Survey on Network Virtualization Using OpenFlow: Taxonomy, Opportunities, and Open Issues

Ahmed Abdelaziz[1]*, Tan Fong Ang[1], Mehdi Sookhak[1], Suleman Khan[1] , Athanasios Vasilakos[2], Chee Sun Liew[1], Adnan Akhunzada[1]

[1] Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia

Email: [ahmedaziz, suleman]@siswa.um.edu.my, angtf@um.edu.my, am.sookhak@ieee.org, csliew@um.edu.my]

[2] Lulea University of Technology, Sweden, Department of Computer Science, Electrical and Space Communications 97187, Lulea,Sweden

[th.vasilakos@gmail.com]

* Corresponding author: Ahmed Abdelaziz

Received August 15, 2015; revised December 10, 2015; revised April 18, 2016; accepted August 23, 2016; published October 31, 2016

Abstract

The popularity of network virtualization has recently regained considerable momentum because of the emergence of OpenFlow technology. It is essentially decouples a data plane from a control plane and promotes hardware programmability. Subsequently, OpenFlow facilitates the implementation of network virtualization. This study aims to provide an overview of different approaches to create a virtual network using OpenFlow technology. The paper also presents the OpenFlow components to compare conventional network architecture with OpenFlow network architecture, particularly in terms of the virtualization. A thematic OpenFlow network virtualization taxonomy is devised to categorize network virtualization approaches. Several testbeds that support OpenFlow network virtualization are discussed with case studies to show the capabilities of OpenFlow virtualization. Moreover, the advantages of popular OpenFlow controllers that are designed to enhance network virtualization is compared and analyzed. Finally, we present key research challenges that mainly focus on security, scalability, reliability, isolation, and monitoring in the OpenFlow virtual environment. Numerous potential directions to tackle the problems related to OpenFlow network virtualization are likewise discussed

***Keywords**: Software Defined Networks, OpenFlow, Network Virtualization, Testbeds*

1. Introduction

Network virtualization (NV) creates a logical, virtual network, by decoupling network functions from the hardware that deliver them. Basically, all network functionality is separated from the underlying hardware and simulated as a "virtual instance". NV today is designed to create virtual networks within a virtualized infrastructure, which makes the network much more portable and scalable. The physical devices are simply responsible for the forwarding of packets, while the intelligence of the network is delivered by software. The decoupling of the control and forwarding planes delivers superior operational efficiencies and reduces costs, due to hardware independence. In general, a virtualized network can offer all the features and guarantees that a physical network could offer, only with greater agility and flexibility.

This unique feature of OpenFlow (OF) [1] can be used towards achieving network traffic isolation. By grouping together flows with different characteristics we create logical partitions of the common physical network infrastructure. If we map these groups of flows to different logical partitions and store this mapping in a centralized entity that has a complete image of the physical network, we will have created a flow-based virtual network abstraction on top of the physical overlay.

OpenFlow is a revolutionary idea in computer networking. This new technology has become popular and has received a significant amount of attention from the academia and related industry. OpenFlow is a novel networking concept, in which the forwarding hardware is decoupled from the control plane. This separation provides a more flexible, programmable, vendor-agnostic, cost-effective, and innovative network architecture. Moreover, network virtualizations (NVs) [2] have become a well-known technology that allows for the virtualization of OpenFlow-based network infrastructure with optimization, performance isolation, and minimal cost of hardware resources. OpenFlow NVs offer businesses a more comprehensive means to create scalable and manageable resources that utilize hardware.

Despite the significance of NVs, the current architecture of NVs is far from being ideal [3]. Conventionally, NVs employed IEEE 802.1Q [4]in virtual local area network (VLAN) tagging for virtualizing networks, wherein only 4,096 VLANs are attached. Therefore, NVs give rise to the scalability problem [5]. To overcome this limitation, IEEE introduced 802.1ad [6] to double the number of OpenFlow VLANs (4,096 ×4,096) by extending the tag attached. Several researchers [7];[8, 9] believe that this technology provides isolation, high security, and effective broadcast domains. However, VLAN only handles a set of MAC or IP addresses that point to one virtual network. For example, an end point in VLAN that belongs to only one virtual network shows that the approach is not flexible. In addition, VLAN cannot assign a certain MAC address to more than one virtual network. Layer 3 network virtualization in VLAN is also subject to subnet network. Therefore, adopting a single technology, such as VLAN tagging or MPLS labelling [10] cannot address all the limitations of current network virtualization such as . The aforementioned current technologies generate logical partitions from the physical network without providing a complete framework to perform network virtualization. OpenFlow NVs overcome most of these limitations and provide a framework that enables end users to fully manage and control their virtual network. However, several new research challenges associated with OpenFlow NV, such as isolation, monitoring, and security, have emerged. In this study, a comprehensive survey on OpenFlow NVs is conducted.

The concept of multiple coexisting networks is not new for many years, network equipment has supported the creation of virtual networks in the form of VLANs[11] and virtual private networks (VPNs) [12]. Therefore, many forms of NV have been presented, including the overlay network, virtual sharing network. Virtual Telnet Networks (VTN) [13]was proposed recently by OpenDaylight. VTN is a software defined network (SDN) controller that allows users to create and manage their virtual network dynamically. This technology offers opportunities to reduce operating expenses and easily deploy and configure multiple virtual networks for end users.

The idea of separating the control framework from underlying switches to create NVs dates back to 1998 when Merwe et al.[14]proposed the Tempest framework. The Tempest framework involves slicing the resources of switches among controllers; this partitioning or slicing is called *switchlet.* It is aims to provide a multi-virtual switch for each controller or group of controllers to have individual virtual networks.

The second attempt at NV was in a virtual network infrastructure (VINI)[15].VINI allows researchers to deploy and evaluate research ideas with an actual routing software, traffic loads, and network events. Researchers employ *Internetin a Slice* (ILAS) to conduct experiments; this network architecture consists of five components: a forwarding engine, a control plane, overlay ingress, an overlay egress, and distributed machines. ILAS utilizes a Click modular software router [16] as its virtual data plane and the extensible open router platform (XORP) routing protocol [17] as the control plane. The Click modular provides the illusion of point-to-point links to other virtual nodes and enables the virtual nodes to forward data packets. XORP manages numerous routing protocols, such as the border gateway protocol (BGP), open shortest path first (OSPF), routing information protocol (RIP), protocol independent multicast-sparse mode, internet group management protocol, and multicast listener discovery. Separating control and data planes in this manner means that XORP can run in a different slice or even on a different node from that of Click.

In 2007,Feamster et al. [18]introduced Cabo, which aims to divide ISPs into two distinct entities: infrastructure and service providers. An infrastructure provider owns the network equipment (*e.g.*, routers and links) that form an infrastructure network. A service provider establishes agreements with one or more infrastructure providers for access to a share of these router and link resources. Cabo facilitates the sharing of physical resources by subdividing a physical node (*i.e.*, router) or link into many virtual nodes and virtual links. A virtual node controls a subset of the underlying node resources and guarantees isolation from other virtual nodes running on the same machine. Similarly, a virtual link is formed from a path through the infrastructure network and includes a portion of the resources along the path. Cabo can guarantee bandwidth or delay properties on these links by using schedulers that arbitrate access to shared resources, such as CPU, memory, and bandwidth.

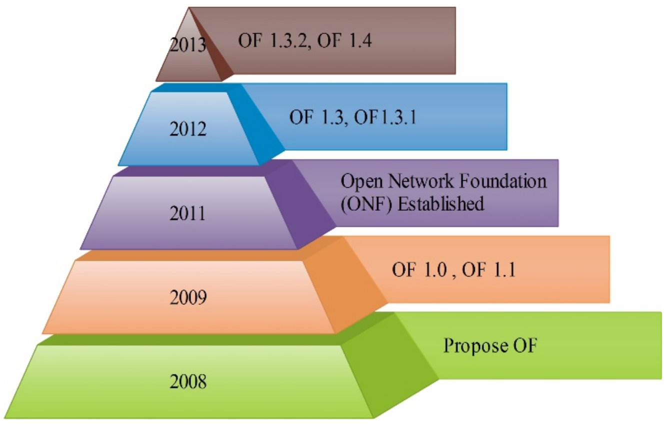

Fig. 1. OpenFlow specification history

OpenFlow was introduced recently as a new technology to facilitate NV. **Fig. 1** shows the development steps of OpenFlow. OpenFlow was proposed by Nick McKeown [19] in April 2008. Two OpenFlow specifications (1.0 and 1.1) were then introduced in 2009. In March 2011, several important organizations, such as Google, Microsoft, and Cisco, established Open Networking Foundation (ONF), which is responsible for issuing OpenFlow specifications. In 2012 and2013,many OpenFlow versions were released. These versions have new features, such as multi-flow, group, and meter tables. The latest OpenFlow specification, OF 1.4, was released in October 2013. This recent version has new additional features, such as bundles and synchronized flow table. The **key contributions** of this paper are highlighted below:

- A critical discussion on OpenFlow NV compared with traditional NV
- Devising a thematic taxonomy of OF network virtualization based on three different approaches
- Compared and analyzed popular testbed that used OF to provide virtual experimental environment
- We present key research challenges and future research directions for OF network virtualization.

This paper is organized as follows. Section 2 presents the OpenFlow components and the relationship between those components. Section 3 provides a definition of the NV layer in different architectures. Our taxonomy is discussed in detail in Section 4. Section 5 presents numerous Testbeds that provide a virtual network for OF experimental environments including case studies that show capabilities of OF virtualization. In section 6, the advantages of the recent OF proxy controller that is developed to enhance OF virtualization is discussed. Key research challenges and future research directions are presented in Section 7. Section 8 presents the conclusion

2. OpenFlow Components

The OpenFlow architecture [20] consists of three main components: OpenFlow controller, which is called the *control layer*; OpenFlow device (switch), which refers to the *data layer;* and OpenFlow protocol [19], which is actually the transportation layer where communication between a switch and a controller occurs. **Fig. 2** shows the components of

OpenFlow architecture. The main feature of the OpenFlow architecture is that the control and data planes are decoupled and abstracted from each other. The northbound interface [21] above the OpenFlow controller allows users to develop their own applications at the *application layer*, and the southbound interface provides standard API to facilitate the communication between the OpenFlow controller and the OpenFlow switch.

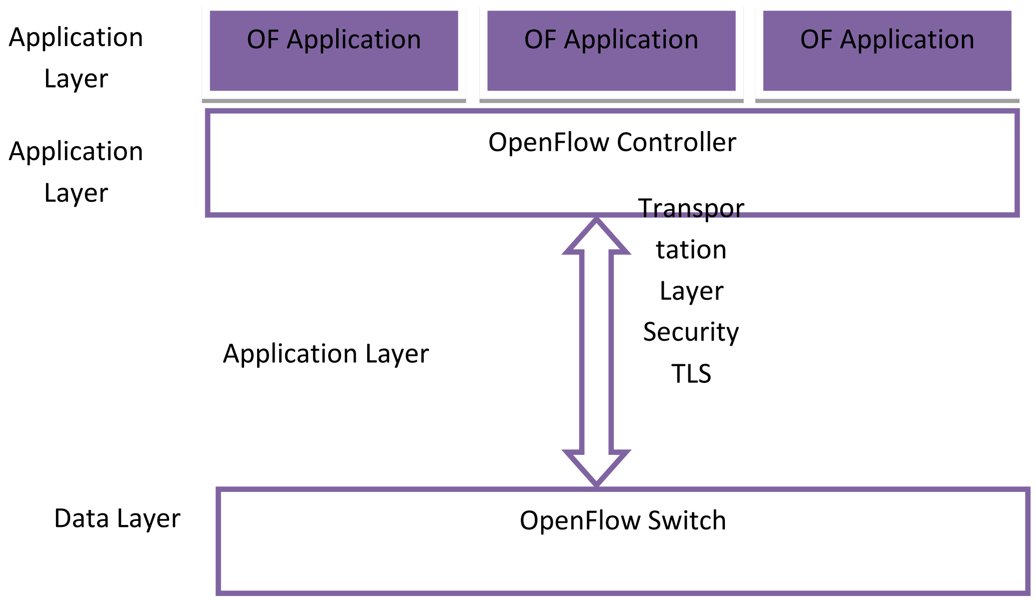

Fig. 2. OpenFlow architecture

2.1 OpenFlow switches

An OpenFlow switch is responsible for capturing, manipulating, and matching packets against flow table entries. The main function of OpenFlow switch is processing the transit traffic based on controller police which decides what to do with packets arriving on an ingress interface. It is manage a number of flow tables and each flow table contains a set of flow entries with their associated actions. OpenFlow switches can have one or multiple flow tables and a different group table that sometimes refers to an OpenFlow pipeline [22],in which a packet interacts with these flow tables. OpenFlow switches are divided into two types: pure (OpenFlow only) and hybrid (OpenFlow enabled)[23]. Pure OpenFlow switches have no legacy features or on-board control. These switches completely rely on the controller to forward decisions. Hybrid switches support OpenFlow as well as traditional operation and protocols. **Table 1** presents a flow table composed of a header field, counter, instruction, timeout, propriety, and cooks.

Table 1. Flow Table

Header Fields – Match Field													Counter	Instruction	Priority	Timeout	Cookes
L1		L2						L3		L4							
Ingress Port	Metadata	Ethernet Source	Ethernet Destination	Ethernet type	VLAN ID	MPLS label	MPLS Traffic Class	IP V4/6 source	IP V4/6 destination	TCP/UDP/SCTP source Port	TCP/UDP/SCTP destination Port	PBB UCA					
*	*	*	*	*	*	*	*	192.168.1.2	*	*	*	*	Flow Enter1				
*	*	*	*	*	*	*	*	*	*	120	*	*	Flow Enter2				
*	*	*	*	*	20	*	*	*	*	*	*	*	Flow Enter3				

A match or header field consists of fields that cover layers1 to 4. A switch works at layer 2 and uses Ethernet source, Ethernet destination, Ethernet type, MPLS, and VLAN tag. The routers search for IPv4, IPv6, and port number to process packages. A firewall uses four layers, namely, TCP, UDP, PBB, and ICMP, to build rules. These differences in layer options provide increased flexibility and diversify the range of rules in the flow table of an OpenFlow-enabled switch. These values of match fields are specified by either a wildcard, which is utilized when the match value is unimportant or exact, or a value that is exactly defined. **Table 2** presents a comparison of the header fields in the OpenFlow switch specifications.

The "counter" field provides statistics for each matched flow entry. This field includes various parameters, such as received packets, received bytes, duration, transmitted packets, and transmit/receive errors [24]. Counters can be maintained per flow entry, port, queue, group, group bucket, meter, and meter band. Handling all counter parameters is not compulsory for a switch; a switch handles only those parameters that are marked (required). A switch is considered a useful tool to troubleshoot and monitor networks.

Each flow entry is associated with a set of instructions or actions that change a package. When an incoming package matches the rule in the flow entry, an action is required. The action might be forwarding a package to a specified port or dropping the package. OpenFlow involves two types of actions: required and optional[25]. A required action must be supported in switches; whereas optional action is set based on the network requirements and could be a query by an OpenFlow controller. For more details, one may refer to[26].

Table 2. Comparison of the header fields in OpenFlow specifications

Header Field	**Description**	**OpenFlow Specification**				
		1.0	1.1	1.2	1.3	1.4
Ingress port	Physical or logical port	✓	✓	✓	✓	✓
Ethernet	Ethernet source and distinct MAC address	✓	✓	✓	✓	✓
Ether type	Ethernet frame type	✓	✓	✓	✓	✓
VLAN id	Input VLAN ID (0x8100)	✓	✓	✓	✓	✓

VLAN priority	Input VLAN priority (0x8100)	✓	✓	✓	✓	✓
IPv4 Src or dst	IPv4 source and destination address	✓	✓	✓	✓	✓
IPv6 src and dst	IPv6 source and destination address	X	X	✓	✓	✓
IPv4 proto/ARP	Apply to IP, IP over Ethernet, RIP package	✓	✓	✓	✓	✓
IPv4 Type OpenFlow Service	Ever package with Ethernet type 0x0800	✓	✓	✓	✓	✓
Transport src/dst port	TCP,UDP, and ICMP source or destination port	✓	✓	✓	✓	✓
Metadata	Data passed between tables (pipeline process)	X	✓	✓	✓	✓
MPLS label	MPLS label	X	✓	✓	✓	✓
MPLS class	MPLS traffic class	X	✓	✓	✓	✓
PBB service	Service instance tag	X	X	X	✓	✓

2.2 OpenFlow channel

The main function of OpenFlow channel is provide secure communication between the controller and switch, where a set of OpenFlow-defined messages can be exchanged. This channel is the interface that connects the switch to the controller. The default TCP port of a controller is 6633 which was revised to 6653 in OpenFlow switch specification 1.4 [27].

During the first communication between the controller and switch, authentication must be performed by exchanging user-configurable certificates. If the connection between the switch and controller is disrupted by the echo request timeout or TLS session timeout, the switch must endeavor to contact a backup controller. If the switch fails after a few attempts, the switch must proceed to emergency mode and immediately reset the current TCP connection. The OpenFlow controller exchanges messages with the switch via a security channel. These messages are divided into three types [28]:controller–to–switch messages that may or may not receive a reply from the switch, symmetric messages that are sent from the switch or controller without solicitation, and symmetric messages that indicate the status of the switch. These messages are shown in **Table 3.**

Table 3. Messages between the switch and controller via the OpenFlow channel

Type	**Message**	**Description**
Controller-to-switch	Features	*Controller*: sends when a TLS session is established
		Switch: must reply with a features reply
	Configuration	*Controller*: sends query configuration parameters
		Switch: responds to a query from the controller
	Modify-State	*Controller*: used to manage the state of switches
	Read-State	*Switch*: statistics parameters sent to the controller
	Packet-out	*Controller*: instructs the switch to send packets to an OpenFlow specified port
	Barrier	*Controller*: Barrier request/reply messages are used to ensure that messages have been met or to receive messages
Symmetric	Hello	*Controller*: sends when a connection is setup *Switch*: sends when a connection is setup
	Echo	*Controller*: Uses echo request/reply for the liveliness of OpenFlow controller-switch connection and indicates the latency and bandwidth

		Switch: request/reply to/from the controller
	Vendor	These messages provide a standard means for OpenFlow switches to obtain additional functionality within the OpenFlow message type space for future revisions of OpenFlow
Asynchronous	Packet-in	*Switch*: sends all packets that do not match the controller
	Flow-Removal	*Controller*: removes flow table entries by using DELETE or DELETE_STRICT message commands
	Port-status	The switch is expected to send port-status messages to the controller as the port configuration state changes
	Error	The switch is able to notify the controller of the OpenFlow problems by using error messages

2.3 OpenFlow controller

The controller is the main device responsible for managing, controlling, and manipulating flow tables in the switch. OpenFlow controller functions as a network operating system that views a comprehensive network topology and manage OpenFlow switch via secure communication channel. In SDN architecture, programmability is a key feature that enables companies and network carriers to rapidly change business requirements [26]. Hence, the OpenFlow controller provides two interfaces: a southbound interface (e.g., OpenFlow) that allows switches to communicate with the controller and a northbound interface that presents a programmable API to network control and high-level policy applications and services. Normally, the OpenFlow Controller runs on a computer with different control configurations depending on the following: flow type (flow routing, aggregated),behavior type (reactive, proactive) [29], and location. **Table 4** shows OpenFlow controllers and their main features.

Table 4. Well-known OpenFlow controllers and their main features

Controllers	**Language**	**Features**
NOX [30]	C++ - Python	Fast support, asynchronous IO
POX[31]	Python	Performs well compared with NOX applications written in Python (especially when operating under PyPy)
Beacon[32]	Java	Cross-platform, modular, Java-based controller that supports both event-based and threaded operations
Floodlight [33]	Java	Can manage both OpenFlow and non-OpenFlow networks
Maestro[34]	Java	Provides *view* abstraction to group related network states into a subset
Trema[35]	Ruby/C	A full-stack programming framework that allows users to develop and test OpenFlow controllers on a laptop
OpenDayLight[36]	Java	Employs OSGi framework and provides REST API
Ryu[37]	Python	Can be integrated with OpenStack, builds a virtual network without using VLAN

Special purpose controllers		
FlowVisor[38]	Java	Utilized to add a level OpenFlow network virtualization to OpenFlow networks
RouteFlow[39]	C++	Provides virtualized IP routing over OpenFlow hardware
ONOS [40]	Java	Builds service provider networks with good performance, scale-out design, and high availability

3. Network Virtualization Layer

Conventional network architecture supports single control plane and single data plane, as shown in **Fig. 3**. For example, in the router, the control plane is responsible for running network controls, such as routing algorithms (e.g., RIP, OSPF, and BGP).At the same time, the router manages network control protocols (e.g., ICMP), in which the forwarding tables and hardware data paths are implemented.

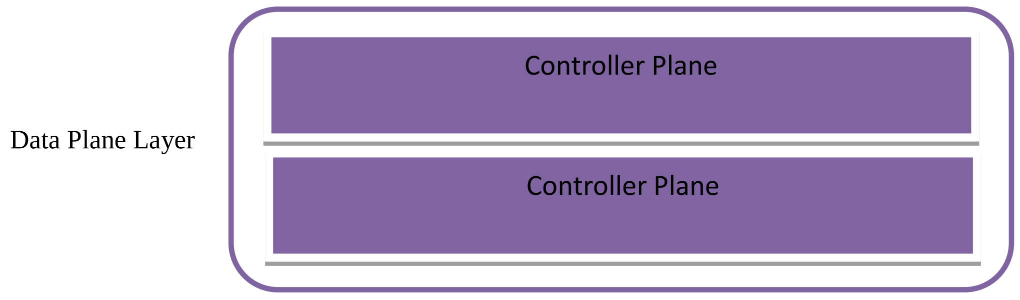

Fig. 3. Conventional network architecture

To virtualize the network, a virtualization layer is placed at a particular level in the network element architecture to allow the coexistence of multiple virtual network elements over a single physical network element. Assuming that this layer is placed between the control plane and the data plane and the control plane is virtualized the data plane is shared by all virtual networks. For example, in VLAN, switches are required to manage existence VLANs that configured based on Port Member, MAC address or IP address. This approach and other approaches used in confidential network such as MPLS, are limited. In SDN, the data plane is decoupled from the control plane. This separation provides a more flexible network virtualization with strong isolation between VNs or slices. Each virtual control plane in a slice has access to only a part of the data plane and cannot interfere with the other parts. **Fig. 4** illustrates virtualization resource abstraction in SDN that allows for the division of a resource into several slices. Similar to the actual resource interface, this abstraction creates a software layer that yields a virtual sliced interface. The virtualization layer decouples the real resource and the above layer and creates several virtual slices over the same resource. Hence, the traffic between the slices is isolated.

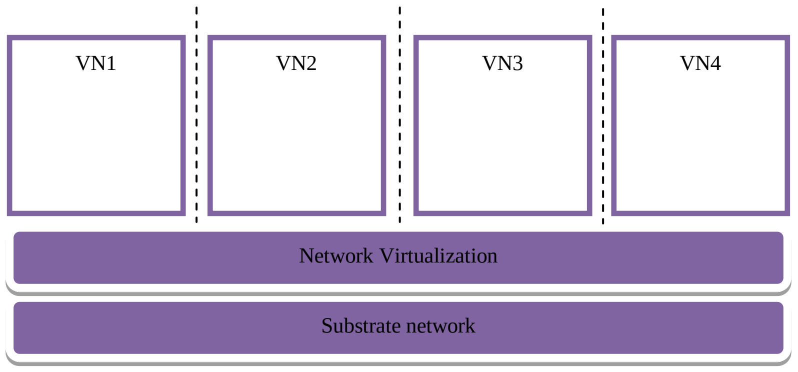

Fig. 4. Virtual slices in a network

4. OpenFlow Network Virtualization

The developed taxonomy aims to provide an overview of the approaches and methods offered by OpenFlow to implement network virtualization. **Fig. 5** shows the elements that comprise our OpenFlow virtualization taxonomy. The main elements and their sub-elements are described in the following sections. The NV-based OF taxonomy consists of three main categories, namely, proxy virtualization, layer 2 virtualization, and programming virtualization. Proxy virtualization basically refers to a transparent proxy virtualization layer, which is located between the controller and the switch, to create slices between virtual networks. Sherwood, R. et al. [38] proposed FlowVisor, a special type of controller placed between a control plane and a data plane to create slices as a set of flows running on a topology of switches to ensure the operations of the guest controller.

One of the limitations of FlowVisor in terms of virtual topology is that it is a sub-set of physical topology. Salvadori [41]attempted to overcome this limitation by allowing users to customize their topology. Moreover, FlowVisor does not authorize access to slices, which is considered a significant feature in Testbeds. The NITOS Testbed employs the NITOS system to provide authorized access to a user to ensure that the user can only access a particular slice.

OpenFlow technology also allows the use of layer 2 prefixes-based virtualizations, such as VLAN ID, MAC, and MPLS, to implement NV without using tunnels. Many researchers [42];[43, 44] proposed the use of layer 2 to implement NV; some of them implemented this approach with FlowVisor, whereas others used a module in the controller to handle packages based on layer 2 prefixes(discussed in detail in Section 5.2).

Many programming languages for OpenFlow, such as Frenetic, have emerged recently. These languages provide an alternative means to create isolated network slices. Gutz, S. et al.[45]established a semantic slice approach that ensures the processing of packets on a slice (separated from one another). Chen, Y. et al.[46]introduced a virtual network management component that allows the creation and management of VNs in OpenFlow based on a programming module in a controller without using a visualization layer. We discuss programming virtualization is Section 3.3.

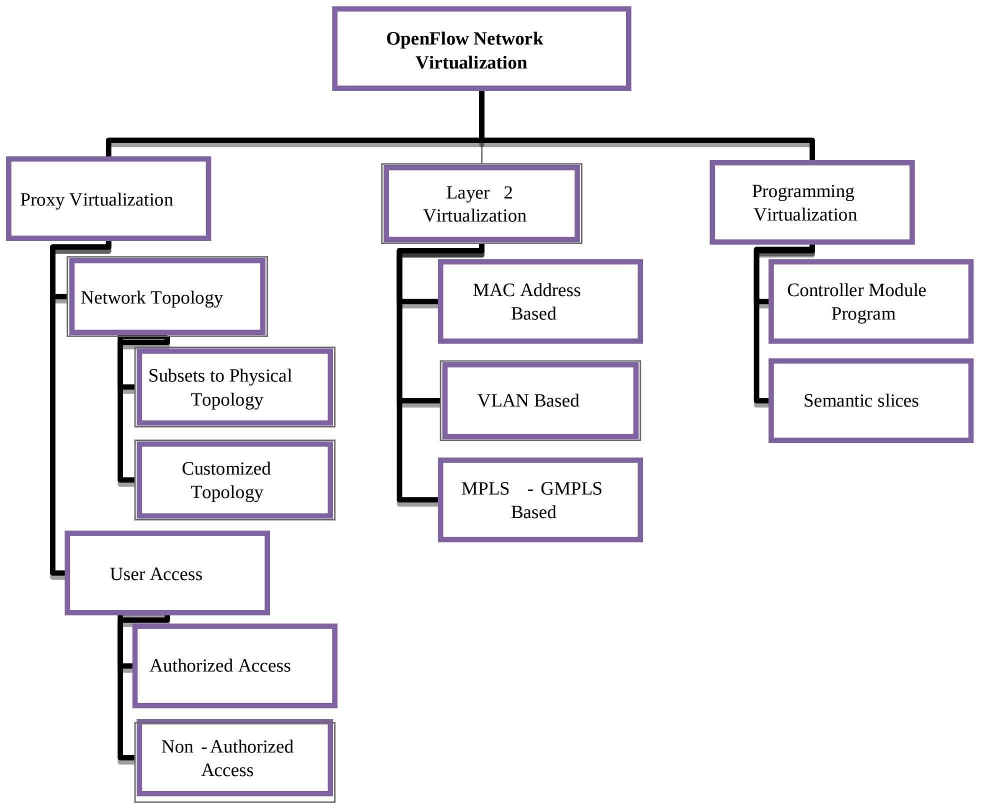

Fig. 5. Network virtualization-based OpenFlow taxonomy

Table 5 shows the reviewed studies classified based on our taxonomy. The studies are also divided based on the SDN layer; the data layer refers to switches, and the controller refers to an SDN controller in the control layer. The last column shows whether the study provides a solution or merely an enhancement.

Table 5. Classifications of the network virtualization studies based on OpenFlow

Publication	**Taxonomy – Classification**			**SDN layer**			**Virtualization**	
	Proxy Virtualization	Programming Virtualization	Layer 2 Virtualization	Data layer	Ctrl layer	App layer	Enhancement	Solution
[38]	✓			✓				✓
[41]	✓			✓			✓	
[47]	✓			✓			✓	
[48]	✓			✓			✓	
[49]	✓			✓			✓	
[50]	✓			✓			✓	
[42]		✓			✓			
[51]		✓			✓			

[43]		✓			✓			
[45]			✓			✓		✓
[46]			✓			✓		✓
[52]	✓			✓				✓

4.1 Proxy virtualization

The principle of the SDN architecture allows the separation of the data plane from the control plane and for multiple controllers to be connected with a single switch. A special type of controller called FlowVisor works as a transparent proxy between two controllers. FlowVisor implements NV by placing the FlowVisor between a guest controller and a physical network. Initially, the aim of the FlowVisor controller is to allow researchers to perform experiments without VNs interfering with one another. **Fig 7** shows the FlowVisor architecture. In this section, we discuss several studies that applied FlowVisor as a proxy for NV.

Sherwood, R. et al.[38]proposed a novel approach to the NV layer or FlowVisor and referred to switch virtualization, in which the same hardware forwarding plane can be shared among multiple logical networks; each virtual network adopts a distinct forwarding logic. FlowVisor divides a network into slices that host several guest controllers. Each slice is controlled by the private OpenFlow controller, which controls and observes its own slice. FlowVisor insulates one slice from another, including the data path traffic that belongs to the slice, and the control of the slice. Each slice has a particular hypervisor that includes a flowspace and a guest controller. This flow space uses a package header field to define a set of characterized flows, and the guest controller is assigned by different IDs and separate queuing buffers. FlowVisor provides conflict resolution among slices by tracking the flows of each slice. **Fig. 6** illustrates FlowVisor that prevents conflict and creates complete isolation of slices. For example, FlowVisor receives a flow table from NV1 and modifies it according to the guest controller's configuration. Several studies have been performed based on this approach, where most of the studies attempted to enhance this approach and overcome several limitations of FlowVisor.

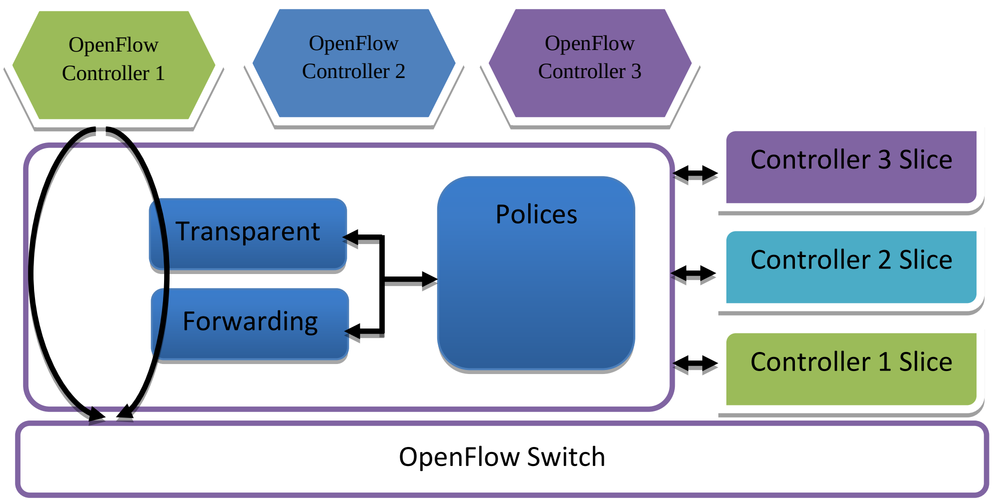

Fig. 6. FlowVisor mechanism

FlowVisor lacks several features, such as virtual topologies and sharing of flow space by two slices without interfering with each other, to enable a full implementation of NV. Salvadori, E. et al. [41]proposed Advanced Flow Visor (ADVisor) to overcome the limitations of FlowVisor. Unlike FlowVisor, ADVisor functions as a proxy virtualization and responds directly to the OpenFlow network to enable the definition of logical topologies that are isolated from the underlying physical network.

Azodolmolky, S. et al. [47]proposed Optical FlowVisor , which employs a PLI-aware virtual optical network composer and a virtual network constructor. Virtualization of an optical switch is achieved by partitioning or aggregating physical optical switches. The associated virtual optical links are defined as the connections among virtual nodes. The granularity of virtualization links depends on the underlying technology (e.g., wavelength channels in a single fiber provided by WDM transmission), which can guarantee the isolation of virtual links while sharing the same physical infrastructure.

VeRTIGO, which is an OpenFlow software that defines a networking platform designed for NV, was proposed in a previous study [48]. VeRTIGO adds an extra feature to FlowVisor, where an infrastructure provider allows customers to obtain different views of the network through various OpenFlow controllers depending on the needs of customers. The customer can select from two options. The first option is full virtual network, where the customer has full control of the network. The topology should be fully customizable according to the needs of the customers. The second option is single (abstract) node, where the customer concentrates on routing policies while leaving the management of the underlying physical layer to the infrastructure provider. The infrastructure provider can differentiate its OpenFlow according to the service level required by customers (e.g., maximum latency or packet loss between two node ports).

A simple virtual topology performed through a web-based control framework [49] has also been introduced. This OpenFlow software allows the reservation of network resources (nodes, links, and bandwidth) and the management of virtual resources (virtual links and virtual ports) based on ADVisor. In ADVisor, virtual topologies are identified through a set of tuples included in the configuration files and by specifying each component of a virtual topology (virtual nodes, virtual links, and virtual ports). The flow space of each switch in the network is partitioned among the virtual topologies through the combinations of bits that involve only the OSI-layer 2 fields of the packet header, such as VLAN ID, MPLS labels, or IEEE 802.1ad-based multiple VLAN tagging.

Xingbinset al.[52] proposed virtualization based on double FlowVisors (VPDF) in a multiple domain production network. VPDF provides network abstraction for physical networks with a unified API. Xingbinargued that VPDF aids in building a virtual network application that can provide a high resource utilization rate and reduced blocking probability. **Fig. 7** shows the VPDF platform that consists of flow visors 1 and 2. FlowVisor 1 is placed between multiple guests OpenFlow controllers and OpenFlow enabled switches. The flow visor is responsible for creating strong isolation of network slices, managing rich extensible policies, dividing link bandwidth, and managing flow tables. FlowVisor 2 is located between the guest control and application layer, provides API for the application layer, and virtualizes NOS.

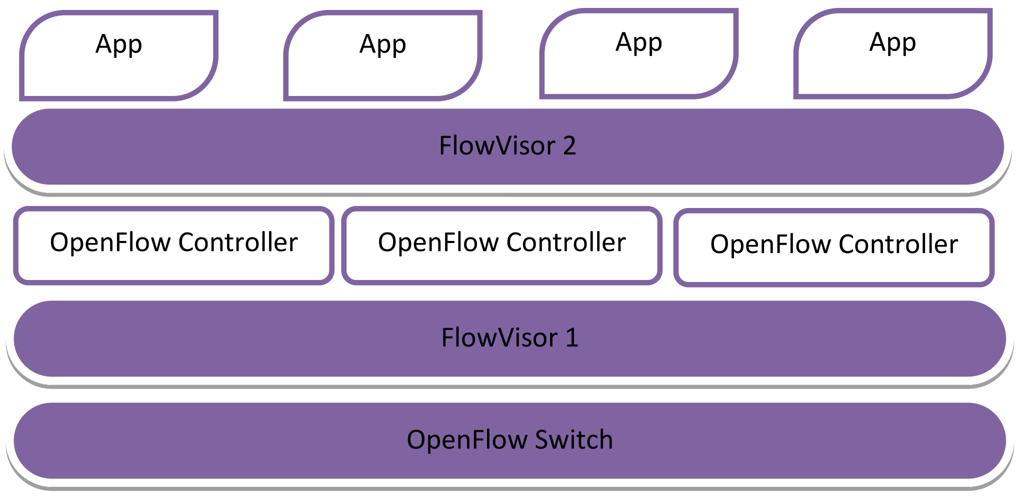

Fig. 7. VPDF virtualization platform

4.2 Layer 2 virtualization

Tsai and P.W. et al.[42]introduced a virtualized architecture to extend the controllability of VLAN tags through an implemented control module in the OpenFlow controller. Through this tag control in the control plane and a set of configurations in the flow table, the management of VLAN tags is addressed in the control plane and leaves the data plane processed tagged traffic transparent in the line rate.

Matias, J. et al. [51]proposed a virtualization framework for Cloud. The framework involves the use of the MAC addressing scheme to achieve NV and overcome scalability problems. The authors argued that to virtualize the physical network infrastructure, implementing a proper abstraction of virtual elements is required. The proposed approach defines three different types of virtual elements,namely, virtual hosts (vhost), virtual nodes (vnode), and virtual links (vlink). When a frame enters a vnode, the appropriate match field is utilized to search for the virtual table and determine the corresponding output port to which the frame would be sent.

Matias, J. et al. [43]proposed NV based on layer 2 prefixes using the layer 2 prefix-based NV (L2PNV) approach. This approch decouples different slices and allows users to have several virtual networks depending on the MAC source/destination. L2PNV allows for the use of VLAN, Q-in-Q, MAC-in-MAC, or MPLS transparently without using tunnels.

Several studies [44, 53] proposed integrated MPLS and GMPLS to achieve NV. Das et al. [44] proposed a unified architecture that combines OpenFlow with a packet and circuit network to speed up innovations and reduce Capex and Opex significantly. Sharafat et al. [54]introduced MPLS-TE and VPN services managed by an application layer of the NOX controller. NOX handles all MPLS features, whereas the OpenFlow switch manages push, swap, and pop actions. NOX modifies flow tables according to their respective switches subject to any changes required on the data plane.

4.3 Programming virtualization

Several high-level network languages, such as Frentic, Netcore, and Procera, have been developed based on comprehensive API. Given the limitations of the virtualization layer, such as isolation and flow entry conflict, many studies haveintroduced network programing languages and modules to achieve NV without a middle layer. In this section, we present OpenFlow NV using a network programming language.

Gutz and Story[45] introduced the semantic isolationof network slices by using a programming language while ensuring that the processing of packets on a slice is independent of all other slices. The authors introduced a slice that extends the network,with new logical switches that can be configured similar to an ordinary switch. The semantics of a slice can achieve two simple principles:a packet entering a slice and packet processing.

Existing NV methods in the OpenFlow network usually have limitations, such as introduction of a proxy, flow entry conflict, and lack of extensibility. Similarly, [46]presentedvirtual network management component (VNMC) torapidly create, configure, and manage virtualnetworks in the OpenFlow network. VNMC is implemented in a single controller without a middle layer. VNMC provides bandwidth guarantee, topology isolation, flow isolation, and control isolation. Moreover, it allows virtual networks to be extended easily because these networks are modular.

5. Testbeds Based on OpenFlow NV

This section presents a discussion of various types of Testbeds that employ OpenFlow technology to allow researchers to perform experiments by creating a virtual network while ensuring that VNs are isolated from one another. **Table 6**provides a summary of the Testbeds.

Table 6. OpenFlow Testbeds

Testbeds	**Projects**	**Tools, Platforms, and Framework**	**Granularity of Virtualization**	**Emphasis**	**Layer of Virtualization**	**Overlay \Cluster**	**Publication**
FIBRE	None	(***CMF***):control and monitoring framework ***Zenoss***: monitoring resources	Like - note	Wireless and optical communications	L2–L3	Cluster	[55]
FITS	FITS	Xen	Full	Security	L2–L3	Cluster	[56]
OFELIA	European FP7	OCF	Full	Future Internet Testbed facility	Layer 2	Cluster	[57]
TWAREN	TWAREN	***VPLS service VM manager*** module	Full	Future Internet Testbed facility	Layer 2	Cluster	[58]
FiRST @PC	FiRST	NetFPGA platforms ENVI	Full	QoS	Layer 3	Overlay	[59]
NITOS	OpenLab	NITOS scheduler	Full	Wireless	Layer 1 Switch port	Overlay	[50]
EmPO	CREAT	Energino,	Full	WiFi -	L2	Overlay	[60]

WER	E-NET	“chassis manager” allowing the Testbed manager to power on/off APs remotely		SDN&NFV			
DOT	DOT	Central management module	Full	SDN	L2–L4	Cluster	[61]
GENI	GENI	***Flack*** graphical resource discovery ***omni*** command-line interface **Myslice**resource discovery tool	Note/Link	Future Internet Testbed facility	L2	Cluster	[62]
ECODANE	ECODANE	NetFPGA-D-ITG	Note/Link	Energy consumption	L2–L3	Overlay	[63]
JGN-X	JGN-X	-	Note/Link	SDN	L1– L2	Overlay	[64]

5.1 FIBRE

FIBRE (Future Internet Testbeds/experimentation between Brazil and Europe) is one of five projects that were approved in response to the 2010 Brazil-EU Coordinated Call in ICT, jointly funded by CNPq (the Brazilian Council for Scientific and Technological Development) and by the European Commission within its Seventh Framework Programme (FP7). FIBRE was launched in October 2011

The Testbed employs FlowVisor to provide virtualization of resources, such as nodes, network devices, and networks. In addition, FIBRE utilizes a control and monitoring framework (CMF). CMF allows the central management site to facilitate and control access to virtualized computing and network resources. CMF also provides support for the measurement of resource usage. One of the important components of the architecture of FIBRE is MySlice. MySlice is a software layer that allows federation abstraction and integration of different FIBRE Testbeds. FIBRE provides a web-based interface that enables users to interact with a large volume of results generated by each Testbed island.

5.2 FITS

Future Internet Testbed with Security (FITS) provides was launched in 2011. It provides an opportunity to increase cooperation among those universities aiming at managing and configuring virtual networks. It is provides two main approaches to virtualize a network: virtualization of an entire network element implemented using Xen technology (which has both control and data planes inside) and decoupling of the control plane and data plane through the use of OpenFlow to virtualize the data plane only. FITS virtualizes a data plane and offers two modes, namely, decentralized or centralized. In the decentralized mode (**Fig. 8**), each virtual network slice has a set of virtual network elements that contain only the control plane, whereas in the centralized mode (**Fig. 9**), each virtual network slice has a control plane centralized in one node that configures all the data planes of the nodes in the slice. FITS has several features, such as flexible design, efficient packet forwarding, network migration, and security. To perform experiments in SDN, FITS provides OpenFlow slices to researchers based on FlowVisor slices. FITS defines a slice based on twelve-tuple match fields, which provide a range of virtualization layers (L2–L4). In addition, FITS applies isolation techniques other than FlowVisor to isolate the four main resources in the OpenFlow network. The four main resources are based on topology, bandwidth, memory, and processor usage.

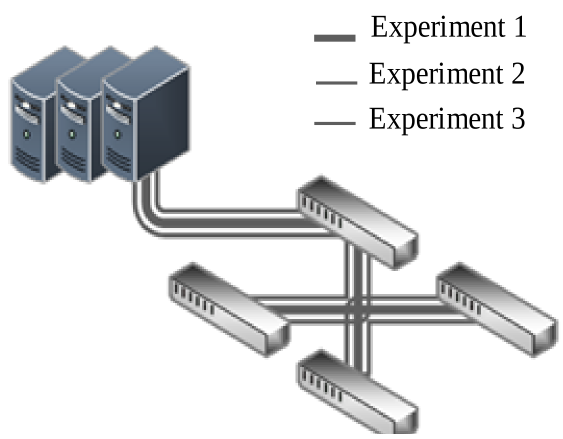

Fig. 8. Centralized approach: One Controller for each Experiment

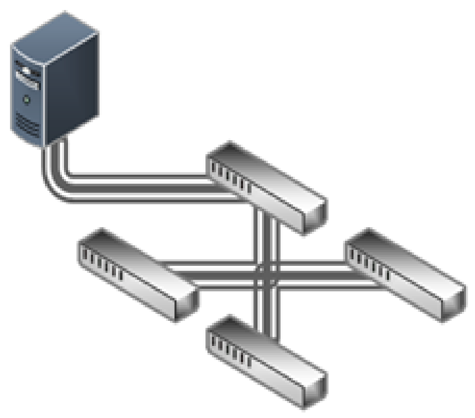

Fig. 9. Decentralized approach, where each controller has its own data plane

5.3 OFELIA

The FIBRE Project ended in October 2014 (co-funded by the Brazilian government and the EC). In 2015 the Brazilian institutions took over FIBRE's legacy infrastructure to start offering the Testbed as a service. OFELIA offers OpenFlow experiment services for users; these services allow for the development and testing of new networking ideas based on layer 2. OFELIA contains different integrated elements to generate experiments, such as OpenFlow experimental networks that are built on OpenFlow-enabled Ethernet segment, and is connected to virtual machines. Control and management networks are used to setup and manage the infrastructure itself and other OFELIA internal services, such as DNS. In addition, the control network may also be connected to another island-local infrastructure. Optical equipment adoption in OpenFlow employs Optical FlowVisor. OFELIA enhances the API of the current flow visor and the OpenFlow aggregate manager.

5.4 TWAREN

Founded in 2009 supported by Taiwan National Science Council (NSC) and deployed to the TWAREN research network. In the TWAREN Testbed, each OpenFlow switch has its own controller that resides in different domains. The controller employs link layer discovery protocol (LLDP) messages among the switches to obtain topology information. However, the controller in different domains cannot exchange these messages. Therefore, LLDP complicates the management of the network. To overcome this problem, TWAREN modifies LLDP messages and several applications in NOX to retrieve links among inter-domains. The dropping policy is modified to store the received LLDP packets from different controller domains.

5.5 FiRST@PC

FiRST@PCwas established to facilitate future Internet experiments based on OpenFlow technology. FiRST@PC is employed by four universities in Korea (Chungnam National University, GIST, Postech, and Kyunghee University).FiRST@PC consists of OpenFlow controllers and OpenFlow switches integrated with NetFPGA platforms that provide QoS. Based on switch ports and VLAN IDs, the OpenFlow switches manage slices by using a FlowVisor controller placed between guest controllers and switches. The switches and controllers are unaware of the process of FlowVisor and can communicate with each other directly.

5.6 NITOS

The NITOS Testbed is managed by the University of Thessaly. NITOS combines the functionalities of Flow Visor with the NITOS Scheduler resource reservation tool. When users request to perform experiments, a slice is created in both NITOS resource reservation and FlowVisor and is connected directly to the NOX controller. Users can select their own controller and receive some node features from the NITOS Testbed. In the beginning of the experiment, the scheduler sends commands to the FlowVisor to insert a flow space for a specific port. At the end of the experiment, the scheduler instructs the FlowVisor to delete the flow space from that port so that the user associated to the slice would not have access to the switch ports anymore.

5.7 EmPOWER

EmPOWER is a unique and open toolkit for research and experimentation on SDN/NFV over wireless and mobile networks. Its flexible architecture and the high-level programming APIs allow for fast prototyping and validation of novel services and applications. It was established based on SDN framework for SDN and FlowVisor research and experimentation. This architecture consists of OpenFlow and an energy consumption monitoring and management toolkit. Currently, EmPOWER has 30 nodes and is employed by both undergraduate and graduate students at the University of Trento. The main components of EmPOWER are Single Master, multiple agents, network application, and Energino. The Single Master provides an entire view of the network infrastructure, and the multiple agents create a logical isolation for each switch port. Network application on top of the controller can facilitate Floodlight REST or any intermediate interpreter, such as Pyretic. Each network application provides efficient control of an isolated slice. Lastly, Energino is an Arduino add-on that manages and measures the energy consumption of nodes.

5.8 DOT

Distributed OpenFlow Testbed (DOT) was developed by a research group at Waterloo University in 2013 to emulate large SDN deployments by distributing the workload over a cluster of compute nodes. The main reason for this development is to overcome the limitations of Mininet, such as scalability and traffic measurement. DOT is capable of emulating a network in different physical machines and utilizes CPU time, bandwidth, and latency for all network elements (switches, hosts, and links). Moreover, DOT provides a central monitoring point. The DOT architecture comprises two components. The first is the central DOT manager used to allocate resources.DOT has two sub-modules, that is, provisioning module and astatistic's collection module. The second component is the DOT Node Manager that allocates the required resources and collects multiple computer statistics. However, DOT only manages a fixed number of physical machines and does not allow multiple users to run multiple emulations on the same physical infrastructure simultaneously.

5.9 GENI

GENI (Global Environment for Network Innovations) provides a virtual laboratory for networking and distributed systems research and education. GENI is managed by researchers in collaboration with Stanford. GENI is utilized to develop an OpenFlow control framework. It facilitates experiments that employ OpenFlow-controlled VLANs. Students can perform experiments within the campus network and can connect to external GENI SDN infrastructures. Using slices and flow space, GENI groups campus network resources. An aggregate manager provides an experimenter access to virtualized network resources. The flow spaces support layers 1 to 4 network virtualization, and each flow space is managed by a controller that may run on a resource within the slice or elsewhere. Implementing OpenFlow in a GENI-enabled campus provides several advantages. For example, OpenFlow allows experimental implementation and production of traffic within the same network. This approach provides benefits to both the experimenter and staff in campus. In terms of federating, OpenFlow can be employed easily and runs in multiple campuses and backbone resources.

5.10 ECODANE

ECODANE Testbed originally outcomes of the ECODANE project. It is combines real network devices with emulation for providing experiment environment of new concepts of energy-efficient data center. ECODANE is based on OpenFlow that consists of hardware devices and a virtual emulation test environment to improve the scalability, flexibility, and accuracy of the experimental environment. ECODANE provides an experimental environment in the energy-efficient data center. The ECODANE Testbed architecture has three main modules: the OpenFlow controller that is used to manage and control parts of the network, the data center network that manages physical and emulated network devices, and a traffic generator (a mathematical model used for real traffic traces measured in the data center based on Internet Traffic Generator (D-ITG). In the ECODANE Testbed, researchers can analyze different performance factors, such as energy proportionality, QoS, and complexity.

5.11JGN-X

JGN-X is sponsored by (NICT) National Information and Communications Technology in Japan. It is start in 2011 aim to advance the research and development environment for New Generation Network. It is mainly focuses on next-generation network technologies, such as SDN. JGN-X provides experiments at several layers (e.g., layer 3) and employsIPv4 and IPv6 with bandwidth capacity starting from 100 Mbps up to 10 Gbps. In layer 2, JGN-Xallows for experiments via Ethernet service and VLANs. JGN-X also provides layer 1 optical Testbed services.

5.12Case Studies

In this section, we introduced some case studies that are currently being deployed on various Testbeds that support NV based on OF. Four case studies have been detailed and dominated recent example of how researchers have used Testbed to facilitate their experiments. **Table 7** shows the four case report studies which explored the use of Testbeds.

Table 7. Case Studies

References	Case	Testbed
[65]	Video-on-Demand	OFELIA
[60]	Energy Programmable WiFi Networks	EmPOWER
[66]	Congestion Control Protocols	GENI
[67]	Cloud computing based collaborative manufacturing	GENI

Cloud computing based collaborative manufacturing: This experiment was carried out in Oklahoma State University, and The Ohio State University using GENI testbed. It aims to test advanced manufacturing problems without the need of expensive local resources. It helps users to collaborate with remote experts via virtual reality environments and other tools. Thus, researchers and experts will be able to collaborate globally in the virtual environments and can rapidly bring advanced products to marketplaces. The experiment example was in the domain of Micro Device Assembly (MDA). Users from different locations can access collaboration involving a simulation application developed by C++, propose and then modify the assembly plans interactively for given part designs. The experiment demonstrates the potential to build future Internet services that utilize access to high-speed networks and cloud infrastructures. Manufacturing and cloud architects that are provided by GENI Testbed can help the functionalities of products with less cost and user convenience.
Energy consumption in WiFi Networks: is second experiment that used EmPOWER testbed infrastructures. It aims to reduce the actual energy consumption of WiFi infrastructures. To achieve this goal, a real–time energy consumption monitoring with Odin, a software, to define networking WiFi networks is used. Authors?? argue that, in WiFi networks, the extent of energy savings is limited by the actual client distribution. EmPOWER deployed single Master and multiple OF controller. Single Master provides a global view of the entire network which includes clients, flows, and infrastructure while multiple agents controller represent a set of logically slices connected to different ports of a switch. Energino, an Arduino add–on, is used to measure the energy consumption of a wireless device and voltage sensor and a current sensor is used as measurement circuit. Two modes of operations are implemented: online mode and offline mode. In the Online mode, an (Access Point) AP and all its wireless interfaces are on. In the Offline mode, the entire AP is turned off and only the Energino is powered. The transitions between modes implemented based on finite state machine (FSM) that is configured for possible events in the Energy Manager module. Recently, EmPOWER added programmable wireless base stations (LTE) that provide heterogeneous scenarios test with programmable cellular networks.
Congestion Control Protocols (CCP): In traditional network, various CCP are proposed such as XCP, RCP. These protocols address the problem of performance issues in high-speed networks. Due to the lack of a platform that supports deep programming inside network and virtualization with independent slice, evaluation testing CCP is limited. In this experiment the Binary marking Congestion Control (BMCC) protocol evaluate GENI testbed. To achieve that OF enabled switch with NetFPGA are used, whereas OF switch provides dynamic forward traffic to NetFPGAs that used to scale the processing. The experimental

setup in the GENI testbed includes two clients and one server that connect two OF switches with two NetFPGA cards.
NetFPGA-1 implements the BMCC router functionality while NetFPGA-2 is connected to client. To perform measurement, a different scenario is implemented. Scenario 1 focuses on bottleneck utilization and packet loss rate, while scenario 2 focuses on bandwidth-sharing properties. The comparison between BMCC and TCP flow is achieved by measuring throughput, and response time.
Video-on-Demand: In this experiment, authors [65] proposed SDN OpenCache for video on demand service in cloud. The solution aims to decrease the distribution load from the VoD content provider to the end-user. In traditional network, proxy and Content Delivery Network (CDN) are used to push content to the edges of the Internet in order to provide higher throughput, less latency between the VoD server and the user. To measure efficacy of OpenCache, three experiments were conducted in OFELIA Testbed that is composed of a number of OpenFlow-capable hardware switches located in various different sites. Each site represents "island" located in different countries connected to each other to create a large federated experimentation environment. For example, in this case study the video client runs on VM located on Switzerland where an OF controller is deployed on OCC and OCN in two different sites. In the other two different OFELIA islands, Italy and at i2CAT in Spain VoD servers were deployed and the experiment was implanted in Belgium.

6. OpenFlow advantages

In our taxonomy that is discussed in section 3, we found that the main type of the OF NV is *Proxy Virtualization*. The first Proxy in OF is FlowVisor, special type of the controller placed between SDN controller and OF enables switch to create virtualization layer. Based on FlowVisor concept, some controllers are proposed to add more advantages. **Table 8** shows five popular OF proxy controllers that enhance OF network virtualization and address limitations of FlowVisor.

Table 8. advantages of OpenFlow proxy virtualization

Cases	OpenFlow Proxy Controller	Features
[68]	OpenVirteX	address virtualization and topology virtualization
[69]	FlowN	Fully tenant VNs with its own controller
[70]	HyperFlex	Virtualization of the control plane
[71]	DFVisor	reduces flow setup latency, and removes the single point of failure in the slice network
[72]	Autoslice	Enhance the scalability of VNs

6.1 OpenVirteX

OpenVirteX comes to overcome the problem of flow-space that is introduced by FlowVisor, and therefore, it builds on the design of FlowVisor. It has two advantages; address virtualization and topology virtualization with the facility to give each tenant its own header fields space. In the topology virtualization, tenant can customize its own arbitrary topology that does not have to be corresponded to the physical network. OVX stores the mapping of NVs in the edge switches in which it re-writes the virtual IP and MAC addresses. Such mapping allows flowspace to be provided for each NVs. On the other hand, OVX manipulates the Link Layer Discovery Protocol (LLDP) controller message and send it back to controller for creating the illusion topology. In addition, OVS provides address virtualization that gives each tenants the ability to assign IP address for their end hosts. To avoid over-lapping, IP address OVX generates globally unique tenant IDs for each tenant. Moreover, OVX allows each VN managed by its own controller which means that a tenant has its own ONS and applications that can program the virtual network switches. To create the "illusion" that every controller is the only OS running in the system, OVX intercepts control packets and multiplexes different control planes onto one.

6.2 FlowN

It is a container based on application virtualization for OpenFlow network. Each container incudes the tenant VNs with its own controller. It provides virtual network topologies to the tenants by entirely abstracts the physical. This abstraction grants that the virtual nodes can be transparently migrated on the physical network without accessing the physical network. Therefore, tenants are not concerned about the resource management actions of the nodes, and they see only their virtual topologies. For mapping the NV to provide virtual address spaces, FlowN uses additional database component. FlowN application virtualization can be deployed in multiple physical servers that host databases, each one includes a particular number of containers. To manage the resource isolation between containers, FlowN assigns one processing thread per container. During the setup of a system, FlowN parsers the topology via configuration file and builds the view of the network without admitting it to the physical network. FlowN provides complete NV system for each tenant at setup time and allows users to add nodes at run time without concerning about resources allocation which is managed by FlowN. During the setup of a system, FlowN parsers the topology via configuration file and builds the view of the network without admitting it to the physical network. The main drawback for FlowN is that it is an extension of the NOX controller, and therefore, users are limited to the capabilities of the NOX controller.

6.3 HyperFlex

Introduced as new OpenFlow proxy controller to tackle the problem of virtualization of the control plane, whereas the existing solution focused on resource isolation of co-existing virtual networks on the physical data-plane infrastructure. The idea of HyperFlex is to separate the OpenFlow proxy network from the physical infrastructure to isolate the virtualization functions from the data-plane. Therefore, it is built based on decomposition of the hypervisor virtualization functions that allows a functional-granularity to achieve a more tailored operation and enhance the resource efficiency. On the other hand, HyperFlex provides translation function that mapping NVs controller messages to their physical resources. This function could be implemented in the network elements according to their packet inspection capabilities or use policy function that ensures the NVs controller can only access their virtual resources. A main feature of HyperFlex is the ability to offload the load of performing the virtualization functions between the software and the proxy network elements. Such the feature allows exploiting the resources of the servers that host the proxy software and extend the resources of the OpenFlow proxy network to improve the virtualization performance.

6.4 DFVisor

A layered overlay proxy controller that comes to improve address space in VNs, reduces flow setup latency, and removes the single point of failure in the slice network. DFVisor has three components enhanced OF enabled switches, enhanced VN controllers, and the distributed synchronized two-level database system. To enhance OF switch, DFVisor introduced three features: hybrid data forwarding that can be easily implemented into OF, tunneling and resource slicing. Tunneling used Generic Routing Encapsulation (GRE) module into the data plane by extending the OF switch specification. The resource slicing in DFVisor is different from traditional slices that lacks a native network addressing mechanism and causes scalability. It used typical statistic mechanism used in OF switch specifications to enhance the scalability by reducing the flow setup latency. In the third component, two-level of the database multiple local databases and a distributed global database are implemented. Multiple local databases store the network configuration and information and can be implemented in Apache Zookeeper wit watch service that can allow each local database to use watchers on the global database and receive a watcher alarm when the database are updated.

6.5 Autoslice

Distributed OF proxy controller that can handle large numbers of flow table and control messages from multiple tenants. It is focused on the scalability of VNse by distributing the controller load. In the FlowVisor, the first OpenFlow proxy slice technique is implemented in the flow table inside the OF switches by partitioning it into so-called flows-paces. Such technique enables OF switch manipulated concurrently by multiple controllers, but this operation includes mapping VNs topologies, installing flow entries for tunneling and enforcing flow table isolation. All these operations require considerable planning and management resources. Therefore, Auto slice proposed to automate the deployment and operation of VNs with minimal intervention. In addition, Autoslice optimizes resource utilization and mitigates flow-table limitations through the monitoring of flow-level traffic statistics. After deploying the VN, each tenant has ability to fully control the flow table that can be sliced into logical segments irrespective of the services that the network provided. Autoslice consists of two components; 1) a management module (MM) that maps the virtual topology to the resources available in each domain 2) and multiple controller (CPX) that is responsible to assign a dedicated CPX to each domain. A policy based approach is implemented for each CPX for accessing the flow table which ensures that the flow entries are mapped onto non-overlapping flow-spaces.
Though, the above proxy controllers provide different features and address various issues of traditional network virtualization. Nevertheless there are several important open future research directions for OF network virtualization in terms of security, scalability and monitoring of VNs especially in multi tenants environment that is widely adopted in current. In section 5 challenges and open issues are discussed.

7. Challenges and Future Research Directions

OpenFlow-based NV has become widely utilized, and controllers, such as Flow Visor, have been further defined. Consequently, new solutions have been proposed, and new challenges have arisen. In this section, we discuss several challenges posed by NV based on OpenFlow. We also provide directions for future research.

7.1 Security challenges in OpenFlow network virtualization

Security concerns are obvious in virtual networks; however, security issues that specifically refer to virtual networks remain unaddressed[73]. These issues arise when networks are programmable. Programmability may cause network vulnerability, subject to the absence of well-structured policies and rules. The research community has not shown the security level of virtual networks compared with traditional networks[74] nor provided sufficient security measures to defend such networks. The security of network virtualization technologies has become increasingly important. Moreover, NV based on OpenFlow technology employs FlowVisor as a transparent proxy. We can divide security issues into three main categories: guest controller, FlowVisor, and switch security issues.

7.1.1 Guest controller security issues

In OpenFlow NV, a user has his own controller that works as NOS[75] for a specific virtual network. NOS contains critical knowledge of the network and is prone to attacks. An example is the denial of service (DOS) attack, where an attacker can generate numerous flows that overload the components, leading to network failure. Moreover, the application layer, which communicates with the controller through the northbound interface, also exhibits potential vulnerability as the application is prone to be reprogrammed by hackers. The installation of compromised applications may affect the entire virtual network.

Field Rewrite is a problem that is specifically related to a situation whereas controller has access to a specific VLAN ID tag. Field rewrite can then create flow entries with actions that change the VLAN ID of its own packets, thereby creating an opportunity for a malicious controller to inject packets into another slice. This occurrence actually occurs when a controller only has access to packets with transport source port A[76]. If a controller attempts to create a flow entry with an unspecified transport source port (wildcard), FlowVisor should rewrite the wildcard value to the valid one (A). However, if this situation does not occur and the flow entry is created with the wildcard field, FlowVisor can match any transport source port. This procedure is repeated for another header value, such as transport destination and protocol type.

7.1.2 FlowVisor security issues

Given that Hypervisor controls everything [77],it causes a single point of failure in the virtual environment. Consequently, a single breach can cause a disaster in the entire environment. FlowVisor provides address space isolation; some of the implementation details of FlowVisor hinder this property. Furthermore, FlowVisor does not implement action isolation, which means no control exists over which types of actions a controller may set on a flow entry. Thus, it is not possible to consider that all virtual networks manage by FlowVisor are secure. A virtual network may affect the traffic of other networks, intentionally or not[78]. Therefore several kinds of attacks can effect virtual network that used Hypervisor controller. For example, modify flow rules by inject packets to steal tenants information, implemented malicious actions or faulty behaviour of the isolation mechanisms.

7.1.3 Switch security issues

Packets that originate from an untrusted host can cause tampering issues. For example, Floodlight Controller send every 5 second LLDP packet to update network topology and manage flow rules in the switch. With forged source addresses cause the controller to install flow rules based on false information. TO avoid this king of attack, Role-based authenticated approach must be implemented for each flow rule producer. In addition, the channel between the controller and switches can also be vulnerable. whereas, the communication between the controller and switch could be in plaintext[76]. However, according to the OpenFlow specification, transport layer security (TLS) can be utilized to secure the communication. TLS can be enhanced by add module that help to detect malformed packets before send it to the controller.

7.2 Future directions and Open issues in OpenFlow network virtualization

7.2.1 Scalability and reliability

FlowVisor is a bottleneck in all virtual networks[79]. It is responsible for handling all packets between the guest controller and the switch. Performance issues arise when too many packets are interchanged. Many experiments on Testbeds have been conducted, with the use of FlowVisor to create isolations among virtual networks. The results show a significant effect, especially when numerous flows are sent. This observation indicates the importance of scalability issues in OpenFlow NV.

In addition to reliability and the problem in network resources [80] ,such as host CPU in the OpenFlow controller, Dabkiewicz [81] found that many new flows also seriously affect the performance of FlowVisor. Numerous errors are recorded in the FlowVisor log file for debugging because of some reasons. When a network disaster, such as flash crowding, occurs, hundreds of errors arise and are recorded every second. These errors could eventually fill the disk up and cause system failure.

In addition, increasing the number of tenants with different abstractions results in scalability challenges. For example, increasing the number of virtual networks (each one with different topologies) provides a means to map a rule or query issued in a virtual network to the corresponding physical switches and to map a physical event (e.g., a link or switch failure) to the affected virtual components.

1.1.1 Isolation

The resources of the switch are separated according to the created slices. A slice cannot enter flow entries for a slice they do not belong to. Separating switch resources can be done based on the application port or VLAN membership. FlowVisor ensures that only the responsible controller can add flow entries. However, when using wildcard flow entries, such as in experiment 7 [81], the switch resources in the separation of OpenFlow do not always work as one would expect. In addition, the isolation provided by OpenFlow primarily implemented in link layer for each tenant, For example, if a tenant has weak network security procedures, information disclosure may occur, resulting in a breach of isolation at higher layers. Moreover, a rogue SDN app with privileges that span beyond isolation borders may impact overall network security by steering traffic to a third party (information disclosure) by over- or under-billing (theft of service) or by dropping traffic. Thus strong security procedures for tenants must be implemented.

1.1.2 Monitoring

Monitoring the virtual network is an important factor in open-access networks [82]. In VTN, virtual networks are assigned to different customers who wish to monitor and manage their network effectively. Several protocols, such as sFlow [83],can be utilized to monitor traffic in the SDN environment. Nevertheless, NV still fails to implement an effective monitoring system. NV may utilize an extension of current monitoring tools, such as Nagios[84].

1.1.3 Handling of switch events

The handling of switch events is performed by FlowVisor in a manner that the switch events are sent to the appropriate OpenFlow controller responsible for the slice where the switch event occurs. However, in a situation where an OpenFlow controller is only connected to a switch port, the messages regarding the switch event cannot be sent to all possible OpenFlow controllers. To address this issue, we can refer to only one place to look for switch events implemented in monitor slices that obtain all switch events[81]. The switch events are sent through the OpenFlow protocol and interpreted by FlowVisor and the OpenFlow controller. A module that obtains the OpenFlow messages from FlowVisor and/or the OpenFlow controller could be developed to receive all messages at a single point.

8. Conclusion

OpenFlow technology has paved the way toward next-generation networks and has provided advanced functionality. OpenFlow technology provides network virtualization that is versatile, flexible, and easy to manage. It also creates an opportunity to solve several network virtualization problems, such as the network address space, virtual topology, and tenant network management. This paper presents a comprehensive survey of OpenFlow-based network virtualization. We began by providing a background overview of the separation of the control framework from underlying switches to achieve NVs. We focused on OpenFlow components and compared conventional network architecture and OpenFlow architecture. We developed an OpenFlow-based network virtualization taxonomy that involves three approaches, namely, proxy virtualization, layer 2 virtualization, and programming virtualization. A comparative analysis of available OpenFlow Testbeds that support network virtualization was also presented. Lastly, several challenges posed by OpenFlow-based network virtualization and future research directions were discussed.

Acknowledgments

This work is fully funded by Bright Spark Unit, University of Malaya, Malaysia and partially funded by Malaysian Ministry of Higher Education under the University of Malaya High Impact Research Grant UM.C/625/1/HIR/MOE/FCSIT/03

References

[1] Nunes, B.A.A., et al., “A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks,” *Communications Surveys & Tutorials, IEEE*, 16(3): p. 1617-1634, 2014. Article (CrossRef Link)

[2] Chowdhury, N.M.K. and R. Boutaba, “Network virtualization: state of the art and research challenges,” *Communications Magazine, IEEE*, 47(7): p. 20-26, 2009. Article (CrossRef Link)

[3] Schaffrath, G., et al., “Network virtualization architecture: proposal and initial prototype,” in *Proc of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, ACM, 2009. Article (CrossRef Link)

[4] Bari, M.F., et al., “Data center network virtualization: A survey,” *Communications Surveys & Tutorials, IEEE*, 15(2): p. 909-928, 2013. Article (CrossRef Link)

[5] Duan, Q., Y. Yan, and A.V. Vasilakos, "A survey on service-oriented network virtualization toward convergence of networking and cloud computing," *Network and Service Management, IEEE Transactions on*, 9(4): p. 373-392, 2012. Article (CrossRef Link)
[6] Wang, A., et al., "Network virtualization: Technologies, perspectives, and frontiers," *Lightwave Technology, Journal*, 31(4): p. 523-537, 2013. Article (CrossRef Link)
[7] Casado, M., et al., "Virtualizing the network forwarding plane," in *Proc. of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, ACM, 2010. Article (CrossRef Link)
[8] Wang, G. and T.E. Ng, "The impact of virtualization on network performance of amazon ec2 data center," in *Proc. of INFOCOM, 2010 Proceedings IEEE*, 2010. Article (CrossRef Link)
[9] Voith, T., K. Oberle, and M. Stein, "Quality of service provisioning for distributed data center inter-connectivity enabled by network virtualization," *Future Generation Computer Systems*, 28(3): p. 554-562, 2012. Article (CrossRef Link)
[10] Guo, C., et al., "Secondnet: a data center network virtualization architecture with bandwidth guarantees," in *Proc. of the 6th International Conference*, ACM, 2010. Article (CrossRef Link)
[11] Yadav, Y. and P. Yadav, Virtual Local Area Network, 1(11): p. 42-249, 2013. Article (CrossRef Link)
[12] Venkateswaran, R., Virtual private networks. Potentials, IEEE, 20(1): p. 11-15, 2001. Article (CrossRef Link)
[13] Jain, R., Introduction to Software Defined Networking (SDN), 2013.Article (CrossRef Link)
[14] Van der Merwe, J.E., et al., "The tempest-a practical framework for network programmability," *Network, IEEE*, 12(3): p. 20-28, 1998.. Article (CrossRef Link)
[15] Bavier, A., et al., "In VINI veritas: realistic and controlled network experimentation," in *Proc. of ACM SIGCOMM Computer Communication Review*, ACM, 2006. Article (CrossRef Link)
[16] Morris, R., et al., "The Click modular router," in *Proc. of ACM SIGOPS Operating Systems Review*, ACM, 1999. Article (CrossRef Link)
[17] Handley, M., O. Hodson, and E. Kohler, "XORP: An open platform for network research," *ACM SIGCOMM Computer Communication Review*, 33(1): p. 53-57, 2003. Article (CrossRef Link)
[18] Feamster, N., L. Gao, and J. Rexford, "How to lease the Internet in your spare time," *ACM SIGCOMM Computer Communication Review*, 37(1): p. 61-64, 2007. Article (CrossRef Link)
[19] McKeown, N., et al., "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, 38(2): p. 69-74, 2008. Article (CrossRef Link)
[20] Bozakov, Z. and V. Sander, "OpenFlow: A Perspective for Building Versatile Networks," *Network-Embedded Management and Applications*, Springer. p. 217-245, 2013. Article (CrossRef Link)
[21] Vaughan-Nichols, S.J., "OpenFlow: The next generation of the network?" *Computer*, 44(8): p. 13-15, 2011. Article (CrossRef Link)
[22] El-Azzab, M., et al., "Slices Isolator for a Virtualized Openflow Node," in *Proc. of Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on*, IEEE, 2011. Article(CrossRef Link)
[23] Azodolmolky, S., "Software Defined Networking with OpenFlow," *Packt Publishing Ltd*, 2013. Article(CrossRef Link)
[24] Mattes, J., "Traffic Measurement on OpenFlow-enabled Switches," PhD thesis. ETH Zürich, 2012. Article (CrossRef Link)
[25] Shahmir Shourmasti, K., Stochastic Switching Using OpenFlow. 2013.Article(CrossRef Link)
[26] ONF, https://www.opennetworking.org/. [accessed: 27/08/2014].Article(CrossRef Link)
[27] Ren, T. and Y. Xu., "Analysis of the New Features of OpenFlow 1.4," in *Proc. of 2nd International Conference on Information, Electronics and Computer*, Atlantis Press, 2014. Article (CrossRef Link)
[28] SUZUKI, K., et al., "A Survey on OpenFlow Technologies," *IEICE TRANSACTIONS on Communications*, 97(2): p. 375-386, 2014. Article (CrossRef Link)

[29] Fernandez, M.P., "Comparing OpenFlow Controller Paradigms Scalability: Reactive and Proactive. in Advanced Information Networking and Applications (AINA)," in *Proc. of 2013 IEEE 27th International Conference on*, IEEE, 2013.

[30] Gude, N., et al., "NOX: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, 38(3): p. 105-110, 2008. Article (CrossRef Link)

[31] POX, http://www.noxrepo.org. [accessed: 27/08/2014]. Article(CrossRef Link)

[32] Erickson, D., "The beacon openflow controller," in *Proc. of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, ACM, 2013. Article (CrossRef Link)

[33] Floodlight, http://www.projectfloodlight.org/floodlight/. [accessed: 27/08/2014]. Article(CrossRef Link)

[34] Cai, Z., A.L. Cox, and T.E.N. Maestro, "A system for scalable OpenFlow control," *Technical Report TR10-08*, Rice University, 2010. Article (CrossRef Link)

[35] Trema, http://trema.github.io/trema/. [accessed: 27/08/2014]. Article(CrossRef Link)

[36] OpenDayLight, http://www.opendaylight.org/. [accessed: 27/08/2014]. Article(CrossRef Link)

[37] Ryu, http://osrg.github.io/ryu/. [accessed: 27/08/2014]. Article(CrossRef Link)

[38] Sherwood, R., et al., "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium*, Tech. Rep, 2009.

[39] RouteFlow, http://cpqd.github.io/RouteFlow/. [accessed: 27/08/2014]. Article(CrossRef Link)

[40] ONOS, http://tools.onlab.us/onos.html. [accessed: 27/08/2014]. Article(CrossRef Link)

[41] Salvadori, E., et al., "Generalizing virtual network topologies in OpenFlow-based networks," in *Proc. of Global Telecommunications Conference (GLOBECOM 2011)*, 2011 IEEE, 2011. Article (CrossRef Link)

[42] Tsai, P.-W., et al., "Supporting Extensions of VLAN-tagged Traffic across OpenFlow Networks," in *Proc. of Research and Educational Experiment Workshop (GREE), 2013 Second GENI*, IEEE, 2013. Article(CrossRef Link)

[43] Matias, J., et al., "Implementing Layer 2 Network Virtualization using OpenFlow: Challenges and Solutions," in *Proc. of Software Defined Networking (EWSDN), 2012 European Workshop on*, IEEE, 2012. Article(CrossRef Link)

[44] Das, S., G. Parulkar, and N. McKeown, "Unifying packet and circuit switched networks," in *Proc. of GLOBECOM Workshops, 2009 IEEE*, IEEE, 2009. Article (CrossRef Link)

[45] Gutz, S., et al., "Splendid isolation: A slice abstraction for software-defined networks," in *Proc. of the first workshop on Hot topics in software defined networks*, ACM, 2012. Article (CrossRef Link)

[46] Chen, Y., et al., "VNMC for network virtualization in OpenFlow network," in *Proc. of Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, IEEE, 2012. Article (CrossRef Link)

[47] Azodolmolky, S., et al., "Optical FlowVisor: an OpenFlow-based optical network virtualization approach," in *Proc. of National Fiber Optic Engineers Conference*, Optical Society of America, 2012. Article (CrossRef Link)

[48] Doriguzzi Corin, R., et al., "VeRTIGO: network virtualization and beyond," in *Proc. of Software Defined Networking (EWSDN), 2012 European Workshop on*, IEEE, 2012. Article(CrossRef Link)

[49] Salvadori, E., et al., "Demonstrating generalized virtual topologies in an openflow network," in *Proc. of ACM SIGCOMM Computer Communication Review*, ACM, 2011. Article (CrossRef Link)

[50] Giatsios, D., et al., "Integrating FlowVisor Access Control in a Publicly Available OpenFlow Testbed with Slicing Support," in *Proc. of Testbeds and Research Infrastructure. Development of Networks and Communities*, Springer. p. 387-389, 2012. Article (CrossRef Link)

[51] Matias, J., et al., "An OpenFlow based network virtualization framework for the cloud," in *Proc. of Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, IEEE, 2011. Article(CrossRef Link)

[52] Yin, X., et al., "Software defined virtualization platform based on double-FlowVisors in multiple domain networks," in *Proc. of Communications and Networking in China (CHINACOM), 2013 8th International ICST Conference on*, IEEE, 2013. Article(CrossRef Link)

[53] Das, S., et al, "MPLS with a simple OPEN control plane," in *Proc. of Optical Fiber Communication Conference*, Optical Society of America, 2011. Article (CrossRef Link)

[54] Sharafat, A.R., et al., "Mpls-te and mpls vpns with openflow," in *Proc. of ACM SIGCOMM Computer Communication Review*, ACM, 2011. Article (CrossRef Link)

[55] Salmito, T., et al., "FIBRE-An International Testbed for Future Internet Experimentation," *Anais do 32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC 2014*, 2014. Article(CrossRef Link)

[56] Moraes, I.M., et al., "FITS: A Flexible Virtual Network Testbed Architecture," *Computer Networks*, 2014. Article (CrossRef Link)

[57] Su-é, M., et al., "Design and implementation of the OFELIA FP7 facility: The European OpenFlow testbed," *Computer Networks*, 2013.Article(CrossRef Link)

[58] Hu, J.-W., C.-S. Yang, and T.-L. Liu, "Design and implementation of an efficient and programmable future internet testbed in Taiwan," *Computer Science and Information Systems*, 10(2): p. 825-842, 2013. Article (CrossRef Link)

[59] Min, S.H., B.C. Kim, and J.Y. Lee, "NetFPGA-based scheduler implementation for resource virtualization of Future Internet testbed," in *Proc. of ICT Convergence (ICTC), 2011 International Conference on*, IEEE, 2011. Article(CrossRef Link)

[60] Riggio, R., T. Rasheed, and F. Granelli, "Empower: A testbed for network function virtualization research and experimentation," in *Proc. of Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, IEEE, 2013. Article(CrossRef Link)

[61] Roy, A.R., et al., "Design and Management of DOT: A Distributed OpenFlow Testbed," in *Proc. of 14th IEEE/IFIP Network Operations and Management Symposium (NOMS 2014)(To appear)*, 2014. Article (CrossRef Link)

[62] Berman, M., et al., "GENI: A federated testbed for innovative network experiments," *Computer Networks*, 2014. Article (CrossRef Link)

[63] Thanh, N.H., et al., "ECODANE: A customizable hybrid testbed for green data center networks," in *Proc. of Advanced Technologies for Communications (ATC), 2013 International Conference on*, IEEE, 2013. Article(CrossRef Link)

[64] Kanaumi, Y., "Large-scale OpenFlow testbed in Japan," in *Proc. of The 31st APAN Meeting*, 2011. Article (CrossRef Link)

[65] Georgopoulos, P., et al., "Cache as a service: Leveraging sdn to efficiently and transparently support video-on-demand on the last mile," in *Proc. of Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*, IEEE, 2014. Article (CrossRef Link)

[66] Tai, D.Y. and F.Y. Xu., "Cloud manufacturing based on cooperative concept of SDN," in *Proc. of Advanced Materials Research*, Trans Tech Publ, 2012. Article(CrossRef Link)

[67] Berryman, A., et al., "Advanced manufacturing use cases and early results in GENI infrastructure," in *Proc. of Research and Educational Experiment Workshop (GREE), 2013 Second GENI*, IEEE, 2013. Article(CrossRef Link)

[68] Al-Shabibi, A., et al., "OpenVirteX: Make your virtual SDNs programmable," in *Proc. of the third workshop on Hot topics in software defined networking*, ACM, 2014. Article (CrossRef Link)

[69] Drutskoy, D.A., "Software-Defined Network Virtualization with FlowN," *Master's Thesis*, Princeton University, 2012. Article(CrossRef Link)

[70] Blenk, A., A. Basta, and W. Kellerer, "HyperFlex: An SDN virtualization architecture with flexible hypervisor function allocation," in *Proc. of Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, IEEE, 2015. Article (CrossRef Link)

[71] Liao, L., V. Leung, and P. Nasiopoulos, "DFVisor: Scalable network virtualization for QoS management in cloud computing," in *Proc. of Network and Service Management (CNSM), 2014 10th International Conference on*, IEEE, 2014. Article(CrossRef Link)

[72] Bozakov, Z. and P. Papadimitriou, "Autoslice: automated and scalable slicing for software-defined networks," in *Proc. of the 2012 ACM conference on CoNEXT student workshop*, ACM, 2012.

Article (CrossRef Link)
[73] Yan, Z., P. Zhang, and A.V. Vasilakos, "A security and trust framework for virtualized networks and software-defined networking," *Security and Communication Networks*, 2015. Article(CrossRef Link)
[74] Psounis, K., "Active networks: Applications, security, safety, and architectures," *Communications Surveys & Tutorials, IEEE*, 2(1): p. 2-16, 1999. Article (CrossRef Link)
[75] Yang, M., et al., "Software-Defined and virtualized future mobile and wireless networks: A Survey," *Mobile Networks and Applications*, p. 1-15, 2014. Article (CrossRef Link)
[76] Victor T. Costa, "L.ı.H.M.K.C.," *Vulnerability Study of FlowVisor-based Virtualized Network Environments*, 2013. Article(CrossRef Link)
[77] Heller, B., R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. of the first workshop on Hot topics in software defined network*, ACM, 2012. Article (CrossRef Link)
[78] Jaiganesh, M. and V.A. Kumar, "SOV2C2: Secure Orthogonal View of Virtualization in Cloud Computing," *SmartCR*, 2(4): p. 278-285, 2012. Article (CrossRef Link)
[79] Wen, H., P.K. Tiwary, and T. Le-Ngoc, "Network Virtualization Technologies and Techniques," *Wireless Virtualization*, Springer. p. 25-40, 2013. Article(CrossRef Link)
[80] Sookhak, M., et al., "Towards Dynamic Remote Data Auditing in Computational Clouds," *The Scientific World Journal*, 2014. Article (CrossRef Link)
[81] Dabkiewicz, S., R. van der Pol, and G. van Malenstein, OpenFlow network virtualization with FlowVisor, 2012.Article(CrossRef Link)
[82] Sookhak, M., et al., "A review on remote data auditing in single cloud server: Taxonomy and open issues," *Journal of Network and Computer Applications*, 43: p. 121-141, 2014. Article (CrossRef Link)
[83] Giotis, K., et al., "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Computer Networks*, 62: p. 122-136, 2014. Article (CrossRef Link)
[84] Harlan, R.C., Network management with Nagios. Linux Journal, (111): p. 3, 2013. Article(CrossRef Link)

Ahmed Abdelaziz received the M.IT. in University of Malaya in 2007. He is currently a Ph.D. candidate at the Department of Compute Systems and Technology, University of Malaya, Kuala Lumpur, Malaysia. His research interests include SDN, NFV, Load Balance in the Cloud and Network virtualization.

Tan Fong Ang did his Masters in computer science from University of Malaya in 2001. He obtained his Ph.D. from University of Malaya in year 2011. His research area includes resource allocation, Cloud Computing, Software Defined Network, and network security.

Mehdi Sookhak is a postdoctoral fellow at Carleton University of Canada funded by Canadian Natural Sciences and Engineering Research Council (NSERC). His areas of interest include Cryptography and Information Security, Mobile Cloud Computing, Computation outsourcing, Access control, Wireless Sensor & Mobile Ad Hoc Network (Architectures, Protocols, Security, and Algorithms), and Distributed Systems

Suleman Khan is currently a PhD student under Bright Spark Scholarship, University of Malaya, Malaysia, since October 2013. He has published more than 20 research peer review articles in different journals and international conferences while having research interest in, Software defined networks, Network Virtualization, Network Forensics, Internet of Things, Big Data, and Cloud computing security.

Athanasios V. Vasilakos is currently a Professor with the University of Western Macedonia, Greece. He has authored or coauthored over 200 technical papers in major international journals and conferences, five books, and 20 book chapters. He served as a General Chair, a Technical Program Committee Chair for many international conferences. He served or is serving as an Editor of the IEEE renowned transactions and, also a General Chair of the Council of Computing of the European Alliances for Innovation.

Chee Sun Liew completed his Masters of Science (Computer Science), in Distributed Computing and Networks from University Sains Malaysia in 2002. He holds a Ph.D. in Informatics from the University of Edinburgh, under Malaysia Ministry of Higher Education scholarship program. His Ph.D. research was related to workflow optimization, under the supervision of Prof Malcolm Atkinson

Adnan Akhunzada is currently a Ph.D. Fellow and an Active Researcher with the Center for Mobile Cloud Computing, University of Malaya, Malaysia. He had a great experience teaching international modules. He is a Senior Lecturer with CIIT, Islamabad, since 2011. He is author/coauthor in several high-impact major journal publications, conferences, and a book chapter.