A Secure Multi-Tenant Framework for SDN

Hao Jiang, Ahmed Bouabdallah, Amin Aflatoonian, Jean-Marie Bonnin
Institut Mines/Telecom – Telecom Bretagne
Site of Rennes
Network, Security and Multimedia Department
35576 Cesson Sévigné – France
{hao.jiang, ahmed.bouabdallah, amin.aflatoonian, jm.bonnin}@telecom-bretagne.eu

Karine Guillouard
Orange Labs
Multi Access Convergence Architecture
4 rue du Clos Courtel
35512 Cesson Sévigné – France
karine.guillouard@orange.com

ABSTRACT
Software-Defined Networking (SDN) promises a flexible and programmable solution for future networks. By extracting the control logic out of forwarding devices into a specific entity as the control plane, it dramatically eases the management work of multi-tenant networks, where several customers share same network resources. Depending on the way and the SDN layer that tenants can interact with, they can be allowed to have higher and differentiated levels of control over their own slices of available resources. This paper discusses multi-tenancy in SDN by proposing a framework on SDN northbound that focuses as a matter of priority on isolation and access control. A new network abstraction layer is introduced between the control layer and application layer on top of which tenants are provided unified APIs with abstract views and pre-defined levels of control over their dedicated virtual networks, with no concerning about the underlying type and number of controllers as well as topology of physical networks. A developed PoC finally shows the soundness of our approach by implementing various levels of isolation together with AAA functions.

CCS Concepts
• Security and privacy~Access control • Security and privacy~Security protocols • Networks~Logical / virtual topologies • Networks~Network manageability • Networks~Programmable networks

Keywords
Software-Defined Networking; Multi-tenancy; Isolation; Access Control.

1. INTRODUCTION
During the past few years, Software-Defined Networking has evolved to be a new network paradigm that gives hope to change limitations of current network infrastructures [1]. By separating and abstracting control plane functions from network elements into a logically centralized entity, which is called SDN controller, SDN simplifies the complex management of various flows, enables programmability and provides better virtualization. In a SDN environment, the network applications communicate and send their network service requests to the controller via Northbound Application Programming Interfaces (NBIs). Accordingly, the controller translates the requests into low-level forwarding rules and installs them in the data plane network devices via Southbound Application Programming Interfaces (SBIs), while providing relevant information up to the network applications [2]. Both the academia and the industry have put a lot of research effort on SBIs, leading to the birth of the OpenFlow protocol as the first SBI standard for the communications between the controller and forwarding devices [3].

Network-as-a-Service (NaaS) is frequently offered in a multi-tenant style, where customers, who are also called tenants, and their end-users share network infrastructure and services, while they are strictly logically isolated from each other [4]. SDN has naturally provided a direct approach to the provision of virtual network services by owners of the network infrastructures to third parties. It leads to various multi-tenancy situations : on different layers, for different purposes, using different techniques, each of which provides different levels of control while requiring different types of isolation among users. For instance, we can have southbound multi-tenancy with several guest controllers sharing the same data forwarding elements, or we may prefer northbound multi-tenancy with several guest applications sharing the whole SDN infrastructure including the SDN controller.

Some solutions have been proposed in order to implement multi-tenancy in SDN environments [5]. FlowN is one example that is designed to provide container-based control-plane virtualization to enable multi-tenancy on the northbound [6]. FlowVisor, a typical southbound multi-tenancy solution, enables multiple guest controllers transparently managing their own slice of networks on top of the same network infrastructure [7]. However, there are quite few comprehensive researches on SDN multi-tenant models. Moreover, the multi-tenancy on northbound is rarely researched due to the lack of standardization [8].

We propose in this paper a comprehensive definition of Software Defined Multi-Tenant Networking (SDMTN), examine the feasible models along with available enabling techniques, and then we investigate some representative solutions that currently exist in academia or industry. In addition, we propose a new framework for providing northbound multi-tenancy of SDN, and then, we evaluate the framework by designing and implementing a prototype based on Mininet and OpenDaylight [13]. Our solution provides satisfactory level of isolation, domain-based fine-grained access control, and good interoperability among SDN provider and tenants.

2. SOFTWARE-DEFINED MULTI-TENANT NETWORKING

2.1 Principles and Characteristics of SDN and NFV
According to Open Networking Foundation [1], SDN architecture mainly comprises three planes, which are respectively Data Plane, Control Plane, Application Plane, and recently many articles often explicitly indicate a management component that vertically spreads interaction of the aforementioned three planes.Inter-plane interactions are provided through NBIs and SBIs.

SIN, July 20-22, 2016, Rutgers University, New Jersey, USA.
 ISBN 978-1-4503-4764-8/16/07...$15.00
DOI: http://dx.doi.org/10.1145/2947626.2947641

NFV (Network Function Virtualization) has to address the operational challenges and high costs of managing the closed and proprietary appliances presently deployed throughout telecommunication networks, by virtualizing and consolidating the network functions using standard IT virtualization technologies that run on high volume servers, switches and storage [9]. These services and functions are in use today as firewalls, IDS/IPS, WAN acceleration, WAN load balancer, AAA, NAT and so on and so forth.

In practice, often a single network service function will not provide by itself a complete product or service for the customer, some of these services will have to be concatenated. By the way, a user, an application, or a content flow must pass through several service functions before being delivered. This is commonly referred as the concept of Service Function Chaining (SFC).

Service Function Chaining is not a new concept. In traditional telecommunication network, hardware-based approach makes it extremely complex and time-consuming to implement, and expensive to scale and manage. With an NFV environment, it becomes much easier, more efficient and scalable. A service provider who follows NFV paradigm will implement a number of virtual network functions (VNFs) as software running on VM of high performance, where multiple VNFs can be used in sequence to deliver a service.

2.2 Multi-Tenancy in SDN/NFV

2.2.1 Definition and Requirements

Software-Defined Multi-tenant Networking (SDMTN) can be defined as a property of network where both control and service functions are software-defined by adopting SDN/NFV principles and the following requirements [2, 5, 10]. **SDN-based**: SDN / OpenFlow allows multiple tenants to transparently share the same set of forwarding devices, and the operator to centrally and dynamically manage all the tenant networks through the SBI of his SDN controllers. **NFV-driven**: NFV allows multiple tenants to share the same platform for the same service functions, and the operator to centrally and dynamically manage all the services through for example the NBI of his SDN controllers or via the transversal management layer. **Tenant Manageability**: the operator should be able to provide tenants with means to manage and monitor themselves their own virtual networks by opening his SDN controllers on its northbound side given by NBI. **Highly Isolated**: tenants should not be aware of and affected by others existence. This includes traffic isolation, control isolation, performance isolation and address space isolation [11]. **Fine-grained Access Control**: the behavior of tenants and their end users should be strictly controlled according to pre-defined policies.

2.2.2 Multi-tenancy Models

According to our definition, tenants should be able to communicate with the controlling functions offered by SDN provider to control their own networks. Theoretically, tenants are able to connect with the SDN provider in different ways. This section mainly analyses possible models. Note: different colors stand for different trust domain which by default is blue color of SDN provider.

2.2.2.1 Northbound Multi-Tenancy

As is shown in Figure 1, in this case, tenants connect their network application with a single controller owned by SDN provider through northbound APIs. The controller can provide an abstract view of the dedicated virtual network that belongs to a tenant, and can also easily manage and monitor all the tenants' behaviors targeting to their networks. This model is flexible and can be further extend to BYOC model discussed below.

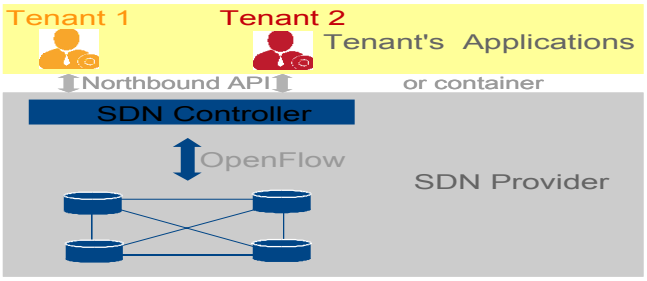

Figure 1. Northbound Multi-Tenancy.

2.2.2.2 Southbound Multi-Tenancy

As is shown in Figure 2, the network elements in data plane are directly connected by tenant controllers. In this case, the controller of SDN provider acts as a master controller and assigns each virtual network to the management of the corresponding tenant controller.

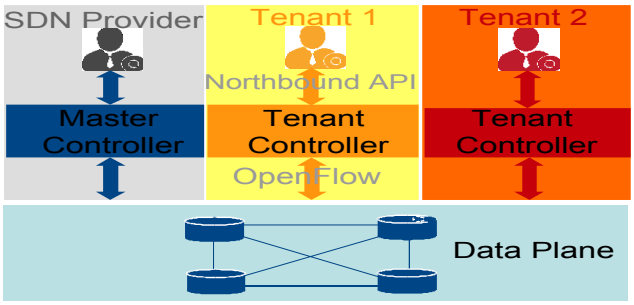

Figure 2. Southbound Multi-Tenancy.

2.2.2.3 BYOC Multi-Tenancy

According to Figure 3, the SDN provider offers an abstract of resources through the controller, while tenants are connected to the SDN controller and manage their dedicated abstract network with their own controllers. In this model, the SDN provider can offer a possibility to tenants to have higher level of control over their own networks, according to Bring Your Own Control (BYOC) principles [12]. In addition to Northbound APIs, another way to do it consists in using "containers", such as FlowN [6].

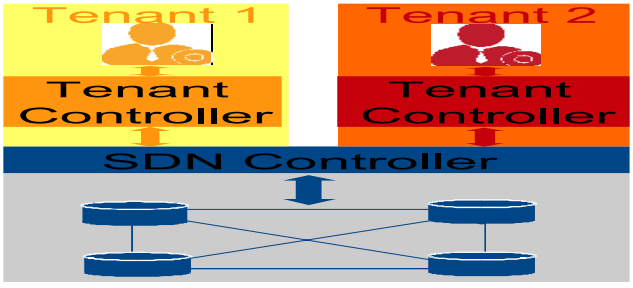

Figure 3. BYOC Multi-Tenancy.

2.3 Comparing Existing Solutions

We recall and compare two main multi-tenant solutions designed for SDN environment, FlowN and FlowVisor, respectively as examples for northbound multi-tenancy model and southbound multi-tenancy model.

2.3.1 The FlowN approach

FlowN is a northbound multi-tenant solution for SDN environment [6]. It provides separate containers on the controller for each tenant to run its application. The controller maps the data traffic of each tenant from the virtual network at the controller plane to the physical network elements in the data plane. For *control isolation*, it adopts data-control mapping method. FlowN allows tenants to run their applications on LXC containers. Using these containers, the applications are able to communicate with the controller by function call and callback APIs. The controller is responsible for providing an abstract view of the resources at the underlying network to the applications. It receives requests from the tenant applications and translates them into the forwarding rules on the network elements in the data plane. Alternatively, the controller receives the requests from the data plane and forwards them to the tenant applications [11]. For *traffic isolation* in the data plane, it adopts encapsulation method. It assigns a VLAN tag for each tenant and these tags are appended to the traffic at the ingress switch and will be removed at the egress switch. However, according to one of the authors, FlowN is not any more under development.

2.3.2 The FlowVisor approach

FlowVisor perfectly illustrates southbound multi-tenancy [7]. It introduces a separate hypervisor layer between the forwarding devices and the controllers to provide isolation and virtualization. FlowVisor is located in the middle of controller plane consisting of multiple tenant controllers and the data plane consisting of forwarding elements. For *traffic isolation*, it divides the network resources into slices according to slicing policies, and allocate each network slice as a virtual network to each tenant's controller. For *control isolation*, it uses OpenFlow message rewriting method. It intercepts all OpenFlow messages, rewrites and forwards them to the corresponding tenant. However, as a common issue of southbound multi-tenancy model, the underlying network infrastructures are exposed to the tenants. Indeed, in the case of FlowVisor, with slicing, the mapping between virtual and physical topologies is visible to the tenants [6].

2.3.3 Comparison

Table 1 gives a comprehensive comparison between FlowN and FlowVisor.

Table 1. FlowVisor-FlowN Comparison

	FlowVisor	**FlowN**
SDMTN Model	Southbound	Northbound
Control Isolation	OpenFlow Message Rewriting	Data-Control Mapping
Data Isolation	Slicing	Encapsulation
Address Space Isolation	Restricted	Full
Performance Isolation	Bandwidth/Switch CPU	Bandwidth
Virtualization Layer	Southbound	Control Plane

2.4 Problematics

There are mainly three models for SDMTN, according to different approaches how tenants communicate with SDN provider, which respectively are southbound multi-tenancy, northbound multi-tenancy and BYOC method as a special variation from northbound multi-tenancy. With the support of OpenFlow as a standard protocol for southbound API, most of current multi-tenant solutions are based on southbound multi-tenant model. However, with more and more research effort turning into northbound, we believe northbound multi-tenancy model has potentials to provide a better performance of multi-tenancy in the future, with higher level of flexibility, security, and virtualization.

3. A SECURE MULTI-TENANT FRAMEWORK FOR SDN

We design on top of an SDN controller, an elementary multi-tenant framework based on SDMTN northbound multi-tenant model. We firstly introduce the requirements of the design and then we illustrate the main principles involved in the proposed framework. We further explain the architectural components and how they interact with each other to complete multi-tenant functions.

3.1 Requirements

Our framework should be designed with the following requirements. **Controller Independent**: this is a rational requirement that can be realized with a standard protocol on the northbound, which however currently does not exist. **Effective Isolation**: as traffic isolation is managed under the control plane, thus our framework is based on the assumption that the underlying master controller supports traffic isolation techniques such as encapsulation. In our solution we mainly focus on control isolation. **Tenant Manageability**: this property has already been identified for SDTMN (cf. §.2.2.1). **Fine-grained Access Control**: the behavior of tenants and their end users should be strictly controlled according to pre-defined policies. **Full Virtualization**: the framework should be able to provide an abstract view of the underlying subset of network resources, in the form of a virtual network to each tenant. In addition, the controller of SDN provider should also be abstracted so that tenants are unaware of any underlying controllers. **Evolutional**: the framework should be able to evolve and adapt to SDN northbound evolution. It should be extendable to BYOC model.

3.2 An Abstract Solution

Our framework introduces a new abstraction layer on top of the control plane. This multi-tenancy abstraction should allow network applications of tenants to express their desired network behaviors without being responsible for implementing that behavior itself. This approach maps abstract configurations that the tenants' applications express based on a simplified, abstract model of the given tenant network into global configurations for the network view exposed by the SDN controller.

As is depicted in Figure 4, the new layer defines a secure boundary between the trust domains of SDN provider and its tenants. Every request from tenants needs to be authorized and then translated into lower level management language that can be recognized by the SDN controller. In reverse direction, the global events or information from the underlying physical network are mapped to the corresponding tenant networks.

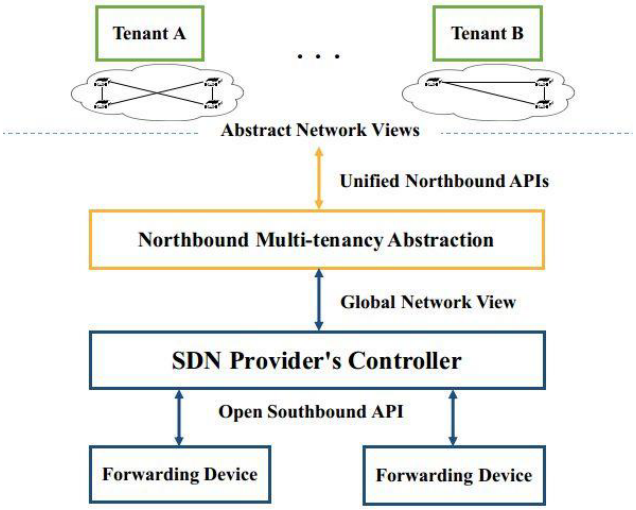

Figure 4. An Abstract Solution.

3.3 Architectural Components

The overall architecture shown in Figure 5, is divided into four layers : the *Controller Adapter* layer, the *Service* layer, the *Tenant Management* layer and the *Application Interaction* layer.

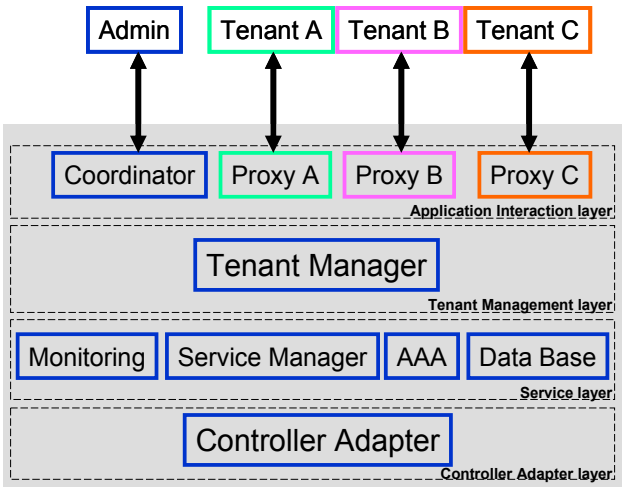

Figure 5. Overall Northbound Multi-tenancy Abstraction Architecture.

3.3.1 Application Interaction Layer

This layer consists of a set of tenant proxies (or domain proxies), and a management coordinator from admin. The tenant proxy is a concrete representation of tenant domain virtualizing the underlying resources. It is a functional component that represents the tenant's resources, capabilities and acts on behalf on the corresponding tenant in the server's environment. It contains details of the virtual networks that are exposed to the SDN applications and authentication information to identify itself when sending requests to the network, while isolating one tenant's services from another. Thus on top of the proxy, a tenant controller or application may consider itself as a direct controller of a suitably abstracted virtual network (VN). Different proxies may expose control over the network at different levels of abstraction (latitudes) or function sets (longitudes). The coordinator installs customer-specific resources and policies received from management, acting on behalf of the administrator in the server's environment. Multiple proxies may exist at the same time to support multi-tenancy, while there is only one logical management interface, thus only one coordinator. It is important to understand that code that executes in the colored proxy boxes inside the Gray domain are installed, maintained, and managed by Gray administration. This is the meaning of the phrase *logically extend of trusted domain* [1].

3.3.2 Tenant Management Layer

Tenant management layer is the core for virtualization and allocation of abstract resources to a specific tenant or application represented by proxy, which differs the meaning of virtualization in NFV, that is to abstract network function away from dedicated hardware. Tenant manager is the main module in this layer. For the upper layer, it is responsible for creating, maintaining and deleting the proxies according to management needs or policies. For the lower layer, it is responsible for coordinating different modules in the framework, handling the process that receives requests from tenant, validates requests against policies and resources assigned to the tenant, and translates the requests into lower layer language, while passing relevant information back up to corresponding tenants. Isolation is enforced in this layer by providing separate tenant proxies and strict data-control domain mapping.

3.3.3 Service Layer

Service layer comprises a set of function modules organized by the Tenant Manager to enable multi-tenancy, such as AAA, database, service manager, monitoring modules, etc.. It can also directly interacts with the lower layer to communicate with SDN controller for management purposes. Other features that could improve performance of multi-tenancy can be added in this layer. The access control functions are provided in this layer by AAA module.

3.3.4 Controller Adapter Layer

Controller adapter layer is responsible for interacting with underlying controllers through NBIs APIs, sending requests from upper layers down to the controller and passing messages from underlying network up to upper layers. This layer is important as the way it interacts with underlying controllers directly influences the performance of multi-tenant framework. For example, RESTful interfaces enable clear and well-structured interaction with SDN controller, while the XMPP-based solution for the NBI gives the possibility to enable BYOC concepts on top of the framework [12].

3.4 A PoC Implementation and Evaluation

A prototype based on OpenDaylight [13] controller and Mininet has been developed to evaluate our approach by focusing on the four key factors defined by SDMTN. OpenDaylight controller provides a virtual network function which is developed by VTN (Virtual Tenant Network) sub-project. VTN feature provides abstract view of tenant data domain but all these virtual networks can only be managed by one administrator with only one administrative API. In this case, tenants are not able to access and manage their own virtual network by communicating with OpenDaylight controller. It perfectly fits the assumption of our framework : the controller provides data domain isolation with no management function outsourced to any tenants. We use Apache Shiro Java framework [14] to provide Authentication and Authorization service function, with adoption of MySQL to store security data for access control and mapping data for control isolation. The architecture of our prototype is shown below.

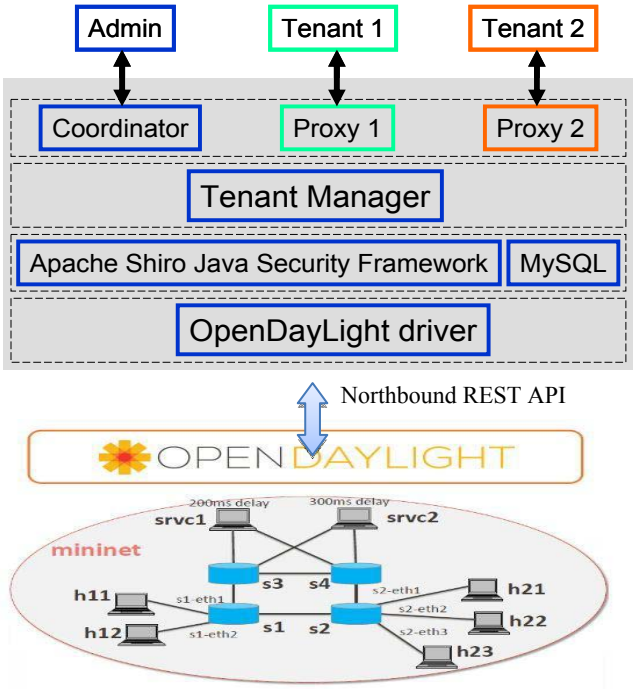

Figure 6. Prototype Architecture.

As the security and mapping data is stored in MySQL, all the requests received from the tenants are validated and then mapped by executing SQL queries. The database maintains four tables: *user* table with user authentication information, *domain_user* table maps all the tenants and their end users to their corresponding domains, *user_role* table maintains the roles that are allocated to each user, and finally *permission_domain_role* maintains the information about which role of which domain can have what permissions.

We evaluate the prototype by a set of operations to enable service chaining following different use cases. It appears that our prototype fulfills SDMTN concepts and requirements. It natively provides isolation with efficient control isolation enabled by SQL based mapping and traffic isolation provided by OpenDaylight controller. In addition, it provides fine-grained access control by providing domain-differentiated, credential-based authentication and domain-differentiated, role-based and also resource/action-specific authorization. Moreover, the prototype provides tenant manageability. It allows tenants to manage its own virtual network and SDN provider to control and monitor their behaviors. Finally, based on SDN/NFV architecture enabled by OpenDaylight, it supports full resource sharing on both network forwarding elements and virtualized services.

4. CONCLUSION

In this paper, we explored the multi-tenancy solutions for SDN environment. We defined SDMTN as NFV-integrated SDN multi-tenancy, and identify four key requirements for enabling SDMTN. We further illustrate possible models and available techniques of SDMTN with examples of current existing solutions for each model. Based on analysis and comparison, we argue that northbound multi-tenant model is a better approach for SDN multi-tenancy despite of the immaturity of NBIs. We thus further explore to design an element northbound multi-tenant framework following SDMTN northbound multi-tenancy model. This new framework consists of four layers which from up to down are respectively Application Interaction Layer, Tenant Management Layer, Tenant Service Layer and at last Controller Adapter Layer. In order to validate and evaluate the framework, we designed and deployed a prototype running on top of OpenDaylight controller and Mininet. The prototype finally turns out to be in line with SDMTN concepts and requirements, which provides isolation, access control, resource sharing as well as tenant manageability. On the other hand, there are also inevitably limitations in some aspects. The framework is designed as PoC without much consideration in terms of scalability, efficiency, etc., which need to be considered more thoroughly in our future works.

5. REFERENCES

[1] SDN Architecture, Issue 1.1 2016, ONF TR-521, https://www.opennetworking.org

[2] Kreutz, D., Ramos, F.M.V., Verıssimo, P.E., Rothenberg, C.E., Azodolmolky, S., and Uhlig, S., 2015. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, Vol. 103, N° 1, January 2015, pp.14-76.

[3] OpenFlow Switch Specification, Version 1.5.1, 2015, ONF TS-025, https://www.opennetworking.org

[4] Manthena, M.P.V., van Adrichem, N.L.M., van den Broek, C., and Kuipers, F., 2015. An SDN-based architecture for Network-as-a-Service. In *Proc. of the 1st IEEE Conference on Network Softwarization (NetSoft)*, (London, 2015).

[5] Shin, Y.Y., Kang, S.H., Kwak, J.Y., Lee, B.Y., and Yang, S.H., 2014. The Study on Configuration of Multi-Tenant Networks in SDN Controller. In *Proc. of the 16th Int. Conf. on Advanced Communication Technology* (Korea, 2014).

[6] Drutskoy, D., Keller, E., and Rexford, J., 2013. Scalable Network Virtualization in Software-Defined Networks.*IEEE Internet Computing*. 17, 2 (March-April 2013), 20-27.

[7] Sherwood, R., Gibb, G., Yap, K.K., Appenzeller, G., Casado, M., McKeown, N., and Parulkar, G., 2009. *FlowVisor: A Network Virtualization Layer*. TR OPENFLOW-TR-2009-1. https://www.opennetworking.org

[8] Real Time Media NBI REST Specification, Version 1.0 March 2015 TR-517, https://www.opennetworking.org

[9] Hernandez-Valencia, E., Izzo, S., and Polonsky, B., 2015. How Will NFV/SDN Transform Service Provider OPEX? *IEEE Network*, May/June 2015.

[10] Sahhaf, S., Tavernier, W., Colle, D., and Pickavet, M., 2015. Network service chaining with efficient network function mapping based on service decompositions. In *Proc. of the 1st IEEE Conf. on Net. Soft. (NetSoft)*, (London, 2015).

[11] Ranjbar, A. 2015. *Domain Isolation in a Multi-Tenant Software-Defined Network*. Master Thesis. Aalto University, School of Electrical Engineering.

[12] Aflatoonian, A., Bouabdallah, A., Guillouard, K., Catros, V., and Bonnin, J.M., 2015. BYOC: Bring Your Own Control a new concept to monetize SDN's openness. In *Proc. of the 1st IEEE Conf. on Net. Soft. (NetSoft)*, (London, 2015).

[13] OpenDayLight. https://www.opendaylight.org/

[14] Apache Shiro. http://shiro.apache.org