International Conference on Transparent Optical Networks (ICTON)*, Jul. 2017, pp. 1–4, iSSN: 2161-2064.

[3] Z. Dai, X. Wang, B. Yi, M. Huang, and Z. Li, "An SDN-Based Self-adaptive Resource Allocation Mechanism for Service Customization," in *Wireless Algorithms, Systems, and Applications*, ser. Lecture Notes in Computer Science, Z. Liu, F. Wu, and S. K. Das, Eds. Cham: Springer International Publishing, 2021, pp. 192–199.

[4] M. T. Beck and J. F. Botero, "Scalable and coordinated allocation of service function chains," *Computer Communications*, vol. 102, pp. 78–88, Apr. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366416303577

[5] A. Boubendir, E. Bertin, and N. Simoni, "Flexibility and dynamicity for open network-as-a-service: From VNF and architecture modeling to deployment," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, Apr. 2018, pp. 1–6, iSSN: 2374-9709. [Online]. Available: https://ieeexplore.ieee.org/document/8406135

[6] L. U. Khan, I. Yaqoob, N. H. Tran, Z. Han, and C. S. Hong, "Network Slicing: Recent Advances, Taxonomy, Requirements, and Open Research Challenges," *IEEE Access*, vol. 8, pp. 36 009–36 028, 2020, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/document/9003208

[7] G. Hallingstad and S. Oudkerk, "Protected core networking: an architectural approach to secure and flexible communications," *IEEE Communications Magazine*, vol. 46, no. 11, pp. 35–41, Nov. 2008, conference Name: IEEE Communications Magazine. [Online]. Available: https://ieeexplore.ieee.org/document/4689242

[8] S. Pédebéarn, S. Abdellatif, P. Berthou, D. Nogalski, and D. Belabed, "Virtual Link Embedding in Collaborative Sliced Multi-Administrative Multi-Domain Networks," Apr. 2024. [Online]. Available: https://laas.hal.science/hal-04502307

[9] "containernet/containernet," Mar. 2024, original-date: 2016-10-14T11:55:30Z. [Online]. Available: https://github.com/containernet/containernet

[10] N. Sousa, D. Perez, R. Rosa, M. Santos, and C. Esteve Rothenberg, "Network Service Orchestration: A Survey," *Computer Communications*, vol. 142, Mar. 2018.

[11] H. Cao, S. Wu, Y. Hu, Y. Liu, and L. Yang, "A survey of embedding algorithm for virtual network embedding," *China Communications*, vol. 16, no. 12, pp. 1–33, Dec. 2019, conference Name: China Communications. [Online]. Available: https://ieeexplore.ieee.org/document/8968720

[12] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013, conference Name: IEEE Communications Surveys & Tutorials. [Online]. Available: https://ieeexplore.ieee.org/document/6463372

[13] N. Toumi, O. Bernier, D.-E. Meddour, and A. Ksentini, "On Using Physical Programming for Multi-Domain SFC Placement With Limited Visibility," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2787–2803, Oct. 2022, conference Name: IEEE Transactions on Cloud Computing. [Online]. Available: https://ieeexplore.ieee.org/document/9305231

[14] T. Taleb, I. Afolabi, K. Samdanis, and F. Z. Yousaf, "On Multi-Domain Network Slicing Orchestration Architecture and Federated Resource Control," *IEEE Network*, vol. 33, no. 5, pp. 242–252, Sep. 2019, conference Name: IEEE Network. [Online]. Available: https://ieeexplore.ieee.org/document/8758980