This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

A Survey on Large-scale Software Defined Networking (SDN) Testbeds: Approaches and Challenges

Tao Huang*†, F. Richard Yu‡, Chen Zhang*†, Jiang Liu*†, Jiao Zhang*†, and Yunjie Liu*†
*State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications, Beijing, P.R. China
†Beijing Advanced Innovation Center for Future Internet Technology, Beijing, P.R. China
‡Depart. of Systems and Computer Eng., Carleton University, Ottawa, ON, Canada

***Abstract*—Recently, several large-scale Software Defined Networking (SDN) testbeds have been designed and developed. These SDN testbeds have spurred numerous network researchers to run their prototypes and experiments, as well as to set up novel architectures for the future Internet. Based on these efforts, SDN testbeds are actively contributing to future network research, bringing forth the attention of both academia and industry. In addition, researchers envisage these large-scale SDN testbeds to be the future Internet. In this paper, we present a comprehensive survey and research challenges for large-scale SDN testbeds. We first introduce the related work and background knowledge. Then, an overview of SDN testbeds is presented. In addition, five typical implementations of large-scale SDN testbeds around the world are described in detail, including design objectives, key technologies, deployment, and experiments. Moreover, an in-depth comparison of SDN testbeds is given. Finally, challenges and future works of SDN testbeds are discussed.**

***Index Terms*—SDN, testbed, OpenFlow, GENI, OFELIA, OpenLab, RISE, OF@TEIN**

I. Introduction

OVER the past decade, with the tremendous advances in the field of information and communications technology (ICT), traditional Internet has faced severe challenges. For example, the rapid development of wireless technology has forced new research on mobility management [1], [2]. Massive large-scale data centers raise serious energy-saving problems [3]. Time-sensitive businesses in the cloud require preferable quality of service from the underlying network [4]. Furthermore, the shortage of Internet Protocol version 4 (IPv4) addresses, urgent need for management automation, difficulties of scalable routing, and many other problems similar to these are puzzling network researchers.

Recently, Software Defined Networking (SDN) [5] has attracted great interests from both academia and industry to solve these issues. The basic idea of SDN is to break vertical integration, to detach the control plane from the forwarding plane, and to introduce the ability of programming the network. The SDN controller, acting as the Networking Operating System (NOS), separates remotely from the underlying devices, so that one can program the network anytime and anywhere. The good features of SDN enable new business to be deployed quickly, new academic networking methods could be easily achieved. Currently, OpenFlow [6] is the most promising technique to standardize the communication between the controller and devices, and it frees the designing of "network middle boxes" from restrictive layer frameworks. The wide spread of OpenFlow is inspiring increasing number of SDN researches.

To evaluate the performance of new network designs and algorithms, simulators, emulation platforms and prototype testbed are usually used by researchers. Compared with simulators and emulation platforms, testbeds are more convincing for network experiments, because testbeds can incorporate real traffic and real network facilities. To serve SDN research, several large-scale testbeds constructed by the SDN method have been designed and developed. Using network slicing technologies [7]–[9], SDN testbeds could transparently provide different researchers with isolated network resources over the same suite of physical facility. Combined with cloud management tools, such as OpenStack [10], SDN researchers could "rent" virtual servers as they need the resources only for a short period of time. And depending on the testbed control framework [11], [12], most of the devices and research projects could be easily managed.

What is more appealing, researchers envisage these large-scale SDN testbeds to be the future Internet. Actually, the past 10 years have seen many convincing attempts to enlarge the SDN domain [13], [14], to construct robust SDN control plane [15], [16], and to interconnect heterogeneous SDN devices [17]–[19]. It is possible that the envision will come true soon, after all, Advanced Research Projects Agency Network (ARPANET) has evolved to be the current Internet in just less than half a century.

Trials of large-scale SDN testbeds are actively advancing in many countries [20]–[24], and most of them are based on OpenFlow. The Global Environment for Network Innovation (GENI) OpenFlow project in the USA [20], OFELIA [21] and OpenLab [24] in Europe, RISE in Japan [22] and OF@TEIN [25] in Korea are the most typical and general ones, that have attracted numerous network researchers to run their prototypes and experiments, or to set up their novel architectures. Based on these efforts, SDN testbeds are actively contributing to future network research, capturing the attention of both academia and industry.

In this paper, some typical and general implementations of large-scale SDN testbeds are introduced in detail. Particularly, we present the design objectives, key technologies,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

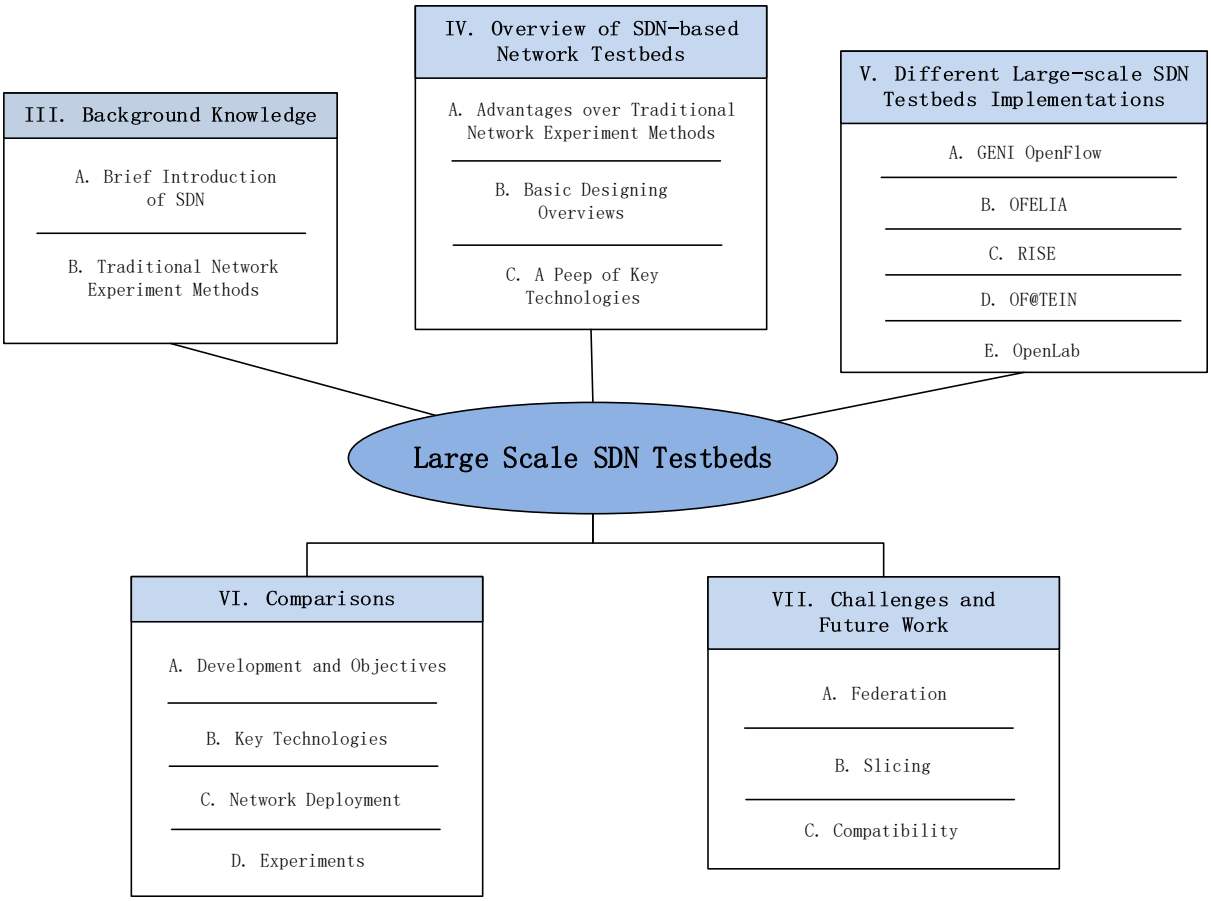

Figure 1. Roadmap of this paper.

network deployment and experiments of these large-scale SDN testbeds. In addition, we discuss the research challenges of large-scale SDN testbeds. To the best of our knowledge, this is the first comprehensive survey on large-scale SDN testbeds.

The structure of the paper is given in Figure 1. In Section II and III, related works and background knowledge are briefly introduced. Section IV gives an overview of SDN testbeds. Section V introduces the implementations of GENI OpenFlow, OFELIA, RISE, OF@TEIN and OpenLab, from the basic elements to the key technologies. There are other interesting SDN testbeds, such as JOLNET in Italy [26] and CoCo in GN3 [27], which are difficult to be described in details in this paper due to the limited space. In Section VI, crosswise comparisons are made among these testbeds. Subsequently in Section VII, we discuss the challenges and future works on SDN testbeds. Finally the paper concludes in Section VIII. Table I lists the main abbreviations used in this paper.

II. Related Work

There are several excellent survey papers about SDN (e.g., [28]–[41]). Kreutz *et al.* [28] have presented a comprehensive view of SDN components with a layer perspective. Xia *et al.* [29] have studied the SDN architecture from a layered approach, and clarified each layer with related works. Hu *et al.* [30] have surveyed some specific topics in SDN/OpenFlow implementations, such as Quality of Service (QoS) and security. Xie *et al.* [31] have researched on issues related to SDN controllers, including architecture, performance, scalability, placement, interface, and security. Bhaumik *et al.* [32] focus on Software-Defined Optical Networks (SDONs), from general concepts to its benefits over Generalized Multi-Protocol Label Switching (GMPLS). Software-Defined Wireless Networks (SDWNs) have been studied in [37], [42]–[44]. SDN hypervisors have been surveyed in [38], which categorizes and sub-classifies SDN hypervisors, and does some valuable comparisons and evaluations. Bari *et al.* [39] have presented a deep review of network virtualization in data centers, and a comprehensive comparison of existing approaches is made in the paper. Rygielski *et al.* [40] have surveyed existing network virtualization methods briefly, and specifically characterized them according to QoS management and performance isolation. Liang *et al.* [41], [45], [46] have surveyed virtualization approaches in wireless network, and discussed future research challenges and directions.

Although some excellent surveys have been done on different aspects of SDN, most of them do not focus on SDN testbeds. To fill this gap, This paper presents a comprehensive survey and research challenges of large-scale SDN testbeds. We hope that it can help readers to have an overall understanding of this field and for researchers to do subsequent studies.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

TABLE I
ABBREVIATIONS

AAA	Authentication, Authorization, and Accounting
ARPANET	Advanced Research Projects Agency Network
BGP	Border Gateway Protocol
CPEX	CaPital Expenditures
DDoS	Distributed Denial of Sevices
DPID	DataPath Identification
EoMPLS	Ethernet over Multi-Protocol Label Switching
EXPRESS	EXPerimenting and Research Evolutions of Software-defined networking over federated testbedS
ForCES	Forward and Control Element Separation
ICN	Information-Centric Networking
IPv4	Internet Protocol version 4
GMPLS	Generalized Multi-Protocol Label Switching
JGN	Japan Gigabit Network
LDAP	Lightweight Directory Access Protocol
MPLS	Multi-Protocol Label Switching
NLR	National Lambda Rail
NS	Network Simulator
NVGRE	Network Virtualization using Generic Routing Encapsulation
OAM	Operation Administration and Maintenance
OF-CONFIG	OpenFlow Configuration Protocol
OFELIA	OpenFlow in Europe Linking Infrastructure and Applications
OPEX	OPerational Expenditures
OVS	Open Virtual Switch
OVSDB	Open-vSwitch-Database-Management-Protocol
OVX	OpenVirteX
PCE	Path Computation Element
PSP-SEC	Secured Path State Protocol
PSTN	Public Switched Telephone Network
RCP	Routing Control Protocol
RIB	Routing Information Base
SSH	Secure SHell
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
WDM	Wavelength Division Multiplexing

III. Background Knowledge

In this section, the background knowledge on SDN and some traditional network experiment methods are briefly introduced.

A. Brief Introduction of SDN

SDN is becoming widespread in recent years. Actually, the history of SDN has gone through a long period of evolution [47]. To achieve network programmability, Active Network (AN) [48], which could introduce some programmability into network devices, was first raised in the late 1990s, however, its control logic was still distributed in devices. The earliest attempt to separate control logic from forwarding devices is the Network Control Point (NCP) [49], subsequently, many similar attempts were proposed, such as ForCES [50], RCP [51], PCE [52] and 4D [51], [53], [54]. Compared to AN, these projects were designed to program the control plane rather than the data plane.

The embryo of SDN was first raised up in 2006 in the Stanford's Clean-State project. Ethane [55], an approach of centralized and programmable control in enterprise network,

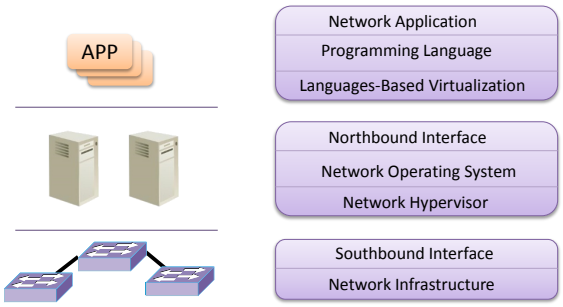

Figure 2. Layered architecture of SDN. The Application plane consists of different types of SDN Apps. The Northbound Interface provides APIs used for management or development. The NOS facilitates network control and management logic. The Network Hypervisor could be used to slice the physical infrastructure to provide logical networks. The Southbound Interface connects the control plane and the data plane. The data plane is composed of SDN forwarding devices.

presented the early stage SDN method. As the name implies, SDN is a new network architecture based on user-defined software logic, and the design concept of SDN architecture is to detach control logic from forwarding devices. Thus, the centralized controller will have a global view of the network resources, which is essential for network optimization, e.g., to improve bandwidth utilization and to ensure differential network transmissions for different traffic. Moreover, SDN is more flexible than the traditional network due to its programmability, allowing users to develop applications to control their network. Thus, new services could be rapidly deployed and the Operation Administration and Maintenance (OAM) costs will be sharply reduced.

A layered SDN architecture is shown in Figure 2. Basically, SDN comprises three layers, including the data plane layer, the control plane layer and the application plane layer, and each layer is composed of different sub-layers. The control plane layer is the "brain" of SDN, and it could consist of several sub-layers [28]: Northbound Interface, Network Operating System (NOS), Network Hypervisor and Southbound Interface. The Northbound Interface is usually a suite of application programming interfaces (APIs) used for management or development. The NOS facilitates network control and management logic. The Network Hypervisor lies right below the NOS to enable logical switches and links, and optionally it could be used to slice the physical infrastructure to provide logical networks. The Southbound Interface plays the most important part as the "artery", connecting the control and forwarding devices as well as transacting information between them.

OpenFlow [6] is one of the most representative SDN Southbound Interfaces, and Figure 3 depicts a toy OpenFlow network [56]. An OpenFlow switch consists of at least three parts: a flow table, a secure channel and the OpenFlow Protocol. The flow table includes some flow entries to determine the processing methods of specific traffic. The secure channel conveys OpenFlow messages and it could be implemented in some secure transporting mechanisms. The OpenFlow Protocol standardizes the OpenFlow messages on the secure channel and the behaviors of OpenFlow switches. When an

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

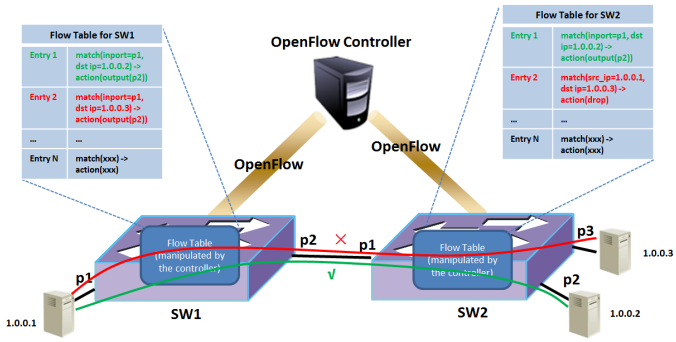

Figure 3. OpenFlow controller distributes flow entries into switches' flow tables to direct traffic forwarding. For example, packets with destination addresses to 1.0.0.2 and 1.0.0.3 are both allowed and forwarded at SW1, but packets from 1.0.0.1 to 1.0.0.3 are denied at SW2 because of some security policy, so 1.0.0.1 can only communicate with 1.0.0.2 as a result of the flow entries.

OpenFlow switch receives a packet, the header will be parsed and matched against flow entries. If matched, the packet will be locally processed according to the entry actions. If the match fails, the switch will submit an OpenFlow PacketIn message, which conveys the headers of the packet (or the whole packet, optionally), to the controller. And usually, the controller will install some flow entries by sending OpenFlow FlowMod messages to the switch, which will account for the local processing of subsequent packets of the flow [56].

OpenFlow is flexible in datapath forwarding control, but it lacks the ability to make configurations of switches, such as ID assignment and tunnel establishment. Thus, OpenFlow Configuration Protocol (OF-CONFIG) [57] is designed as a supplement to OpenFlow, which enables remote configurations of virtual or physical OpenFlow switches. While Open-vSwitch-Database-Management-Protocol (OVSDB) [58] is specially designed to configure Open vSwitch (OVS) [59], which is the most popular virtual OpenFlow switch. Combining OpenFlow with either OF-CONFIG or OVSDB would be good enough for the management and control of OpenFlow-based networks.

B. Brief Introduction of Traditional Network Experiment Methods

Usually, simulators, emulation platforms and network testbeds are used by researchers to evaluate the performance of new network designs and algorithms. Simulators (e.g., NS2/NS3 [60]) establish the corresponding simulations based on the network topology and characteristics, with forged traffic [61]–[66]. Emulation platforms (e.g., Emulab [67]) use virtualized server clusters to simulate distributed network based on software, and real traffic could be introduced.

In addition to simulators and emulation platforms, network testbeds could provide researchers with real traffic and real network devices. There have been many large-scale network testbeds. APE [68] aims to provide real assessing environment for ad-hoc routing protocols. Planetlab [69] is an overlay testbed for broad-coverage services. Motelab [70] is a wireless sensor network testbed. UltraScienceNet [71] is set up to develop networking technologies, which serves for the next-generation large-scale scientific applications. StarBED [72] is a network testbed focusing on running actual program codes for software and hardware implementations. LISP-LAB [73] is a network testbed that focuses on LISP (Locator/ID Separation Protocol) application and open-source LISP implementation. GENI [20] is a distributed experiment environment for large scale network experiments. GpENI [74] is a programmable network testbed which provides researchers with the ability to programme the whole networking stack.

TABLE II
ADVANTAGES AND DISADVANTAGES OF NON-SDN TEST PLATFORMS.

	Advantages	Disadvantages
Simulators	Flexible, Low Cost	Forged Traffic
Emulation Platforms	Flexible, Low Cost , Real Traffic	Low Performance
Traditional Network Testbeds	Real Traffic, High Performance, Scalable	Inflexible, Insufficient Automation

These network testbeds usually depend on different types of tunnels to connect experimental networks over the Internet. Some of the traditional tunneling technologies could still be useful in SDN-based testbeds. Virtual Local Area Network (VLAN), known as 802.1q, uses a specified field to virtualize Ethernet. Q-in-Q, known as 802.1ad, is a method of VLAN stack to provide virtualization in carrier-grade Ethernet. MultiProtocol Label Switching (MPLS) [75], uses a 2.5 shim layer to forward traffic faster on Internet routers. Ethernet over MPLS (EoMPLS) [76], uses a method that accommodates Ethernet traffic over the MPLS network. Generic routing encapsulation (GRE) [77], standardizes a generic encapsulation form to route the traffic of a network over another network. Virtual eXtensible Local Area Network (VxLAN) [78], uses a User Datagram Protocol (UDP) encapsulation for the Ethernet traffic. Network virtualization using GRE (NVGRE) [79], describes the usage of the GRE header for network virtualization.

Compared with simulators, network testbeds are more convincing for network experiments, because real experiment traffic is introduced. Compared with emulation platforms, network testbeds involve hardware network devices and they could run over the product network, providing high performance and scalability. However, traditional network devices in these testbeds are too closed to control, and usually need many manual configurations, which reduce the experiment flexibility and automation. Table II shows the advantages and disadvantages of these three kinds of network experiment methods.

IV. OVERVIEW OF SDN TESTBEDS

In this section, we give an overview of SDN testbeds from three perspectives: advantages of SDN testbeds over traditional network experiment methods, design issues of large-scale SDN testbeds, and key technologies of large-scale SDN testbeds.

A. Advantages of SDN Testbeds over Traditional Network Experiment Methods

As discussed in the last section, network testbeds are more convincible for network experiments because of its high

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

performance and scalability, but they are usually lack of automation and flexibility. Fortunately, SDN could provide a promising method to change how the traditional network looks like and works. Applying SDN methods, especially OpenFlow, into network testbeds could perfectly solve the inflexible and insufficient automation problems with the following advantages:

1) Easier Setup: In non-SDN testbeds, testbed administrators need to configure many network devices separately to setup an experiment environment, which is time-consuming, tedious and prone to error. However, SDN makes all of these easier by exposing programming interfaces to administrators, and SDN-based network testbeds could finish most of the setup procedures automatically.

2) Optimal Control*:* Traditional network devices adopt the way of distributed traffic-forwarding decisions, and devices lack the knowledge of network global view, which may result in sub-optimal forwarding logic. SDN decouples the control logic and forwarding devices, making network operations and management easier. The SDN controller has the global view of the network, so SDN applications could install flows in real time to make better use of network resources.

3) Enhanced Network Virtualization: Network virtualization is important for testbeds. Compared with traditional testbeds that are usually virtualized with many manual tunnel configurations, SDN hypervisors can automatically ease the complex and error-prone operations. Running advanced algorithms in SDN hypervisors can also enhance the flexibility of testbed network virtualization.

Up to date, trials of large-scale SDN testbeds have been actively advancing in many countries [20]–[24], and encouraging improvements have been seen in these SDN testbed implementations.

B. Design Issues of Large-scale SDN Testbeds

Designing large-scale SDN testbeds involves many theoretical and practical aspects. Several issues need to be carefully considered when designing large-scale SDN testbeds. In this subsection, we discuss some design issues as follows.

1) Automation: Running experiments with real hosts and network devices in a traditional way needs many manual configurations, which could be very complex and may somehow result in the slow pace of network innovation in the past decades. Considering that SDN provides programmability of network automation, some computer virtualization technologies [80], [81] could provide interfaces for VM scheduling and provision. SDN testbeds should combine these two technologies to provide fully automated network innovation environment.

2) Flexibility: Amenability to change could sometimes be very important for network research, e.g., when a research focuses on how a newly proposed loop-free algorithm performs, it may require to be tested in different scales. SDN testbeds should provide the flexibility to allow researchers to change the number of network devices or tune the link bandwidth between devices in their experiments.

3) Scalability: As is known, SDN has many advantages over the traditional network with its centralized control. However, scalability could in turn be a problem, especially when a large-scale SDN testbed could be a nationwide or even continent wide occurrence. How to connect geographically distributed or even heterogeneous network domains, with a unified management and control, should be carefully considered.

4) Security: Security is an important issue in all IT infrastructures. As for SDN testbeds, it implies to guarantee security when different network experiments are carried out over the same infrastructure. This should mainly include the management security with Authentication, Authorization, and Accounting (AAA) mechanisms and traffic isolation with slicing technologies.

Considering all these facts, SDN testbeds could be regarded as "cloud especially for network researcher", providing compute and network resources (storage could also be integrated, if needed) to serve network research tenants that are usually referred to as "slices".

C. Key Technologies of Large-scale SDN Testbeds

For a large-scale SDN testbed, three main technologies should be carefully implemented, including management, networking, and slicing. Again, it should be pointed out that real implementations of SDN testbeds require many technologies, and this paper is not trying to cover all these technologies. However, management, networking and slicing are the key technologies needed to construct and run an SDN testbed, and efficient implementations of the three technologies would help to achieve the above basic designing views.

1) Management Technologies: It should be first pointed out that management technologies in this paper are differentiated from SDN management technologies (such as OF-CONFIG and OVSDB) that only care for network, while management technologies in SDN testbeds look at things from a different perspective, caring for virtual machines (VMs), network, and experiment running states. The whole implementation of these management technologies is usually called "control framework".

The Web portal interacts with users, and users could submit their resource applications, which describe the topology of their experiments. Underlying resources such as VMs and network devices should be scheduled and provided according to the applications. VM control depends on server virtualization technologies [80], [81] and network devices' control depend on slicing technologies (described latter), and management technologies just call their APIs to execute the experiment applications. Once the experiment has been activated, statistics should be collected and analyzed. When the experiment finishes, the relevant resource should be released.

In other words, the whole life-cycle of experiments should be supported with management technologies. In addition, AAA mechanisms for security and federated mechanisms for inter-testbeds scalability should be supported.

2) Networking Technologies: Data plane networking refers to how SDN switches connect, both physical and logical.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

In a pure single SDN domain, switches could be directly connected in copper or fiber. Sometimes, the data plane sits over the traditional network, e.g., inter-domain connections, and switches could be connected through either label-based network, such as MPLS [75] or VLAN (802.1q), or tunnel-based network, such as GRE [77].

Management/control plane networking refers to how management/control elements connect. The management element could include software that implements management technologies, and the control element could include slicing tools or experiment SDN controllers. Management/control plane networking mainly cares about access, so L3 connectivity could just be enough.

3) Slicing Technologies: To isolate experiment traffic, slicing technology, which is an alias of network virtualization in SDN testbed, must be implemented. SDN-based virtualization is usually integrated in the general SDN control platform as an application, such as [82], [83]. This method is widely accepted in data center virtualization: the application manages all the network forwarding logic to connect or isolate traffic, and data center users never need to care about how the network forwards packets. However, this could be totally different in an SDN testbed, because users in the SDN testbed may want to control their experiment network's forwarding to carry out some L2-L4 innovation, rather than only run their L7 applications on top of the network, and these users usually need their own SDN controllers. So, slicing technologies in the SDN testbed should coordinate different users' controllers to isolate users' data plane traffic, and make users' controllers have the illusion that there are no other users' networks and controllers.

In order to achieve the above targets, slicing implementations in SDN testbeds [84]–[88] should exist beyond the general SDN controller platform. They are usually based on OpenFlow and sit between physical OpenFlow switches and users' OpenFlow controllers, acting as an OpenFlow proxy. They modify most OpenFlow messages, to isolate data plane traffic with some fields as user tag, and to "treat" users' controllers as if they each own the whole testbed network.

Slicing tools in SDN testbeds could be regarded as network hypervisors, and it is such a wide-ranging technology that we could not put too many details in this paper. More details on network hypervisors can be found in [38], which provides numerous comprehensive and deep insights.

V. Different Large-scale SDN Testbeds Implementations

In this part, five typical large-scale SDN testbeds will be introduced. Each of the five SDN testbeds will be deeply introduced in terms of four aspects. "Design Objectives and Development" gives an overview of the testbed, including why it is constructed and how it evolves, in addition to stating the service availability/openness. "Key Technologies" introduces management, networking and slicing technologies. "Network Deployment" displays the implementation of each testbed. Finally, "Experiments" introduces some interesting SDN experiments that have been carried out.

A. GENI OpenFlow

1) Design Objectives and Development: GENI (Global Environment for Network Innovation) [89] is sponsored by the U.S. National Science Foundation (NSF) and uses emerging network technologies such as network virtualization, OpenFlow and SDN, to enable network innovations and at-scale experimental activities [90], [91]. GENI uses the spiral evolution development model. In the first two stages, an end-to-end working prototype was created, and it developed towards continuous experimentation. In the third spiral stage, GENI introduced the OpenFlow technology, and deployed OpenFlow switches to the experiment environment of campus network [92]. Currently GENI is in the sixth spiral stage. Major objectives of this stage include the following [93]: (1) Attracting more experimenters with better tools and services; (2) Growing scale by deploying GENI racks and by GENI-enabling campuses; and (3) Revising instrumentation and measurement systems for GENI.

OpenFlow has become a supporting technology in GENI, which enables GENI to provide users with real environment to do network experiments at scale, as well as to achieve GENI's key concepts of slices and deep programmability. In GENI, OpenFlow switches are providing deep programmability, which gives network administrators the ability to centrally define policy to manage security on the network [94]. GENI OpenFlow is still available to the public, and users could access via the GENI portal [95].

2) Key Technologies: The GENI key technologies mainly include three parts: control framework, networking technologies, and slicing technologies. In the following we will introduce the three parts in detail.

a) Management Technologies: In order to build a large scale and coherent infrastructure, GENI is built using a federated model, i.e., GENI embraces a range of resources from different providers. Each resource provider manages its own resources and provides the ability for experimenters to use their resources. GENI's Control Framework (GCF) contains four parts: GENI Meta-Operations Center (GMOC), Clearinghouse, Tools, and Aggregates [96].The relationships among these parts are shown in Figure 4 [20].

- GENI Tools are mainly used for experimenters to manage and control GENI resources. For example, resource reservation tools are used to draw experimental topologies and manage slices, measurement tools are used to measure statistics, and monitoring tools provide insight into the resource utilization, etc.
- GENI Aggregates provide resources to experimenters. Different aggregates provide different types of resources, such as compute resources, networking resources, etc.
- GMOC currently has two functions. The first function is to notice, report and escalate the GENI infrastructure for the health maintenance. The second function is to connect the experiments and the GENI operators [11].
- GENI Clearinghouse federates experimenters, GENI Aggregates and GMOC. The Clearinghouse provides the required trust, authentication, accountability, and authorization services for GENI [97].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

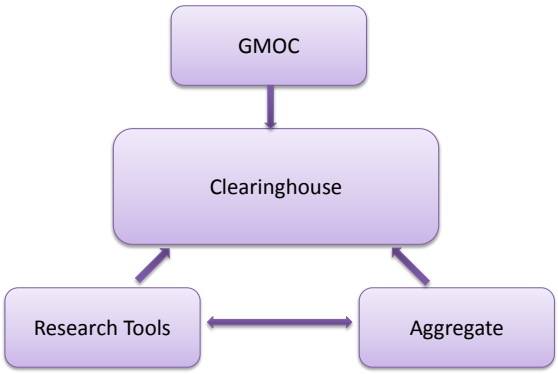

Figure 4. GENI Control Framework (GCF).

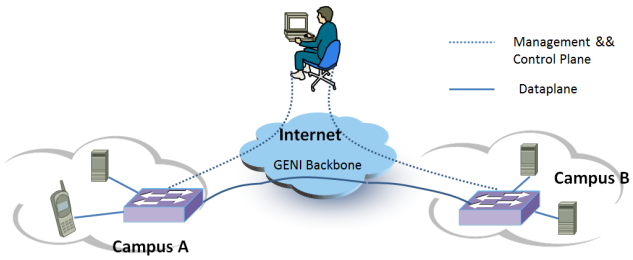

Figure 5. A simplified GENI network connection. Each campus deploys its own network and they are interconnected by the GENI backbone network. These networks can be managed and controlled remotely.

These four parts communicate with each other in an orchestrated manner. The Clearinghouse APIs define interfaces between tools/aggregate managers and the Clearinghouse [98]. The GENI Aggregate Manager APIs specify the interfaces between tools and aggregate managers, allowing aggregates to advertise resources and to allocate resources for slices in the form of slivers (A slice is a collection of resources allocated for an experiment, and each independent resource is called a sliver). The GENI AM API is an interface of the SFA (Slice-Based Federation Architecture), which is used to orchestrate resources in different testbeds to serve as a slice [99]. SFA has been implemented by PlanetLab [69], ProtoGENI [93], ExoGENI [100], InstaGENI [101] and GENI OpenFlow [92]. FlowVisor OpenFlow Aggregate Manager (FOAM) [102] is an aggregate manager for network resources, which reserves OpenFlow slivers and implement several support functions for aggregate administrators.

The lifecycle of a GENI experiment mainly includes three stages [103]: Design/Setup, Execute, and Finish. The first stage can be divided into three steps. In the initial step, users should set up a GENI account and join a GENI Project. In this step, GENI aggregates virtual machine services that have local policies. These policies determine which coordinator they trust and accept to approve users, projects, and slices and to issue various credentials. In the second step, users query the Clearinghouse for the list of aggregate managers. The Clearinghouse performs registration, deletion, and resolution of various principal objects, current slices, users, slice authorities, aggregate managers, and components. In the last step, users query the aggregate managers, which export well-defined, remotely accessible interfaces, for available resources to create slices and reserve resources. In the second stage, users could login to nodes through GENI Tools to carry out their experiments and monitor the experiments' execution. In the third stage, users should delete resources and tear down their experiments. The entire lifecycle is guaranteed by GMOC, which connects the experiments and the GENI operators.

b) Networking Technologies: In this part, we mainly introduce the networking technologies used in GENI OpenFlow. GENI supports several networking options depending on experiments' needs, existing networks, and possibilities for new connections. In Figure 5, we use the inter-campus connection as an example to show a simplified typical network connection topology [104].

As shown in Figure 5, GENI OpenFlow testbed is composed of the Control and Management Plane, and the Data Plane. The control and management plane mainly configure and control the experiments executed in the data plane. Users can access the slices through SSH connections over the Internet or higher layer tunneling [105]. The control path is usually the management traffic such as control and monitoring data.

The data plane contains users' slices and executes experiments. The GENI data plane can include both Layer 3 (L3) and Layer 2 (L2) network technologies. L2 options are usually used to connect campus, regional and national research networks. L3 options are usually the higher layer tunneling, which connect the data plane and control plane, and also used by experimenters to manage their slices.

L2 networking technologies which are used in GENI testbed mainly include the following [104]:

- Single VLAN. This is the most straightforward way. When using this method, all unique network need to negotiate a common VLAN ID.
- VLAN Translation. VLAN translation is used to solve the confliction problem when different networks use overlapping VLAN schemes. This approach allows different networks to use different VLAN IDs to participate in the same experiment. The experiment's VLAN ID is translated at the edge of each network.
- L2 Tunneling. GENI testbed often uses Q-in-Q to tunnel [106] between an agreed upon VLAN ID through an intermediate network. Both Internet2 and National Lambda Rail (NLR) GENI [107] core networks support Q-in-Q tunneling, as do most regional research networks. MPLS technology is also used in some networks.
- Fiber Connection. This method could provide higher L2 throughput and more flexibility, so GENI uses direct fiber to connect campuses and Internet2 or NLR's OpenFlow core networks.

c) Slicing Technologies: GENI adopts the concept of slice-ability from the PlanetLab testbed [69]. Slice-ability is the ability to support virtualization for simultaneous ex-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

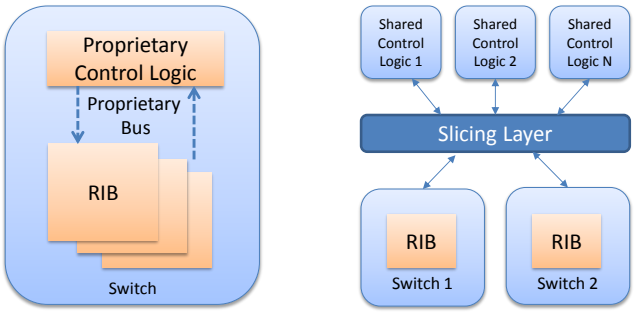

Figure 6. Traditional network device and SDN-based transparent slicing layer.

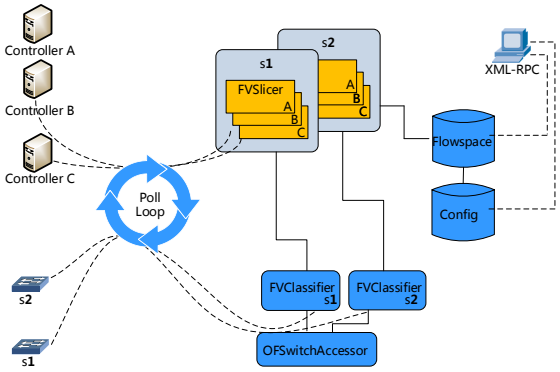

Figure 7. Architecture of FlowVisor. Each slice has the ability to control a set of flows, called flowspace and each slice has its own controller to manage the experiment network. Its main limitation is that flowspaces for different slices cannot be overlapped.

periments to share network resources. Traditional network uses VLAN to implement the basic virtualization strategy. However, in GENI, in order to provide a more flexible network virtualization approach, GENI uses the OpenFlow-enabled [6] network components, which comprise a transparent proxy between OpenFlow controllers and switches (shown in Figure 6) to allow multiple controllers to manage the same OpenFlow switches [20].

In the following paragraphs, we introduce three SDN slicing tools used in GENI: FlowVisor, OVX, and Flowspace Firewall.

- FlowVisor

FlowVisor [7] is a software platform that supports slicing where multiple experimenters get their own isolated slice of the infrastructure and control it using their own NOS and a set of control and management applications. FlowVisor, first developed at Stanford University, has been widely used in experimental Research and Education networks.

Figure 7 shows the architecture of FlowVisor's implementation. XML-RPC enables FlowVisor to provide Web services to users. Poll Loop is used to listen to the FlowVisor event circularly. FVClassifier is used to maintain connection with the physical OpenFlow switching equipment, handle I/O requests, and record OpenFlow switch information, such as ports and performance. Each FVClassifier corresponds to an OpenFlow switching equipment. OFSwitchAccessor helps FVClassfier access to physical switches and connects them with FlowVisor, and then delivers OpenFlow messages from switches to the correct slices. FVSlicer is used to maintain the connection with the controllers, manage the OpenFlow sessions and process the signals issued by the controllers. Flowspace database stores the matching rules for different slices using OpenFlow 12 tuples [56]. When a flow from one physical OpenFlow switching equipment reaches the Flowspace database, the OpenFlow message will be sent to FVSlicer according the slicing rules and then to the corresponding controller.

The core principle of FlowVisor is to map upward messages to corresponding controllers and filter downward messages to correctly forward the traffic. With FlowVisor, hardware resources would be shared among slices. These resources mainly include topology, bandwidth, device CPU, the forwarding tables. FlowVisor defines the slices using definition policy language. With FlowVisor, each slice has the ability to control a set of flows, called flowspace, and each slice has its own controller to manage the experiment network [7].

However, FlowVisor has some defects. For example, flowspaces for different slices cannot overlap, resulting in a fragmented address header space for each slice. Moreover, FlowVisor can only provide users with restrictive topology that must be a part of the physical one.

- OpenVirteX

OpenVirteX (OVX) [85] is developed by ON.Lab, which could slice a single physical SDN infrastructure into multiple vSDNs. With OVX, users can specify their experimental network topology and deploy the network OS of their choice, with the help of network embedder [9], as Figure 8 shows. First, the embedder receives information of the virtual experimental network from users, such as topology and addressing scheme. Then the embedder generates a virtual-to-physical mapping, and sends it to OVX. Next, OVX records the mapping information, and instantiates the virtual experimental network over the physical infrastructure. OVX also allows runtime reconfigurations when users want to change their experimental network.

Compared with FlowVisor, OVX recognizes slice hosts according to their access point (e.g., DPID + port ID) and assigns each slice with a unique tenant ID, which is less flexible but rather practical than the flowspace method. Furthermore, OVX provides each user with a full addressing space, by rewriting MAC addresses at the ingress switch to carry users' identifiers, and setting back to the original MAC addresses at the egress switch. If a user's controller makes another MAC rewrite action, OVX would record it and accordingly set the new MAC address at the egress edge. In addition, OVX allows tenants or users to specify their own arbitrary topology, which is not restrictive to the actual physical network.

- FlowSpace Firewall

FlowSpace Firewall (FSFW) [86] is developed by Indiana University as a plugin of Floodlight [108] to provide network virtualization function. Compared with other slicing tools, FSFW concerns about the consistency between different users'

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

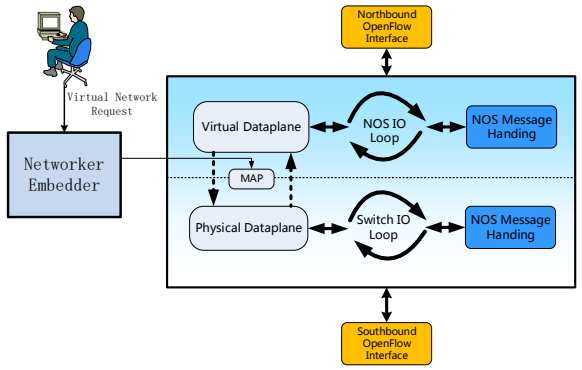

Figure 8. Architecture of OpenVirteX. OVX could provide each user with overlapping addresses, by rewriting MAC addresses at the ingress switch to carry users' identifier, and setting back to the original MAC addresses at the egress switch.

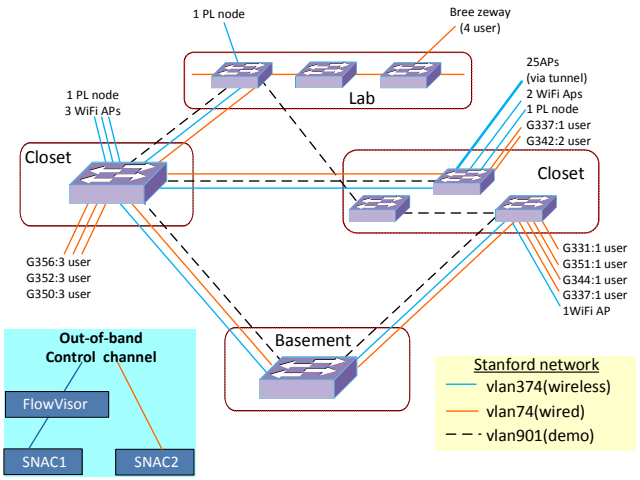

Figure 9. The OpenFlow network deployed at Stanford [109].

controllers: if a controller sends a rule that could result in data plane contradictions, FSFW could just deny the rule and send an error back to the controller. Because of this special role, FSFW could just sit between physical OF switches and other slicing tools.

FSFW has several primary classes: VLANSlicer, proxy, FlowSpaceFirewall, ControllerConnector, FlowStatCache and FlowStatCacher. The VLANSlicer class provides the capabilities of VLAN/Port Slicing. The proxy class is the router between switch and controller. The FlowSpaceFirewall class is responsible for listening to all events from Floodlight and sending them to the slices. The ControllerConnector class is basically a timer that contains a HashMap (dictionary) of Switch ID to a list of Proxies in order to continually try and connect to controllers. The FlowStatCache and FlowStatCacher classes work together to store the most recent FlowStats from the devices.

There is limited scope for FSFW: it only supports slicing on VLAN/interface, which is not flexible enough, and VLAN-based slicing means that users cannot use VLAN field for OpenFlow operations because it is specially used to isolate users' traffic. However, this in turn allows for simpler and faster slicing of requests.

3) Network Deployment: In this part, we introduce the GENI network deployment from two aspects: campus networks and national backbone network. By merging multiple campuses and backbone resources, researchers can run experiments at a larger scale.

a) Campus Networks Deployment: The experimental OpenFlow network was first deployed in campus and it has proven to be a very appealing trial. Some of the universities includes Princeton, Stanford, Clemson, Georgia Tech, etc. Figure 9 shows the topology of the OpenFlow-based network at Stanford. Different OpenFlow network has been deployed: production, experimental and demonstration network [109], with the VLAN-based slicing implemented by FlowVisor.

b) National Backbone Network Deployment: At a larger scale, the core of GENI OpenFlow consists of some interconnected OpenFlow enabled switches on Internet2 [110] and NLR [107] network. Figure 10 shows the details of backbone integration [111]. The connection between the OpenFlow core network and the NLR network is implemented with HP6600 switches deployed in Sunnyvale, Seattle, Denver, Chicago, and Atlanta and as well as the NetFPGA switches deployed in Sunnyvale, Houston, Chicago, and New York [112]. The connection to the Internet2 network is through OpenFlow enabled switches which are installed in Los Angeles, New York, Washington DC, and Atlanta [113].

There are currently three standing backbone interconnected VLANs (3715, 3716, and 3717) carried on the five switches in the core network, which are located in the five cities shown on the maps below. VLAN 3715 and 3716 are topologically a broken ring (to help prevent accidental loops). The third VLAN 3717 is a ring. The gap in each of the broken-ring VLANs is between two different links, providing either a longer or shorter path through the core between two switches. These three VLANs are not interconnected in the core network, and should not extend beyond the backbone providers into regions or campuses.

4) Experiments: The experiments mainly include wireless and multi layer SDN. Here we describe the two experiments in detail.

a) Wireless: In recent years, Clemson University has designed a system to provide GENI with the ability to enable experimenters to use wireless resources. GENI could use the system to provide IPv4 mobility across heterogeneous wireless networks. The system framework could be used to enable researchers to test network applications and different handover algorithms, in addition to eliminating the triangle routing problem.

Figure 11 shows the system's general architecture, and how it is integrated within GENI. The root OpenFlow switch acts as the system's ingress/egress from/to the out side network, and other OpenFlow switches are responsible to detect migration events and route client packets. The Floodlight controller [108] is designed to manage client IP addresses globally with the DHCP module, and to handle migration events with the Mobility module. The client-level component works on

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

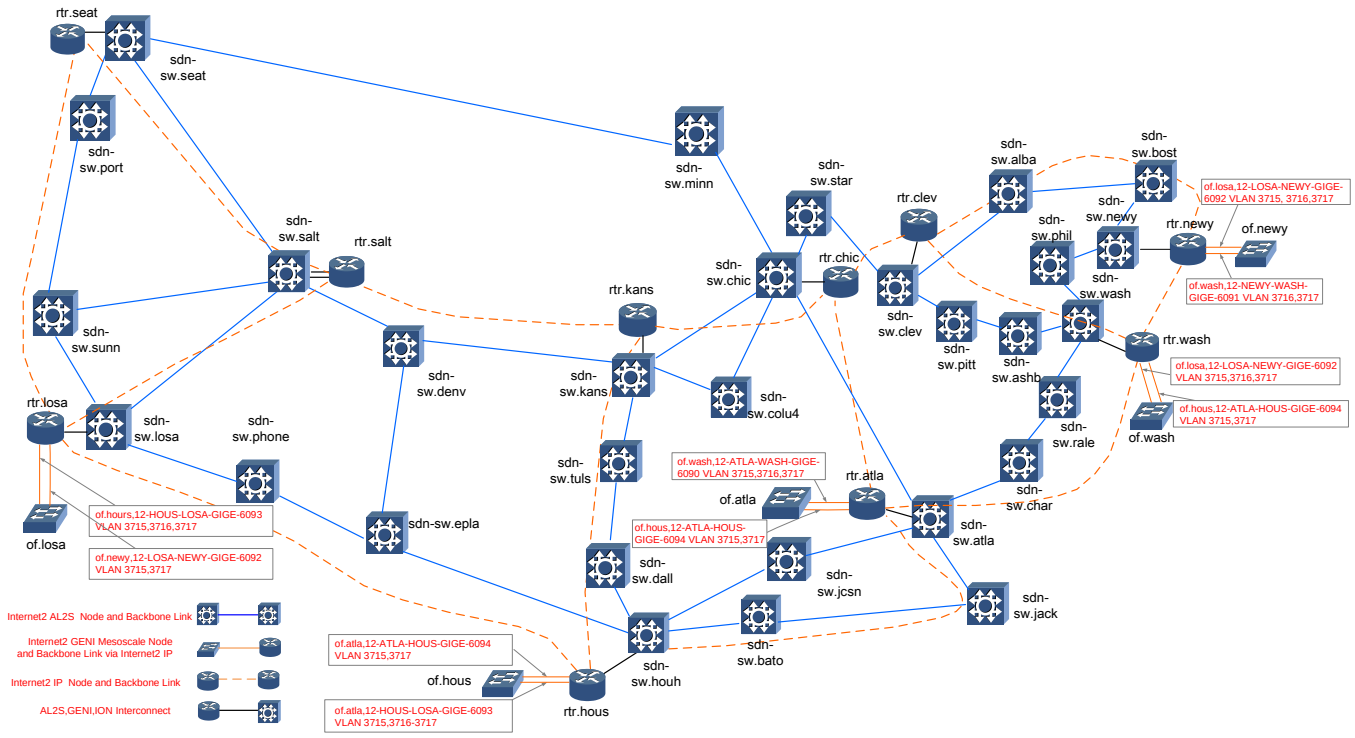

Figure 10. Backbone integration of GENI OpenFlow [111]. VLANs 3715, 3716, and 3717 are carried on the five switches in the core network, and these VLANs should not extend beyond the backbone providers into regions or campuses.

mobile endpoint, and it chooses different physical interfaces in different environment with a handover decision [17]. While the interface choice is transparent to the applications through a virtual network interface (VNI), which is installed on the client as an adaptor of different physical interfaces. The client also has its own Floodlight controller, which routes packets from the virtual network interface to the chosen physical interface, through a client built-in OVS [59].

b) Multi layer: In 2012, Infinera first demonstrated an SDN Open Transport Switch (OTS) prototype in the Long Island Metropolitan Area Network (LIMAN) testbed [116]. The demonstration extends OpenFlow with the dynamic optical transporting ability to enable application-driven optical transport bandwidth services, such as the converged wavelength technologies. Based on the OTS, Infinera, Brocade, and the U.S. Department of Energy's Energy Sciences Network (ESnet) have successfully demonstrated a multi layer networking using SDN technologies in October 2013 [116].

Figure 12 shows the architecture of the system. On-demand Secure Circuits and Reservation System (OSCARS) [117] is an open source implementation developed by ES-net to provide circuit network control, and the SDN controller is the standard Floodlight [108]. The data plane consists of some Brocade MLXe routers and other OpenFlow enabled switches in the packet layer, as well as some Infinera DTN-X platforms in the optical layer. The DTN-X platforms are installed with the OTS software module, so that OpenFlow control could be provisioned over both packet and optical layers by OSCARS combined with Floodlight.

In order to provide bandwidth on demand (BOD) across the packet and optical layers, OSCARS-TE has been developed as a multi layer traffic engineering application within Floodlight. It collects information via the optical version of standard OpenFlow (OTS) [118], and accordingly OSCARS-TE could be triggered to re-route some important flows from congested packet paths to optical light paths dynamically, which could be very useful in data center WAN interconnection scenario.

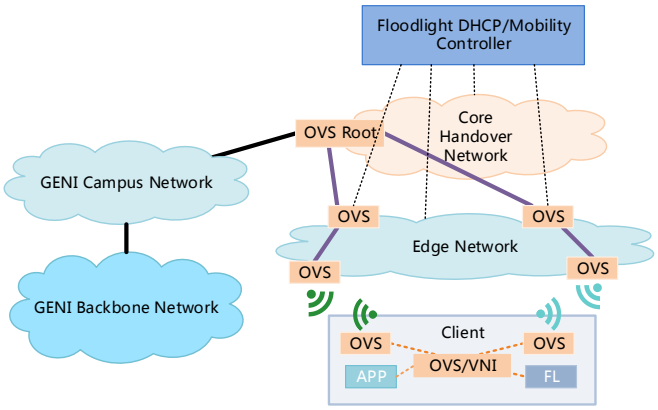

Figure 11. Wireless integration in GENI OpenFlow. Root OpenFlow switch plays as the system's ingress/egress. Other OpenFlow switches are used to route client packets and detect migration events. Mobility management has been studied extensively in recent years [2], [114], [115]. The Floodlight controller manages client IP addresses globally and handles migration events. The mobile client switches different physical interfaces with some handover algorithms.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

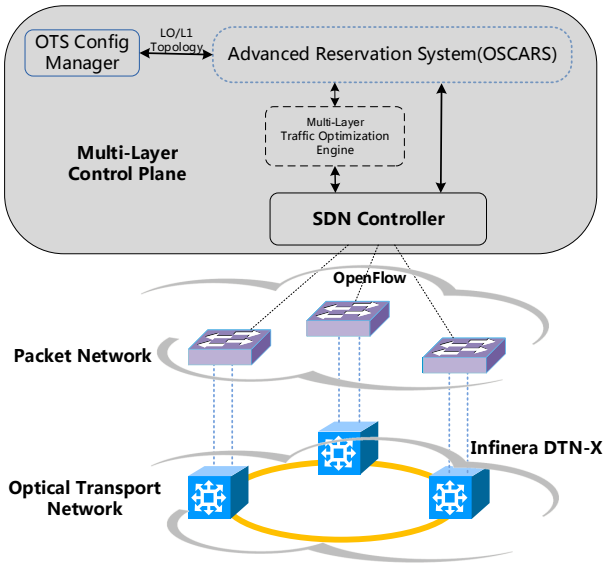

Figure 12. Multi Layer demonstration in GENI OpenFlow. OpenFlow control could be provisioned over both packet and optical layers by OSCARS combined with Floodlight. A TE application is developed for BOD to reroute traffic.

B. OFELIA

1) Design Objectives and Development: OFELIA was launched by European Seventh Framework Programme (FP7) in the autumn of 2010, and its main objective is to set up a multi layer, multi domain and geographically distributed Future Internet testbed facility [21], which could accommodate different types of traffic, both for daily use and for new network experiments [21] based on OpenFlow. In order to make these goals, the designing of OFELIA conforms to the following principles [119]:

a) Flexibility and Modularity: Pre-defined functions should be limited and development of modules should be extensible.

b) Fidelity use and Easy Use: Production traffic should be accommodated and OAM should be simplified.

c) Resource isolation and Security: Virtualization should be implemented to isolate different traffic and AAA is needed.

d) Island autonomy and Federation: Management of different islands should be administratively separated, and federation among islands should be supported.

The first duration of OFELIA was scheduled into three phases. In the first phase, OFELIA succeeded in setting up the underlying facilities, including some OpenFlow switches and VM instances. In the second phase, OFELIA implemented the interconnection among different islands, and extended SDN experiments in the wireless and optical domain. And then in the third phase, the OFELIA control framework was evolved, providing connections to external facilities of the Internet and other testbeds. OFELIA access is no longer given, and it is given only in the frame of EU-limited open calls.

2) Key Technologies: In this section, we present the key technologies used in OFELIA, including the management technologies, the networking technologies, the slicing technologies, and finally the Hardware Abstraction Layer (HAL) method developed during the project.

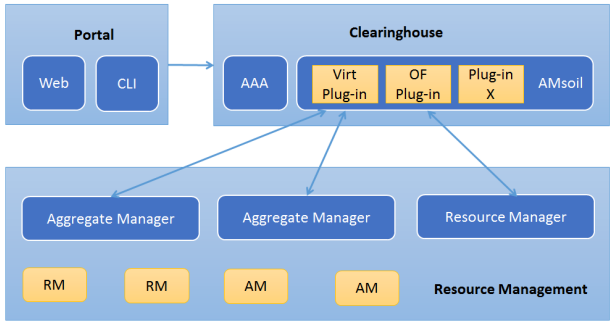

Figure 13. OFELIA Control Framework (OCF).

a) Management Technologies: To enable operators' management of the underlying facilities and users' access to their experiments, OFELIA develops the OFELIA Control Framework (OCF) [12] as the orchestrator. OCF mainly aims to arbitrate the resources allocation, automate testbed management, and simplify the lifecycle of experiments. Figure 13 shows the OCF architecture design, which conforms to the Slice-based Federation Architecture (SFA) for federation with other testbeds. Three layers are defined in the architecture: Portal, Clearinghouse and Resource Management. Interfaces and interactions among these layers are also clearly defined.

- Portal and Clearinghouse

The Portal of OCF is developed to deal with users' operations of their slices and ease management for administrators. Clearinghouse is responsible for synchronizing users' accounts and privileges among islands, and it provides unified authentication with the Lightweight Directory Access Protocol (LDAP) server. Clearinghouse design is mainly based on Expedient [12], an open source centralized control framework. As a Django-based Web application, Expedient follows a modular approach, so Clearinghouse's function could be extended by developing and driving new plug-ins in AMsoil, a pluggable light-weight framework for creating and managing testbed resources [120]. These plugins use the native interfaces, follow the same APIs and conform to the same AAA frameworks.

- Resource Management

The basic modules of Resource Management are Virtualization Aggregate Manager and OpenFlow Aggregate Manager. Virtualization Aggregate Manager, AM for short, binds Resource's Management interfaces to Clearinghouse and responsible for aggregating virtual machines for slices. OpenFlow Aggregate Manager, an OpenFlow tool based on Opt-in Manager [121], is responsible for the management of network slicing rules and configurations.

Apart from these two basic modules, there are some other useful plugins. VT Planner [8], a module that implements an efficient path computing algorithm and maps virtual links to real switches or links to support arbitrary network slicing, is integrated in AMsoil. BOWL [122] is a module primarily based on OpenWRT [123], and with BOWL plugged in, OCF

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

could support SDN controlled access for wireless Internet staff.

The original design method of OCF is derived from the use-cases and experiences in GENI. The lifecycle of an OFELIA experiment mainly includes three stages: Configure, Execute, and Finish. The first stage can be divided into three steps. In the initial step, users should configure their experiments through the Web-based user interface, which is supported by the Portal of OCF. In the second step, users request Clearinghouse for their privileges. Clearinghouse ensures the synchronization of user accounts and privileges among islands, and it provides unified authentication. In the last step, users query Resource Manager for available resources to create slices and reserve resources. In this step, Virtualization Aggregate Manager binds the resource management interfaces to Clearinghouse aggregates virtual machines for slices. OpenFlow Aggregate Manager manages network slicing rules and configurations. In the second stage, users could login to the facility to do their experiments. In the third stage, users should delete and release resources. The control framework supports monitoring mechanisms for both the control framework components and the slice resources in the entire lifecycle.

b) Networking Technologies: In addition to the network for experiment, OFELIA also has two out-of-band networks for control and management. All the OFELIA islands (sub-testbeds in OFELIA) are currently interconnected to the OFELIA hub in a star topology. The OFELIA Hub is located in Belgium and acts as the transfer station to relay both experimental and control traffic between different islands [21].

- Experimental Network

The experimental network spans all the OFELIA islands. The network connects as a pan-European single L2 segment, enabled with OpenFlow v1.0 [56], allowing experiments across different islands or countries. The experimental network has the central hub connecting all the remaining of the networks in iMinds, Belgium. More than 25 OpenFlow-enabled switches from different vendors are running, as presented in the left panel of Figure 14. Servers in the experimental network are in general connected to more than one switch, creating a more stable environment for experiments. Interestingly, loops could be intelligently avoided in the experimental network. When experimenters use switches to make loops by choosing a loop inducing flowspace via the control framework, OCF will warn the user that the requested flowspace is making loops, either locally in the island or across different islands.

Experimental network interconnection is implemented by 1 Gbit/s L2 dedicated circuits via the GEANT backbone network [124], and L2 tunnels bridged using OVS [59] over the public Internet as a redundancy. Circuits and tunnels are deployed in mesh topology for redundancy in both control and experimental networks [21].

- Control and Management Network

The control network in OFELIA (right in Figure 14) is an out-of-band network interconnecting different OFELIA islands. The network is IPv4 OSPF routed with a private address scheme, assigning each island with a subnet. Each island has its gateway to perform routing advertisement and process other routing information between the control network subnets. Moreover, all the islands connect to the OFELIA hub and some of them make a redundancy for a more stable control network. These connections are implemented via either a dedicated GEANT circuit or a L3 VPN tunnel over the Internet. Each island provides some automation services, such as LDAP and DNS through the control network. OFELIA also provides users with remote access to the control network and the server instances that are connected to the experimental network via VPN point (OpenVPN) [21].

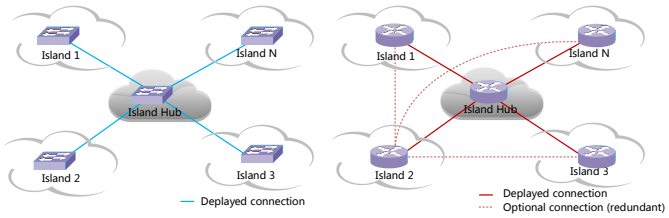

Figure 14. A sketch of experimental and control network in OFELIA [21].

The management network is used to monitor and administer projects and slices over the underlying infrastructure. In fact, the management network is optional, some islands just make these managements over the control network.

c) Slicing Technologies: FlowVisor could be used to slice the physical network in OFELIA. However, there are many limitations in FlowVisor, so some researchers under OFELIA project propose VeRTIGO to solve some of the limitations. In addition, slicing method in the optical domain is also in progress.

- VeRTIGO

As mentioned in another study [21], one of FlowVisor's limitations is that it can just establish virtual logical topologies with restrictions to the real physical topology. To overcome this shortcoming, a new slicing tool has been developed within OFELIA named VeRTIGO (ViRtual TopologIes Generalization in OpenFlow networks) [87]. Figure 15 presents the architecture of VeRTIGO, and the interaction among different components in VeRTIGO during the communication between OpenFlow controller and switches. The Web UI receives users' slice requests, and the VT Planner [125] implements an efficient path computing algorithm to embed these requests into the physical resources, which is fundamentally required by other modules. The Storage is a database to record VT configurations and flow statistics. After the slice is scheduled, OpenFlow channels would be virtualized by the following four modules: The Classifier classifies Switch-to-Controller OpenFlow messages according to traffic's header fields. The Node Virtualizer instantiates an abstract switch node for user if needed, providing VeRTIGO with the ability to virtualize several physical switches into a single logical switch. The Port Mapper replaces the port number in OpenFlow messages with the corresponding number as the virtual topology configuration schedules. And the Internal Controller is responsible for distributing flow entries over virtual links.

VeRTIGO has been integrated into OCF, and experimenters are able to setup topologies with any pattern to run their

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

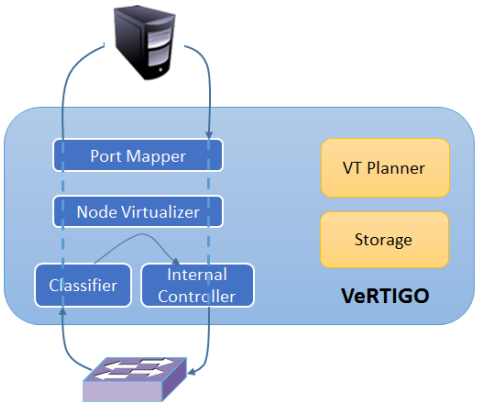

Figure 15. Architecture of VeRTIGO.

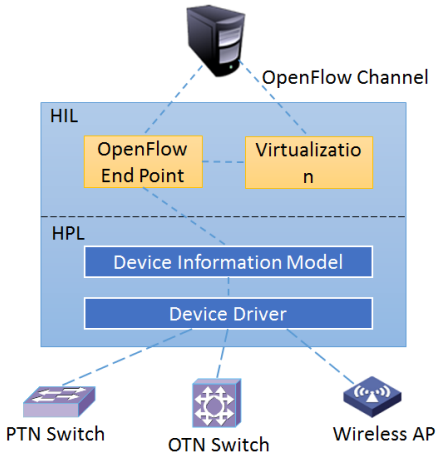

Figure 16. Architecture of HAL.

desired experiment. They can either create virtual nodes or links on available network elements manually, or sketch their desiring topology through the OCF Portal, and then they may just leave VeRTIGO to select the best way to accommodate their experimental network over the available resources.

- Optical FlowVisor

With OpenFlow circuit extensions already in place [88], research on slicing optical OpenFlow-enabled devices has been carried out. OFELIA follows these researches by extending FlowVisor. The Optical FlowVisor has visibility in both packet and optical domains, and some other optical fields are specified so that the experimenters could slice in the circuit domain based on either ports or wavelengths [126].

d) Hardware Layer Abstraction: As SDN becomes commonly widely accepted, new networking devices are separating their control and forwarding elements. Enabling OpenFlow on non-OpenFlow devices motivates the assumptions for Hardware Abstraction Layer (HAL) [19]. HAL is generated from the Alien project, which is a slice running on OFELIA infrastructure mainly aiming to integrate non-OpenFlow hardware platforms into the OFELIA OpenFlow-based infrastructure and to control and manage heterogeneous networking devices uniformly. HAL works to hide different technologies and vendor specific features from OpenFlow controllers, and its framework is shown in Figure 16. There are two separated layers in HAL: the Hardware Interface Layer (HIL) and the Hardware Presentation Layer (HPL).

- HIL

Components in HIL work together as common device control and management protocols, such as OpenFlow or other southbound SDN protocols (e.g., SNMP), and they are hidden from the hardware platform complexity. HIL components sit between the controllers and forwarding elements as proxy, consisting of the following two sub-components: the Virtualization is responsible for providing virtualization capabilities to the HAL compatible devices; The OpenFlow Endpoint ends the path of OpenFlow, maintaining the connection with upper controller and sending messages with OpenFlow-like tables downward to HPL.

- HPL

HPL components are dissimilar in different kinds of devices. Generally, the northbound API of HPL receives extended OpenFlow-like forwarding tables from HIL to control forwarding. The Device Information Model represents the state of OpenFlow switches which is independent of the underlying platform hardware. The Device Driver performs data processing with help of real hardware. It is platform dependent, translating all the OpenFlow tables into platform specific languages, and orchestrating to simulate behaviors of an OpenFlow switch.

3) Network Deployment: OFELIA started from September, 2010. Initially, it was set up with five academic testbeds each at iMinds, University. of Bristol, ETHZ, i2CAT and TUB. And now OFELIA consists of 10 islands. Figure 17 shows the specialties and capacities of these islands, and Table III shows some of the equipment deployed in several islands. The first duration lasted for 3 years. After that, OFELIA became a real experimental networking substrate, which is programmable, scalable and protocol agnostic, allowing quick deployment for new networking methods [127]. OFELIA is now open and operational as a best-effort service provider [21]. Based on OpenFlow, OFELIA allows for experimentations on multi layer and multi domain networks, providing an environment to create innovations for the future Internet.

4) Experiments: Many remarkable experiments have been carried out over OFELIA. In this section, we introduce an optical integration method and an Information-Centric Networking (ICN) supporting method in SDN network.

a) Optical Integration: Optical devices are important with their high bandwidth in transport network, and there have been several OpenFlow trials on optical integration. Compared with packet domain OpenFlow switches, the cross-connection table in optical switches cannot support flows lookup, because there is no buffering space in these switches and the optical ports have no visibility into the packets. Figure 18 shows one of the proposed OpenFlow-GMPLS collaborated architecture [18], in which OpenFlow controller is well integrated with

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

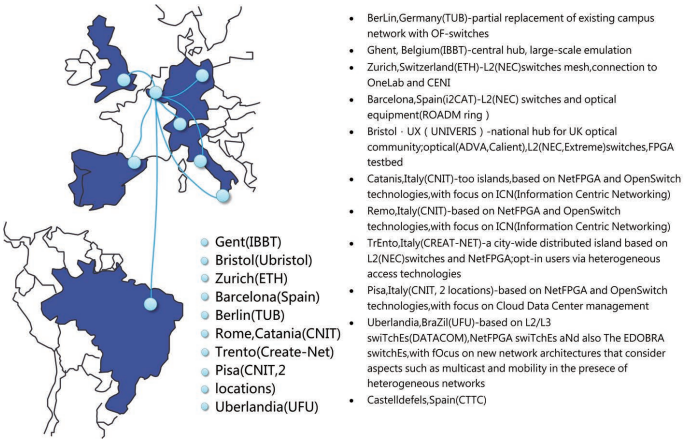

Figure 17. Specialties and capabilities of each islands in OFELIA [128].

TABLE III
EQUIPMENT DEPLOYMENT IN OFELIA [127].

island	OF-capable Ethernet Switches	Servers	NetFPGA cards, optical, wireless
i2cat	5×NEC IP8800/S3640-24T2XW,3×HP E3500-48G-PoE+yI	5×SuperMicro SYS-6010T-T	
IBBT	1NEC IP8800/S3640-48T2XVV-LWw/XFP	Virtual Wall(100 server Emulab instance)	WiLab facility,10 NetFPGA cards
UBristol	4×NEC, 3×Extreme Networks,3×AVDA FSP 3000 ROADMs, Calient optical switch	5×Dell PowerEdge servers	Ultra HD video streaming, 10TB storage, 2×Virtex-4 FPGA boards
ETHZ	3×OpenFlow switches NEC IP8800/S360-24T2XW with two optical 10G Base interface	3 servers w/36 GByte RAM	
TUB	5×NEC IP8800/S3640-48TWLW	3 servers	2×NetFPGA, BOWL testbed
Create-Net	3×NEC, 2×HP ProCurve 3500	5 servers	4×NetFPGA cards

the traditional optical forwarding intelligence. The GMPLS control plane is responsible for setting up lightpaths between optical switches, and the interconnections between optical domains and the packet domain would be set up by the edge OpenFlow switch, which is extended with tunable WDM interfaces. Initially, the first packet enters the ingress edge switch in the packet domain when the switch has no knowledge of how to deal with the packet. Then the switch submits the packet to the optical-extended controller. The controller identifies the egress edge switch of the lightpath according to the destination field, and accordingly requests the GMPLS control plane, via the OpenFlow GateWay (OFGW) module, to set up a lightpath. Once the lightpath was set up, the controller will immediately update the flow table of the two edge switches to cooperate packet and optical domains for end-to-end communication.

In addition to this method, there is another pure OpenFlow solution. A new OpenFlow message CFLOW-MOD [88] is specially proposed for the controller to update optical flow tables, rather than combining with the control and management of GMPLS. The flow setup also depends on an extended

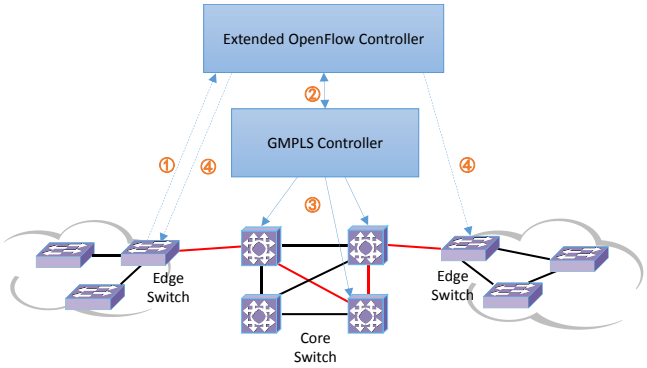

Figure 18. OpenFlow-GMPLS integration in OFELIA. GMPLS is responsible for path calculation and set up in the optical domain. The extended OpenFlow Controller could control forwarding in the packet domain, and it interacts with GMPLS with a gateway module to set up an end-to-end path.

OpenFlow agent on each optical-capable nodes in the network, and the abstraction for these nodes is implemented using the Simple Network Management Protocol (SNMP). The agent translates the CFLOW-MOD messages from the extended-controller, and accordingly sets up or destroys cross-connections via SNMP [21]. Both the methods encourage deep work on Software Defined Optical Network (SDON).

b) ICN Support: The basic idea of ICN [129] is to re-design the network to support content based networking primitives, to identify the content with proper name and to route accordingly between ICN nodes [130]. In OFELIA, there are two ICN supporting methods: the "short term" one uses existing OpenFlow standard to realize functionality in a compromise way. And correspondingly, the "long term" solution depends on special ICN capable switches with an extended OpenFlow interface.

The "short term" solution [131] supports ICN over standard OpenFlow v1.0, so that the equipment deployed in OFELIA could be reserved and content-based routing could be implemented. The content tags, which are referred hereafter as ICN-ID, are calculated uniquely for each content traffic, and these ICN-IDs would be stored in the UDP source and destination fields considering the 12-tuples. In general, the border nodes are responsible for pushing and popping such tags, and the core ICN nodes route the content traffic accordingly. This experiment has been deployed and run in the Barcelona island, and it is concisely shown in the left part of Figure 19.

For the "long term" solution [132], the ICN routing intelligence is decoupled from the forwarding and caching functions. The proposal architecture of this solution is depicted in the right of Figure 19. The data plane consists of ICN forwarding nodes, servers and hosts, and the Name Routing System (NRS) nodes play together as the control plane. A set of extensive OpenFlow controllers are implemented for the NRS intelligence, and interact with the data plane using an extensive OpenFlow interface. The extension of OpenFlow for "long term" ICN solutions can be implemented in two methods. The first method is to have an IPv4 Option field specified for ICN functions, if it is supported in OpenFlow. The second method

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

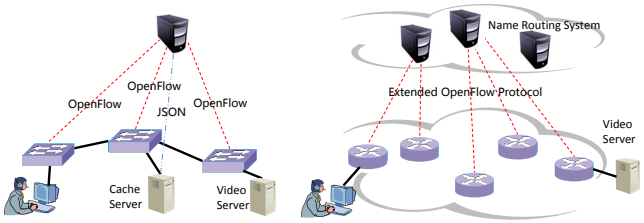

Figure 19. OpenFlow-based ICN solutions in OFELIA. The left part shows the "short term" method that uses standard OpenFlow fields, such as UDP port, to carry content information. The right part shows the "long term" method that builds a stronger ICN control plane with ICN-extended OpenFlow.

wishes to extend the OpenFlow interfaces with special ICN fields to support more content-related conceptions, such as key management and caching.

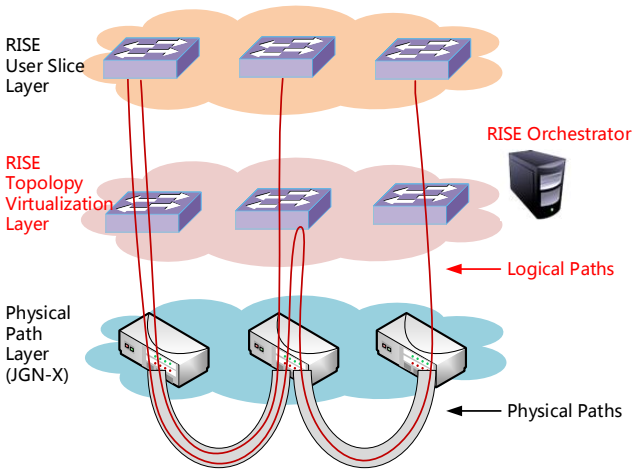

Figure 20. Architecture of RISE 3.0 [137].

C. RISE

1) Design Objectives and Development: Since 2009, National Institute of Information and Communications Technology (NICT) has been developing the Research Infrastructure for large-Scale network Experiments (RISE) [22] on top of JGN-X [133]. RISE is OpenFlow-based SDN testbed, and its objective is to provide a multi-tenancy and large-scale environment for network researchers to test SDN experiments. RISE is still open to the public, and authorized users could draw his/her requirements for experiments [134] to get the RISE resources. Up to now, RISE has gone through three versions [135].

a) RISE 1.0: The first version started from 2009. Three significant features are as follows. 1) Single user occupied the whole OpenFlow network. 2) Q-in-Q [136] was used to deliver user packets between OpenFlow switches. 3) Didn't not provide virtual machines.

b) RISE 2.0: This version was provided to the public from November 2011. Three major improvements were made in comparison with RISE 1.0. 1) Multiple users were allowed to share the physical infrastructure. 2) Pseudo wire technology was used between sites interconnection instead of Q-in-Q. 3) VMs could be provided to users.

c) RISE 3.0: In RISE 2.0, there were the following two problems: 1) Poor capacity in terms of the number of concurrent users and 2) Inflexible topology of underlay networks. To solve those issues, NICT designed layered OpenFlow networks in RISE in 2014, including User Slice Layer, Topology Virtualization Layer and Physical Path Layer, as shown in Figure 20, which is referred as RISE 3.0.

2) Key Technologies: Specialized wide-area communication lines between the OpenFlow Switches (OFSs) require vast expense, so RISE is constructed as an overlay network on top of JGN-X. In Figure 20, the Physical Path Layer is composed of JGN-X existing switches distributed nationwide, and the networking technology in RISE just involves about how to share the dedicated lines among other JGN-X services [135]. The Topology Virtualization Layer focuses on how to slice RISE's network to serve different users. RISE Orchestrator is responsible for managing RISE's underlying resources and experiments.

a) Management Technologies: RISE wishes to provide customizable SDN experiment at user's request, and automate operation is required to build these experiments. RISE Orchestrator [138] is developed in RISE 3.0, and it helps RISE administrators to manage the network and users' experiments.

In RISE, the Frontend communicates with users through Web browser, using RESTful API to receive experiment configurations and show the running status of experiments. The Project Manager manages all the underlying resources and upper experiments, and it directs the relevant manager modules to offer virtual servers and virtual switches. RISE Orchestrator knows all the topology information of the physical network by sending and receiving probes as well as the users' preconfigured experiment network, and this information is needed to manage flow entries in the physical OpenFlow switches. RISE Orchestrator is friendly for users to customize their experiments and the configuration time of a slice is much reduced to less than 10 minutes [137].

The lifecycle of a RISE experiment mainly includes three stages: Setup, Execute and Finish. In the first stage, users setup their experiments through the Frontend. Next, the Project Manager directs the VM Manager and OFS Manager to offer VMs and experimental network to users. In the second stage, users could login to nodes through the Frontend to do their experiments. In the third stage, users delete and release resources after their experiments finish.

b) Networking Technologies: JGN-X is a VLAN-based network testbed, and RISE initially utilized Q-in-Q to run overlay of JGN-X. However, there were some problems while using Q-in-Q. It has been believed that EoMPLS technologies [139] are the most suitable for the accommodation of OpenFlow Networks in JGN-X since RISE 2.0.

- Q-in-Q

In Q-in-Q tunneling [106], the inner tag is called the Customer VLAN (C-VLAN) which is used locally and transparent to service provider network, the outer tag is called the

Selective VLAN (S-VLAN) which uniquely identifies each customer on the service provider network. RISE 1.0 used Q-in-Q tunnels, when a packet with a C-VLAN enters the JGN-X network, an S-VLAN will be added at the ingress device to traverse the JGN-X network, and removed at the egress device.

When Q-in-Q is utilized, RISE user's MAC addresses have to be exposed to the JGN-X switches for forwarding. Thus, these switches may suffer a shortage of MAC address table entries when the user's network connects a huge number of devices. And Q-in-Q multicast traffic from users is also illegal for the JGN-X Ethernet switches [135].

- EoMPLS

MPLS [140] is a high-performance telecommunication networking mechanism, which could encapsulate packets of various network protocols. To configure Ethernet tunnels with MPLS, EoMPLS [139] could be utilized, which maintains the mappings between Ethernet VLAN tag and MPLS label for tunneling on the boundary of the service provider network.

Since RISE 2.0, Pseudo Wire (PW) technology that employs EoMPLS has been utilized to replace Q-in-Q, and JGN-X switches will not be involved with the shortage of MAC address table entries because the forwarding will totally depend on MPLS labels rather than MAC addresses. Besides, an MPLS label is much longer than VLAN tag, which can be more scalable in RISE's networking.

c) Slicing Technologies: In RISE 1.0, once a user requests to experiment on RISE, he or she will occupy the whole OpenFlow resources until the experiment is completed. In other words, RISE 1.0 is time-based "sliced", and it does not conform to the initial design objective of multi-tenancy. In RISE 2.0, Virtual Switch Instance (VSI) is implemented to offer at the most 16 different slices at the same time. User VMs are required to send packets with specified VLAN tags to isolate from other slice traffic, so the VLAN field must be hidden from user's OpenFlow controller. Although RISE 2.0 could somehow slice the network, it is far from real multi-tenancy – 16 slices are not scalable and this slicing method is not very flexible because the topology of each slice is tightly bound to that of JGN-X [137].

In RISE 3.0, the Topology Virtualization Layer is inserted between the RISE network and JGN-X network for more flexible slicing methods. It implements the "logical path" [135] to stitch physical paths in JGN-X network by MAC address rewriting, as shown in Figure 21, decoupling the user OpenFlow network topology from the JGN-X topology. Based on this method, slicing scalability is improved two to three times up to ¿50 slices. However, there are still some demerits using MAC address rewriting. For example, address rewriting will bring some inconvenience for trouble shooting, and more overhead will be introduced with packets rewriting.

3) Network Deployment: Up to now, RISE has been deployed at 10 sites with partially connected mesh topology in Japan, and three sites overseas (Los Angeles, Bangkok and Singapore), and they are the so-called EVNs, as shown in Figure 22 [137]. Upon the EVNs are the OFNs and OFSs. All of the OFNs are logical (virtual) networks, which are built over the EVNs. The site in Tokyo has connections with the sites in Los Angeles, Bangkok and Singapore. Users can

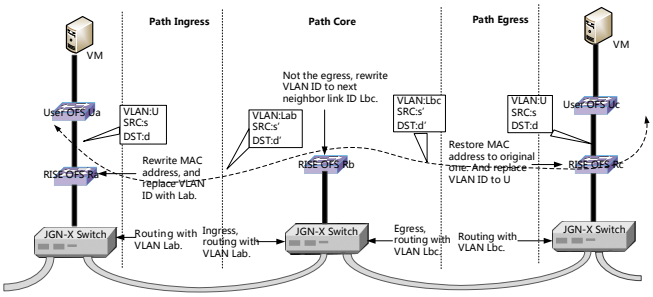

Figure 21. Logical Path implemented in RISE 3.0 [141]. Rewritten MAC addresses to contain information of the logical path, OFs forwarding depends on new MAC address. Rewritten VLAN tags contain physical path information, JGN-X switches depend on new VLAN to route packets.

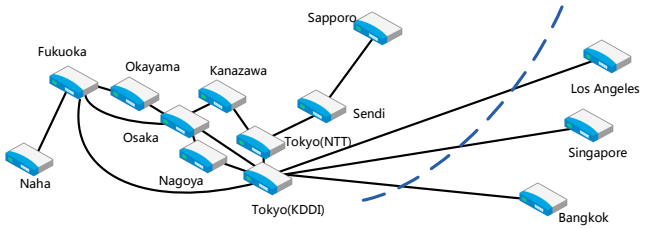

Figure 22. Network deployment of RISE [137].

still ask for various slice topologies by utilizing the Topology Virtualization Layer to slice the RISE network, although the physical deployment is not a full mesh topology.

4) Experiments: A variety of experiments have been implemented on RISE. Here, we introduce the video streaming experiments of the Sapporo Snow Festival [142]–[144] and the optical scenarios [145].

a) Video Streaming Experiments in Sapporo Snow Festival: The following introduces video streaming experiments during Sapporo Snow Festival of several years.

- Experiment in 2010 Sapporo Snow Festival

In the 2010 Sapporo Snow Festival, a high-quality and timely video streaming application was demonstrated over RISE 2.0, which was designed to confirm the performance and reliability of OpenFlow in large-scale network. In this experiment, NICT utilized a NEC-based OpenFlow controller, which could demonstrate the logical topology as well as network flows.

Results of the experiment indicated that the video transmission satisfied unicast streaming. However, in multicast streaming, packet loss was measured. Thus, other experiments explored the reasons of loss. Moreover, video streaming and transmission visualized in OpenFlow network showed more details such as real-time dynamic motion.

Operator configuration of both normal L2 switches and OpenFlow functions increased the operation cost of an OpenFlow-based network. The mismatch between traditional L2 switches and OpenFlow switches led to problems on the network. Thus, NICT considered that OpenFlow could not decrease the manipulation expenses when established on an R&D network.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

- Experiment in 2013 Sapporo Snow Festival

NICT implemented a broadcast experiment to transmit the festival video stream on several SDN networks in 2013. The experiment transmitted HD video streaming among various nodes both in Japan and foreign countries, evaluating different operational functions in large-scale network. In this time, the quality of video was improved to 4K HD resolution. In consideration of developed operation in the future, measurement and analysis of network status was also included in the experiment. Cooperating with Philippine and Singapore, NICT experimented VOD and 4K live streaming. Video stream transmitting experiments on partial-region WiFi and broadcast for smart phones with different operating systems were also conducted. Besides, transferring video streaming in an ultra-speed of 100Gbps was tested among festival scenes and JGN-X.

In this experiment, NICT successfully transmitted the broadcast video with high resolution, which satisfied the increasing requirement of tenants and networks. The experiment results made for basic network service techniques to flexibly choose proper network.

- Experiment in 2014 Sapporo Snow Festival

In 2014 Sapporo Snow Festival, a breakthrough was made to successfully transmit 8K video streaming without compression as well as 4K multicast video streaming. This is the worlds first successful experiment of transmitting such high-resolution streaming in long range. The 8K uncompressed video streaming was transmitted with 4K multicast video streaming in a parallel and hybrid way, which indicated that the techniques satisfied requirement of service interim to next generation.

NICT also plans to promote JGN-X to accommodate Big Data experiments, which was thought extremely hard before. Besides, JGN-X is hoped to contribute to the evaluation of SDN techniques and capability to operate technologies of 100 Gbps bandwidth. It is a persistent goal for NICT to keep working for the New Generation Network, satisfying various approaching requirements.

b) Optical Scenario: An optical and circuit integrated (OPCI) network has been designed and developed over RISE, to implement a high speed network infrastructure which could be used in the Metropolitan or Wide Area Network [146], [147]. A mechanism was implemented for OPCI multi-ring [148] network to cooperate with OpenFlow Networks (OFNs). In OPCI network, two kinds of switching are supported in one fiber network, including optical packet switching (OPS) and optical circuit switching (OCS). Therefore, lightpaths can be used to transfer flows in demand of high QoS, while large capacity OPS links can be used to transfer other flows.

Figure 23 shows the experiment setup. Two OFNs are assumed to access the multi-ring OPCI network. Each of the OFN is composed of an OpenFlow Switch (OFS) and a host. OFS 1 and 2 utilize RISE equipment. Meanwhile, host 2 utilizes JGN-X equipment. Each host accesses the OFS with two path interfaces and a packet interface. The dashed lines in Figure 23 mean schematic connections, which may include complicated topological connectivity. The OpenFlow Controller (OFC) directly controls the OFNs, and indirectly controls the OPCI network via the OPCI Network Controller [149].

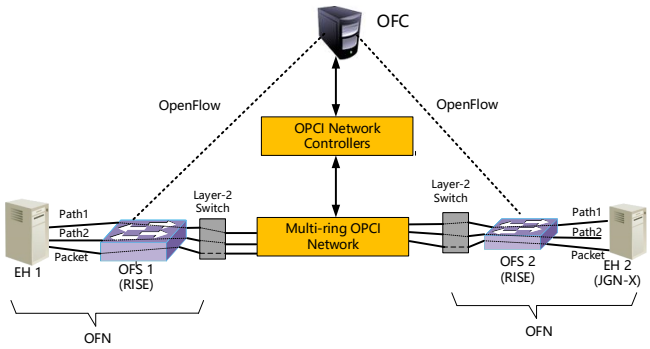

Figure 23. Experiment on the optical scenario in RISE. OpenFlow Controller (OFC) cooperates with Optical Packet and Circuit Integrated (OPCI) to provide an end-to-end path with high-bandwidth.

The results indicate that the experiment succeeds to send and receive ICMP packets by two different optical switching methods (OCS and OPS) in different environment (OPCI multi-ring network and RISE). Thus, cooperation between OPCI network and OpenFlow network has been realized successfully.

D. OF@TEIN

1) Design Objectives and Development: OF@TEIN [25], one of the e-TEIN projects sponsored by Korean Government via NIA (National Information Society Agency), was launched in July 2012. It aims to gradually establish and manipulate an OpenFlow-enabled SDN testbed over TEIN4 (Trans-Eurasia Information Network 4). OF@TEIN is implemented by a union of Korean universities and international collaboration sites, led by GIST (Gwangju Institute of Science & Technology), Korea.

OF@TEIN's design focuses on three tasks: Designing SmartX Racks and making validations, sites deployment and interconnection, Developing some useful SDN tools. So far, OF@TEIN has achieved these goals. However, OF@TEIN is not open for the public, actually it can be accessed from National Research and Education Network (NREN).

2) Key Technologies:

a) Management Technologies: Based on OFELIA Control Framework (OCF), OF@TEIN developed OF@TEIN Portal. It offers Experiment UI to create slice and monitors experiments' status. With the Portal Interfaces, SmartX Rack and network resources could be aggregated to serve experiments, flowspace resources could be remembered to slice the network. Specially, OF@TEIN builds the SmartX Automation Center to manage the infrastructure.

The lifecycle of an OF@TEIN experiment mainly includes three stages: Setup, Execute, and Finish. In the first stage, users setup their experiments through the Portal, which offers Experiment UI to create slice and monitors the experiments' status. With the Portal Interfaces, SmartX Rack and network resources could be aggregated to serve experiments, and flowspace resources could be remembered to slice the network.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

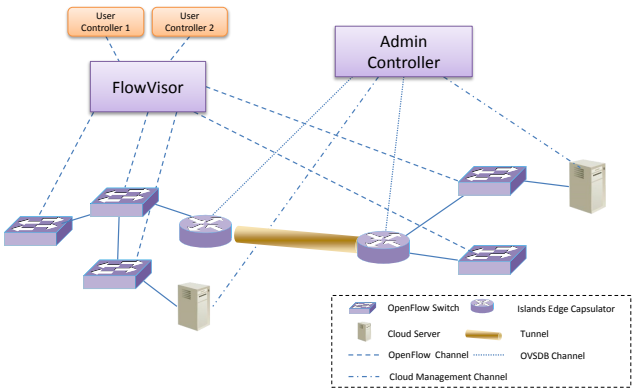

Figure 24. Architecture of OF@TEIN.

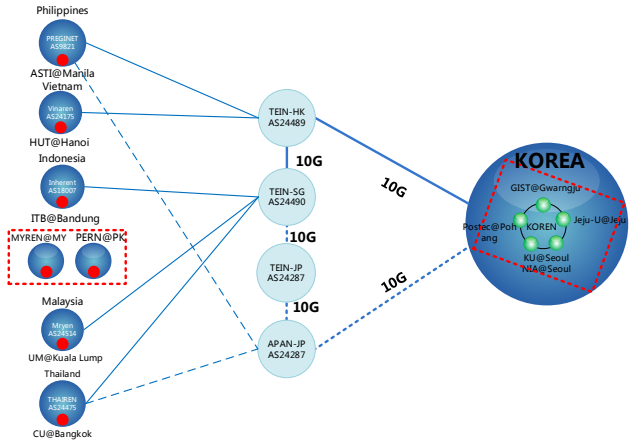

Figure 25. Network deployment of OF@TEIN [25].

In the second stage, users could login to the facility to do their experiments. In the third stage, users finish their experiments and then release resources.

b) Networking Technologies: The networking infrastructure of OF@TEIN is shown in Figure 24. Sites interconnections are implemented by the NVGRE tunneling of the OpenFlow-aware encapsulators [150].

To facilitate the international collaboration with TEIN NRENs (National Research and Education Networks), several types of SmartX Racks [23] have been designed and deployed in every site of OF@TEIN. These racks contain either Open vSwitch (Type A Racks) or OpenFlow data plane (Type B Racks) to connect VM servers. And WAN across data plane-interconnections among these sites is implemented by the Capsulator Nodes in SmartX Racks. Considering that multiple interconnections will be required for each site, NVGRE [151] is chosen to be the tunnel among OF@TEIN sites, which is supported in either Open vSwitch or Narinet in SmartX Racks [23].

The control and management network in OF@TEIN is designed out-of-band as overlay network. In addition to OpenFlow channel, OF@TEIN administrators could manage the tunnels remotely by OVSDB Management Protocol [58] channel, which makes the networking among different sites more automatic.

c) Slicing Technologies: As Figure 24 shows, OF@TEIN uses FlowVisor to share the infrastructure among multiple users with VLAN-Based slicing scheme. The flowspace for each slice just contains DPIDs and Port-ID ranges, and all packets in a slice are pushed through the same VLAN ID at the ingress switch, and finally popped at the egress switch. The intermediate switches submit the VLAN tagged packets to FlowVisor, which recognizes the VLAN field and relay the packets to correct controllers accordingly.

3) Network Deployment: OF@TEIN connects 12 sites distributed in 7 countries (Korea, Indonesia, Malaysia, Thailand, Vietnam, Philippines, and Pakistan) with TEIN4 international network connection. In Figure 25 [152], the red dots represent cities where SmartX is installed while the blue dots represent NRENs. The solid lines and the dashed lines represent primary paths and secondary paths respectively.

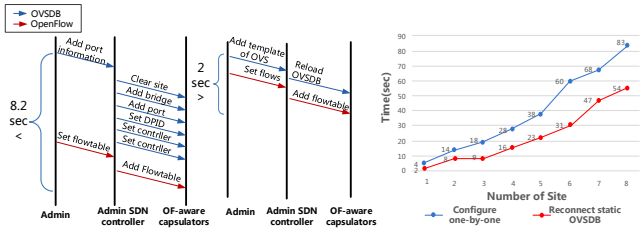

Figure 26. OVSDB configuration experiment in OF@TEIN. The left part shows how the experiment system works, and the right part shows a quicker inter-connection (lower line) comparing with step-by-step configuration (upper line).

4) Experiments: Experiment has been operated over OF@TEIN to verify the operational efficiency of the configuration automation tools. Lifecycle experiment is also interesting to learn. The details are as follows.

a) OVSDB Configuration Experiment: As [153] introduces, a configuration tool for NVGRE tunneling was designed and implemented to automate the OF@TEIN multi-point L2 connections among OpenFlow islands. The configuration utilizes OVSDB [58], which is specially designed to manage OpenVSwitch (OVS), to automate the tunnel establishment. In the left of Figure 26, the time diagram of OVSDB configurations are shown, which indicates a huge reduce of configuration procedures. Besides, the state of configurations can be quickly recovered just by reconnecting to the OVSDB server. In the right of Figure 26, consuming time is compared between step-by-step configuration and OVSDB automation. The result shows that quicker interconnection of the multi-point international OpenFlow islands can be achieved by leveraging the proposed configuration automation tool, comparing with step-by-step configuration.

b) Automated Bandwidth Measurement Experiment: This experiment exemplifies efforts in prototyping lifecycle management by measuring the real-time inter-site bandwidth. A single script controls the whole experiment, which begins with the validity examination of flowspaces and ends with Iperf bandwidth tests across each VM pairs.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

There are two approaches that are valid to verify the allocation of resources and the execution of experiments. One approach is the SDN experimenter UI in the simultaneous utilization experiment, which displays the resource allocation, traffic status and experiment results. Another approach is the script which contains the text details of the experiments and resources. With many convenient tools similar to what this experiment demonstrates, administrators of OF@TEIN could easily manage both the physical infrastructure and the users' experiments.

E. OpenLab

1) Design Objectives and Development: OpenLab [24] is an FP7 network research project, aiming to bring together some existing testbeds to be an open, scalable, and sustainable network infrastructure, as well as providing advances to the early network prototypes or systems for the future Internet research. OpenLab was launched from September 2011, and the project has deployed the hardware and software in its branch testbeds with a diverse set of new applications and network protocols. SDN methods such as OpenFlow have been introduced into the testbed, and now it is SDN-capable in both wireless and optical domains. OpenLab is still available to the public through the OneLab Portal [154], although some nodes are down now.

2) Key Technologies:

a) Management Technologies: There are a number of available testbeds in OpenLab, and most of them use their own control framework. In OpenLab, SFA acts as the orchestrator for resource federation, and meanwhile lots of efforts have been done to develop other frameworks for federation, such as Teagle and OMF. For example, Teagle [155] was redesigned to be the FITeagle framework [156] with focus on the interoperability between SFA AM v3 and ProtoGENI Registry v1.0. FITeagle decouples its core modules, making the core functionalities to be protocol agnostic and unify the portal and interfaces for testbed owners' federation.

The lifecycle of a OpenLab experiment includes three stages: Setup, Execute, and Finish. In the first stage, users setup their experiments through the Experiment UI. The Aggregate Managers provides available resources to create slices and reserve resources. In the second stage, users login to the nodes to do their experiments. In the third stage, users finish their experiments and release resources.

b) Networking and Slicing Technologies: The NITOS and PLE (PlanetLab Europe) testbeds in OpenLab project are primarily extended with OpenFlow. We introduce how these two testbeds are networked and sliced.

- NITOS

NITOS [157] is located in the University of Thessaly, it mainly supports wireless domain researches. The experimental network of NITOS, which is extended to be an OpenFlow testbed, works as the converging network for the wireless access nodes. The control network of NITOS controls the accurate execution of SDN experiments, and also takes charge of monitoring the underlying devices. The management network of NITOS is responsible for controlling the status of

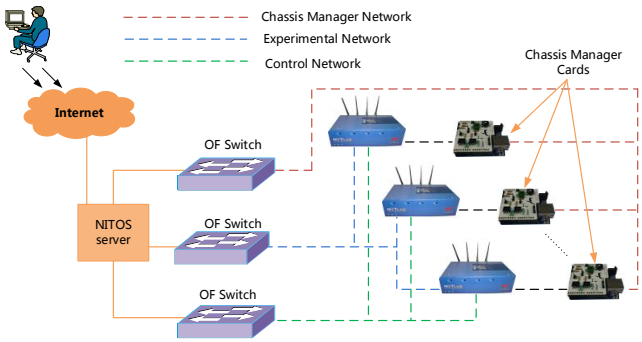

Figure 27. OpenFlow Network in NITOS [157].

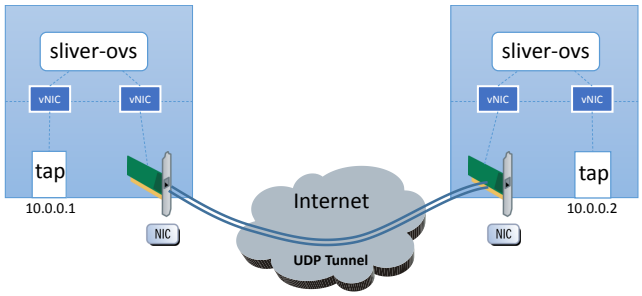

Figure 28. Tunnel implemented by sliver-ovs.

the accessing nodes through HTTP requests to the Chassis Manager Card, which is integrated in each node.

NITOS provides remote access to its OpenFlow switches, and users can easily create their experiments, which are generally sliced via FlowVisor. The whole user flow is orchestrated by the NITOS scheduler, which configures the FlowVisor at the beginning of a new reservation slot.

- PlanetLab Europe

PlanetLab Europe (PLE) [158] is part of the PlanetLab testbed [69], and its OpenFlow capabilities are supported by sliver-ovs [159], a modified Open vSwtich. Sliver-ovs can instantiate one or more virtual OpenFlow switches, which is referred to as slivers, on each node. Slivers in the same PLE slice communicate with each other through the "virtual cables" implemented by UDP tunnels over the Internet. Figure 28 shows the tunnel for the 10.0.0.0/24 slice, users are able to create their overlay OpenFlow experimental network using these tunnels. Initially, slivers have no access to the 10.0.0.0/24 slice. If an application running on 10.0.0.1 tries to send traffic to 10.0.0.2, it will receive an error of "Access Denied" until the tunnel for this subnet is established by standard PLE policies.

Sliver-ovs also performs the slice isolation. The "virtual cable" endpoint can only accept traffic from identified peers, so the overlay network of 10.0.0.0/24 could never be L2 accessible by other slices, which guarantees the basic security of this slicing mechanism.

3) Experiments:

a) EXPRESS: The EXPRESS project in OpenLab [160] is a resilient SDN system, which is designed to extend SDN

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

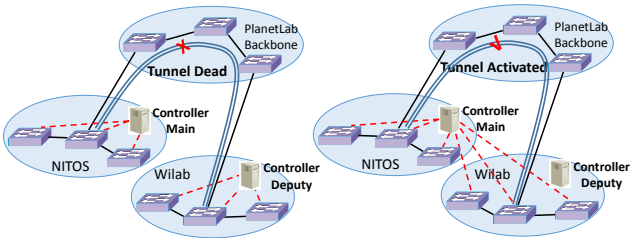

Figure 29. Intelligent controller selection demonstration in OpenLab. The left part shows a local controller connection before different testbeds are interconnected. Once interconnected, controller selection would be based on controllers' priority.

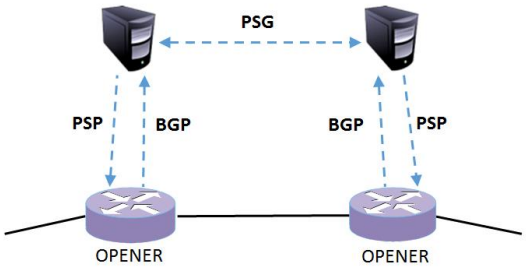

Figure 30. Architecture of PSP-SEC in OpenLab. PSP-SEC intercepts the BGP legacy UPDATE messages and evaluates them to consolidate BGP security.

domains with capability of working in an intermittently connected network circumstance. Figure 29 shows the EXPRESS network deployment: NITOS and w-iLabs.t run two separate Wireless Mesh Network, and they are interconnected through the backbone link emulated with the "virtual cable" across PlanetLab Europe .

The EXPRESS architecture proposes OLSR, an IP-based routing protocol, to route traffic. In addition to OpenFlow, this OLSR-based control plane that also supports special controller to controller messages to scale the SDN domain. Nodes that implement the OLSR protocol are named as the Wireless Mesh Router (WMR). WMR is connected to different controllers, and it supports many resilient control plane strategies, such as controller selection and failure recovery. This example illustrates the intelligent controller selection procedure implemented by the EXPRESS network.

Initially, the tunnel implemented as the "virtual cable" across PlanetLab, between NITOS and w-iLabs.t is inactive, and the WMRs connect to their local available controllers. Once the tunnel is activated, the OLSR messages start to flow between these two wireless testbeds. Thus, the representative WMR in WiLab learns the route to the high-priority controller which is located in NITOS. And then the representative WMR judges the activeness of the remote controller by sending an OpenFlow Hello message. If it is active, the prior controller in NITOS will take control of all the WiLab WMRs to be a more resilient control network.

b) Secure BGP Routing: Securing inter-domain routing (SIDR) challenges ISP networks. Some solutions are in development, but they change BGP too much. The Secured Path State Protocol (PSP-SEC) experiment in OpenLab proposes a novel SDN approach for secure BGP routing.

The PSP-SEC implementation [161] in OpenLab is based on OPENER [162], and the architecture is shown in Figure 30. PSG (Path-State Graph) helps to keep consistent forwarding paths with domain-level interconnections. The Path-State Protocol (PSP) messages spread to manage these PSGs over the network, and monitor the general BGP messages. PSP-SEC is implemented as the counter part of PSP to intercept the BGP legacy UPDATE messages and evaluates them.

Two different PSP-SEC tests have been running [163]. The first one hijacks traffic by pretending to own the particular network prefix, and the other one attacks by advertising an invalid AS-Path. The experiment results show the improvement in BGP routing security, although extra delay would be caused.

VI. Comparison of SDN Testbeds

In this section, five SDN testbeds are compared in terms of design objectives, key technologies, network deployment, and experiments.

A. Design Objectives

Each testbed construction has certain design objectives, which are closely sourced from actual needs. GENI OpenFlow is constructed over GENI, and it is US nation-wide, mainly serves for large-scale network researches. OFELIA is underneath FIRE FP7, and it connects islands across Europe, targeting to address OpenFlow to evolve the Internet network infrastructure. RISE has been developed by NICT as an overlay OpenFlow network across JGN-X, and it aims to provide large-scale and tailor-made environment for cutting edge SDN experiments. OF@TEIN is launched mainly to build an OpenFlow based SDN testbed over TEIN4. And the objective of OpenLab is bringing together existing experimental facility to be a large-scale unified platform.

We can see that all these testbeds wish to address current Internet challenges, such as mobility management and effective content delivery, by developing large-scale, flexible and automatic networks that could accommodate innovative network researches. SDN is born to take this work and OpenFlow is usually chosen to be the fundamental technology.

B. Key Technologies

1) Management Technologies: SDN testbed accommodates different slices for various networking researches, so the control framework is needed for the management work. It is generally a set of software tools, controlling the experiments' lifecycle: reservation, instantiation, configuration, monitoring and uninstantiation.

GENI's Control Framework (GCF) contains four parts: GMOC, Clearinghouse, Tools, and Aggregates. Using the SFA model [99], in which multiple domains cooperate to provide a coherent facility, GENI embraces contributions from a range of networks. Each domain maintains its local autonomy and the ability to set the policies of how to use its own resources. At

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

TABLE IV
COMPARISON OF SDN TESTBEDS

	Objective and Development	Key Technologies			Network Deployment	Experiments
		Management	Networking	Slicing		
GENI OpenFlow	Over GENI • At-scale • Network-innovating	GCF • SFA model • 3A services	• Single VLAN • VLAN Translation • Layer 2 Tunneling • Direct Fiber Connection	• FlowVisor • OpenVirteX • FlowSpace Firewall	US nation wide • Campuses • National backbone network	• Wireless scenarios • Optical scenarios
OFELIA	Pan-Europe testbed • Multi layer • Multi technology • Geographically distributed	OCF • Life-controlling	• Star topology • GEANT L2 circuits • VPN tunnels	• FlowVisor • Optical FlowVisor • VerTIGO	10 islands • Brazil: 1 • Europe: 9	• Optical integration • ICN supporting
RISE	Over JGN-X • Large-scale • Tailor-made	RISE Orchestrator • Easy-managing	• Q-in-Q • Pseudo Wire, based on EoMPLS	• Logical path • MAC rewriting	13 sites • Japan: 10 • Fremdness: 3	• Video streaming scenarios • Optical scenario
OF@TEIN	Over TEIN4 • OpenFlow-based	OF@TEIN Portal • Based on OCF	• NVGRE tunneling	• FlowVisor • VLAN-based	12 sites • All in Asia	• OVSDB automation • Lifecycle verification
OpenLab	Unified platform • Multi-prototype	FITeagle • Easy-federating	• "Virtual Cables"	• FlowVisor (in NITOS) • Sliver-ovs kernel (in PlanetLab Europe)	Some branch testbeds • NITOS • PlanetLab Europe • OSIMS etc.	• Controller selection • Secure BGP routing • PSTN scenario

the same time, GCF provides a mutual trust and collaboration mechanism. The OFELIA Control Framework (OCF) is the orchestration software for OFELIA FP7 facility. There is a formal definition of three separate layers in OCF: the Portal layer, the Clearinghouse layer and the Resource Management layer. Interactions and interfaces among these layers are formally defined. RISE Orchestrator plays the role of control framework in RISE 3.0, it helps to ease the management work for administrator. OF@TEIN Portal is developed based on OCF, and SmartX Automation Center is specially built to manage the infrastructure. OpenLab focuses on the interoperability between existing control frameworks, and a more sustainable approach that is agnostic to the actual federation protocol that is being implemented within the FITeagle Framework.

2) Networking Technologies: Networking technologies help to connect different sites in SDN testbeds to make a large-scale and distributed network.

a) Control & Management Plane Networking: Control & management plane of a testbed is usually an L7 overlay network, because the channel is usually based on TCP/IP, and users and administrators could access these plane remotely over Internet. While in OFELIA, the control plane is deployed out-of-band, using OSPF to route control traffic and uses private IP schema with a subnet assigned to each island. Interconnections of the islands' in the control plane network are implemented via either a dedicated GEANT circuit or an L3 tunnel over the Internet.

b) Experimental Plane Networking: In GENI OpenFlow, the experimental plane uses the existing campus connections to regional and national research networks. The network options used in GENI data plane mainly include: single VLAN, VLAN translation, L2 tunnel and fiber connection. In OFELIA, the experimental plane is an L2 network sliced using VLAN. OFELIA islands are currently interconnected in a star topology to the OFELIA hub and the interconnections between islands are implemented via either dedicated GEANT L2 circuits or L2 tunnels over the public Internet. In RISE, Pseudo Wire technology that uses EoMPLS has been utilized to configure ethernet tunnels between OpenFlow switches. In OF@TEIN, multiple OpenFlow islands interconnect with each other via NVGRE tunnels. In OpenLab, interconnections are often implemented by the "virtual cables" UDP tunnels.

3) Slicing Technologies: Like FlowVisor, slicing tools sit between the physical network and control network, allowing multiple controllers to control the same forwarding elements, thus providing a multi-tenancy network environment.

In GENI OpenFlow, FlowVisor, FlowSpace Firewall and OpenVirteX are adopted to slice the substrate network. In OFELIA, VeRTIGO and Optical FlowVisor are developed. Meanwhile, RISE 3.0 slices its network by Logical Path and OF@TEIN chooses VLAN via FLowVisor. OpenLab leaves the slicing work to NITOS with FlowVisor and PLE with "virtual cable" socket isolation. As many surveys [38], [40] have put forward, these slicing strategies have their characteristics and defects. Flowspaces in FlowVisor could overlap with each other by misconfiguration, in which case FlowVisor would fail to isolate different slices [85]. FlowSpace Firewall can prevent interference but does not support topology virtualization.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

OpenVirteX utilizes MAC address to slice and provide address and topology virtualization. VeRTIGO developed based on FlowVisor provides dynamic optimization of virtual link with VT Planner. Logical Path in RISE rewrites MAC address to match physical paths in JGN-X networks.

C. Network Deployment

The deployment of GENI OpenFlow first started from campus network. Then, at a larger scale, GENI provides an OpenFlow core network with interconnected OpenFlow enabled switches on Internet2 and NLR networks. OFELIA was designed initially with five academic partners setting up individual testbeds within their premises. Afterwards, it expanded with another five partners. After the first duration, OFELIA created a real-world experimental networking substrate that allows flexible control down to individual flows. OpenLab started from September 2011, until now the project has deployed the hardware and software to support different network protocols and applications. RISE has deployed thirteen sites with 10 in Japan and 3 overseas (Los Angeles, Bangkok and Singapore). OF@TEIN has connected 12 sites spread over 7 Asian countries (Korea, Indonesia, Malaysia, Thailand, Vietnam, Philippines, and Pakistan).

D. Experiments

SDN testbeda are important in promoting network innovations. And in turn, experiments could also promote the further development of the testbed. Actually, it is the experiments running over SDN testbed that makes real sense, rather than the testbed itself.

In the optical scenario, Infinera successfully demonstrate a prototype of Open Transport Switch (OTS) in GENI. And OFELIA has attempted several methods to control the circuit switches. In RISE, the research of Optical Packet and Circuit Integrated (OPCI) network is done. In the wireless scenario, Clemson University has designed a system to provide wireless resources for experiments over GENI. In the ICN scenario, OFELIA has designed a simple prototype and proposed a method towards real Name Routing System.

There are also many other interesting experiments that have been supported over SDN testbeds. In RISE, huge demonstrations have been performed on high-quality, real-time video streaming applications. In OF@TEIN, a prototype lifecycle experiment that attempts to link bandwidth measurements has been demonstrated. In OpenLab, the Express project has designed and evaluated an SDN system with intelligent controller selection procedure, and the PSP-SEC experiment has made some progresses in secure BGP routing.

VII. Challenges and Future Works

In this section, the research challenges and future works of SDN testbeds will be discussed.

A. Federation

Federation of global SDN testbeds is a certain trend. Federation implements resources integration, which helps different SDN testbeds to expand the experimental scale and achieve mutual improvement. GENI OpenFlow, OFELIA, RISE and OF@TEIN have all been devoted to implementing federation with other SDN testbeds.

1) The Federation of Control & Management Plane: GENI OpenFlow is developing an international federation API for Clearinghouse functions, which is supported by multiple Clearinghouses and joins US/EU capability up. It is running with shared access to resources from FIRE and GENI.

OFELIA has started to migrate the Expedient based control framework towards emerging de facto standards, because a single unified control framework is still not in sight. Thus, OFELIA makes its clients capable of establishing communication with other testbeds, such as GENI and other SFA-based testbeds.

To interconnect with RISE and Internet2, NICT develops the Extended RISE controller (eRISE) [164]. eRISE is a modified version of OESS, which is implemented by replacing NOX [165] with Trema-based controller [166].

However, there are some important issues that the current federation mechanism needs to resolve and improve. Take eRISE for example, it does not support reactive control method because OESS is originally designed for circuit networks, where paths must be set up proactively. However, reactive control such as OpenFlow PacketIn should be considered for dynamic network control.

2) The Federation of Experimental Plane: Jointly with the partners from GENI, Fed4FIRE has organized the second GENI/FIRE Collaboration workshop in Boston (US) in May, 2014. The workshop scheduled some time to discuss new topics including SDN. Moreover, GENI is investigating and prototyping standards for experimenter-driven resources negotiation and provisioning, extending experimental slices to other research networks including Japan, Korea and Australia.

In addition to the collaboration between GENI and FIRE, a connection has been established via GEANT to connect the MANLAN open exchange point from Internet2. This provides de facto connectivity to the OpenFlow related testbeds in US, Korea, Japan, and Brazil. Moreover, due to the requirements of the FIBRE project, OFELIA is acquiring an alternative link to Brazil via RedIRIS and RedClara. However, in order to make a load balance of experimental traffic, OFELIA also tries to establishing a direct link via TEIN-3 to the Asian OpenFlow communities in Korea and Japan.

RISE has been interconnecting with OFELIA in Europe and OS^3E in US. With OFELIA, the interconnection starts from using each other's OpenFlow testbed by OpenVPN. And with OS^3E, the interconnection is implemented by utilizing OESS. Moreover, RISE is also collaborating with TWAREN through US academic networks.

OF@TEIN has joint SMARTFIRE [167] since 2013. SMARTFIRE is also an wireless network testbed, which aims to enable SDN experiments across Korea and Europe.

It can be discovered that the current federations between SDN testbeds are just on a small-scale, and have many problems to solve. We believe that in the future the federations and collaborations will have a larger scale and a deeper depth.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2630047, IEEE Communications Surveys & Tutorials

B. Slicing

SDN testbeds are expected to serve as an innovative cradle for all future network researchers. Network slicing mechanism could segment the testbed's physical network resources, which are traditionally considered indivisible, and then aggregate them for the isolated slices of different testbed users. Slicing becomes the hotspot of all the SDN testbeds.

FlowVisor, developed based on OpenFlow v1.0, is designed as the transparent proxy between SDN controllers and switches, it could schedule the slice policies using the well-known 12-tuples, and it acts as the slicing tool or the slicing prototype of the SDN testbeds. VLAN is generally used for slicing, and it works in GENI, OFELIA and RISE etc., where the limited 4096 VLAN tags are not actually enough.

However, FlowVisor has the following awkward limitations:Could not make arbitrary separation of the underlying network; Flowspaces could not be overlapped and have no address virtualization; Only supports limited 12-tuples combined rules, and the fields in use could not be multiplexed for different slices; Only available in packet-switching domain, excluding wireless and optical scenarios.

Some new methods have been proposed to overcome these limitations: VerTIGO in OFELIA could arbitrarily separate the physical network by introducing the virtual link technology. OVX exploited by Stanford fixes the address multiplexing problem by address rewriting. OFELIA also explores the OFV (Optical FlowVisor) in optical domain. However, there is still a long way to go. Big gaps should be filled in to implement CPU and bandwidth separation mechanisms [168], without which the logical networks could not be really isolated, and this may require proper configuration protocol (e.g., OVSDB) to be integrated by the slicing tools. Supporting OpenFlow 1.x to introduce the pipeline flexibility still requires many efforts, both theoretical and practical ways. Advanced network embedding algorithms [169]–[171] could be introduced into the slicing mechanism for a more flexible virtualization. Last but not the least, because all the control traffic has to pass through the slicing tool, it could probably need some strong clustering mechanisms to avoid becoming a bottleneck.

C. Compatibility with Existing Network Equipment

To support diverse research efforts in different network domains, SDN testbeds should not only be OpenFlow-enabled for the packet domain, but also support wireless and optical SDN devices; Moreover, the compatibility with existing IP or circuit devices should also be carefully considered. All these different network devices should be interconnected and controlled and managed in a uniform manner.

Many SDN testbeds have been exploring the problem. GENI introduces SDN into Wimax, achieving the prototype of software defined wireless access. OFELIA supports optical switching control by integrating GMPLS module in OpenFlow controller. OSMIS, an OpenLab branch testbed, deploys OpenFlow in PSTN, making it more flexible and well-managed [172]. RISE also tries to provide access to various testbed facilities, such as wireless and optical network, to enhance OpenFlow testbed facilities.

The existing methods of heterogeneous device networking are concluded as belonging to the following two categories: First, extend SDN South Protocol, such as OpenFlow, in the corresponding network domain. The second method comprises the extend existing control and management protocols, such as SNMP or GMPLS to act as an interpreter between SDN controller and Non-SDN forwarding devices. In addition, the corresponding protocol APPs on the controller are also required for unified control. Alien is one of the sub-projects of OFELIA, and it proposes the Hardware Abstraction Layer (HAL), which performs a systematic search of heterogeneous device networking [19].

VIII. Conclusion

This paper addresses large-scale SDN testbeds, which provide experimental environments for newly developed technologies. We began our discussion with some related survey papers and background knowledge. Thereafter, an overview of SDN testbeds was presented, including advantages and design issues, and some key technologies were also briefly introduced. Then, different large-scale SDN testbed implementations were discussed in detail, including design objectives, key technologies, network deployment and experiments. Next, we presented the comparison of different SDN testbeds in term of these four aspects. Finally, we discussed the challenges and future research directions of large-scale SDN testbeds, including federation, network slicing, tools and deployment, multi domain and compatibility with existing network equipment.

To conclude, many efforts have been carried out to deploy SDN testbeds. However, there is still a long way to go. In this paper, we try to introduce some current large-scale SDN testbed implementations, and raise some open issues. Moreover, we have been exploring for years to construct a national wide SDN testbed in China from 2012. A campus-scale SDN testbed, called C-Lab, has been successfully set up in Beijing University of Posts and Telecommunications with great efforts of several laboratories. The success of C-Lab inspires us to move forward, and the China Environment for Network Innovations (CENI) represents our future vision of a large national-scale SDN testbed. The deployment of CENI has already covered some main cities of China, and the enlargement is still in progress. CENI aims to serve for researches of next generation network architecture such as Service Customized Networking [173], Application Driven Network [174] and Hyper-Converged Network, and new network applications such as 4K/8K/VR video, cloud-based business and IOT network intelligence. So far, great efforts have been made on scalable network operating system, network virtualization platform, programmable chip, forwarding devices, and secure federated technologies, which should be valuable as new methods to solve some open issues of large-scale SDN testbeds.

Acknowledgment

We thank the reviewers for their detailed reviews and constructive comments, which have helped to improve the

quality of this paper. This work is supported by the National High Technology Research and Development Program(863) of China (No. 2015AA016101), Beijing Nova Program (No.Z151100000315078).

REFERENCES

[1] I. F. Akyildiz, J. McNair, J. S. Ho, H. Uzunalioğlu, and W. Wang, "Mobility management in next-generation wireless systems," *Proceedings of The IEEE*, vol. 87, no. 8, pp. 1347–1384, 1999.

[2] L. Ma, F. Yu, V. C. M. Leung, and T. Randhawa, "A new method to support UMTS/WLAN vertical handover using SCTP," *IEEE Wireless Commun.*, vol. 11, no. 4, pp. 44–51, Aug. 2004.

[3] V. Sharma, A. Thomas, T. Abdelzaher, K. Skadron, and Z. Lu, "Power-aware QoS management in web servers," in *Proc. Real-Time Systems Symposium*, 2003.

[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[5] S. Shenker, M. Casado, T. Koponen, N. McKeown *et al.*, "The future of networking, and the past of protocols," in *Proc. Open Networking Summit*, 2011.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Commun. Review*, vol. 38, no. 2, pp. 69–74, 2008.

[7] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. M. Parulkar, "Can the production network be the testbed?" in *Proc. Operating Systems Design and Implementation*, 2010.

[8] M. Gerola, R. Doriguzzi Corin, R. Riggio, F. De Pellegrini, E. Salvadori, H. Woesner, T. Rothe, M. Suñe, and L. Bergesio, "Demonstrating inter-testbed network virtualization in OFELIA SDN experimental facility," in *Proc. IEEE INFOCOM WKSHPS*, 2013.

[9] "OVX.onlab." [Online]. Available: http://ovx.onlab.us/

[10] "Openstack Official." [Online]. Available: http://www.openstack.org/

[11] "GMOC." [Online]. Available: http://globalnoc.iu.edu/gmoc/index.html

[12] M. Su and Fundaci, *1st Version of the OFELIA Management Software*. [Online]. Available: http://www.fp7-ofelia.eu/publications-and-presentations/public-deliverables

[13] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A distributed control platform for large-scale production networks," in *Proc. Operating Systems Design and Implementation*, 2010.

[14] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett, "SDX: A software defined Internet exchange," in *Proc. ACM SIGCOMM'14*, 2014.

[15] "OpenDaylight Official." [Online]. Available: www.opendaylight.org

[16] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "ONOS: towards an open, distributed SDN OS," in *Proc. the Third Workshop on Hot Topics in Software Defined Networking*, 2014.

[17] R. Izard, A. Hodges, J. Liu, J. Martin, K.-C. Wang, and K. Xu, "An OpenFlow testbed for the evaluation of vertical handover decision algorithms in heterogeneous wireless networks," *Testbeds and Research Infrastructure: Development of Networks and Communities*, vol. 137, pp. 174–183, 2014.

[18] S. Azodolmolky, R. Nejabati, E. Escalona, R. Jayakumar, N. Efstathiou, and D. Simeonidou, "Integrated OpenFlow–GMPLS control plane: an overlay model for software defined packet over optical networks," *Optics Express*, vol. 19, no. 26, pp. B421–B428, 2011.

[19] *HAL WhitePaper*. [Online]. Available: http://www.fp7-alien.eu/files/deliverables/ALIEN-HAL-whitepaper.pdf

[20] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "GENI: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, 2014.

[21] M. Suñé, L. Bergesio, H. Woesner, T. Rothe, A. Köpsel, D. Colle, B. Puype, D. Simeonidou, R. Nejabati, M. Channegowda *et al.*, "Design and implementation of the OFELIA FP7 facility: The European OpenFlow testbed," *Computer Networks*, vol. 61, pp. 132–150, 2014.

[22] Y. Kanaumi, S.-i. Saito, E. Kawai, S. Ishii, K. Kobayashi, and S. Shimojo, "RISE: A wide-area hybrid OpenFlow network testbed," *IEICE Trans. on Commun.*, vol. 96, no. 1, pp. 108–118, 2013.

[23] J. Kim, B. Cha, J. Kim, N. L. Kim, G. Noh, Y. Jang, H. G. An, H. Park, J. Hong, D. Jang *et al.*, "OF@TEIN: An OpenFlow-enabled SDN testbed over international SmartX Rack sites," in *Proc. Asia-Pacific Advanced Network*, 2013.

[24] "The Project of Openlab." [Online]. Available: http://www.ict-openlab.eu/project-info.html

[25] "OF@TEIN Official." [Online]. Available: http://oftein.net/projects/of-tein/wiki

[26] "JOLnet: a shared network infrastructure for SDN research." [Online]. Available: http://www.telecomitalia.com/tit/it/notiziariotecnico/numeri/2014-2/capitolo-05/approfondimenti-02.html

[27] "CoCo: an exploration of Software Defined Networking (SDN)." [Online]. Available: https://blog.surf.nl/en/coco-an-exploration-of-software-defined-networking/

[28] D. Kreutz, F. M. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[29] W. Xia, Y. Wen, H. F. Chuan, H. Xie, and N. Dusit, "A survey on software-defined networking," *IEEE Commun. Surveys and Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.

[30] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and OpenFlow: from concept to implementation," *IEEE Commun. Surveys and Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014.

[31] J. Xie, D. Guo, Z. Hu, T. Qu, and P. Lv, "Control plane of software defined networks: A survey," *Computer Commun.*, vol. 67, pp. 1–10, 2015.

[32] P. Bhaumik, S. Zhang, P. Chowdhury, S.-S. Lee, J. H. Lee, and B. Mukherjee, "Software-defined optical networks (SDONs): a survey," *Photonic Network Commun.*, vol. 28, no. 1, pp. 4–18, 2014.

[33] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Survey and Tutorials*, vol. 18, no. 1, pp. 602–622, 2016.

[34] L. Cui, F. R. Yu, and Q. Yan, "When big data meets software-defined networking (SDN): SDN for big data and big data for SDN," *IEEE Network*, vol. 30, no. 1, pp. 58–65, Jan. 2016.

[35] Q. Yan and F. R. Yu, "Distributed denial of service attacks in software-defined networking with cloud computing," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 52–59, Apr. 2015.

[36] Q. Chen, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "An integrated framework for software defined networking, caching and computing," *IEEE Network*, 2016, accepted, online.

[37] M. Yang, Y. Li, D. Jin, L. Zeng, X. Wu, and A. V. Vasilakos, "Software-defined and virtualized future mobile and wireless networks: A survey," *Mobile Networks and Applications*, vol. 20, no. 1, pp. 4–18, 2015.

[38] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Commun. Surveys and Tutorials*, vol. 18, no. 1, pp. 655–685, 2016.

[39] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," *IEEE Commun. Surveys and Tutorials*, vol. 15, no. 2, pp. 909–928, 2013.

[40] P. Rygielski and S. Kounev, "Network virtualization for QoS-aware resource management in cloud data centers: A survey," *PIK-Praxis der Informationsverarbeitung und Kommunikation*, vol. 36, no. 1, pp. 55–64, 2013.

[41] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Commun. Surveys and Tutorials*, vol. 17, no. 1, pp. 358–380, 2015.

[42] K. Wang, H. Li, F. R. Yu, and W. Wei, "Virtual resource allocation in software-defined information-centric cellular networks with device-to-device communications and imperfect CSI," *IEEE Trans. Veh. Tech.*, 2016, accepted, online.

[43] R. Huo, F. R. Yu, T. Huang, R. Xie, J. Liu, V. C. M. Leung, and Y. Liu, "Software defined networking, caching, and computing for green wireless networks," *IEEE Commun. Mag.*, no. 11, pp. 185–193, Nov. 2016.

[44] Y. Cai, F. R. Yu, C. Liang, B. Sun, and Q. Yan, "Software defined device-to-device (D2D) communications in virtual wireless networks with imperfect network state information (NSI)," *IEEE Trans. Veh. Tech.*, no. 9, pp. 7349–7360, Sept. 2016.

[45] C. Liang, F. R. Yu, and X. Zhang, "Information-centric network function virtualization over 5G mobile wireless networks," *IEEE Network*, vol. 29, no. 3, pp. 68–74, May 2015.

[46] C. Liang and F. R. Yu, "Wireless virtualization for next generation mobile cellular networks," *IEEE Wireless Comm.*, vol. 22, no. 1, pp. 61–69, Feb. 2015.

[47] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN," *Queue*, vol. 11, no. 12, p. 20, 2013.

[48] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE Commun. Magazine*, vol. 35, no. 1, pp. 80–86, 1997.

[49] D. Sheinbein and R. Weber, "Stored program controlled network: 800 service using spc network capability," *Bell System Technical Journal*, vol. 61, no. 7, pp. 1737–1744, 1982.

[50] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "Forwarding and control element separation (ForCES) protocol specification," *IETF RFC 5810*, 2010.

[51] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, "Design and implementation of a routing control platform," in *Proc. 2nd Conf. on Networked Systems Design and Implementation*, 2005.

[52] J. Vasseur and J. Le Roux, "Path computation element (PCE) communication protocol (PCEP)," 2009.

[53] J. Rexford, A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, G. Xie, J. Zhan, and H. Zhang, "Network-wide decision making: Toward a wafer-thin control plane," in *Proc. HotNets*, 2004.

[54] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4D approach to network control and management," *ACM SIGCOMM Computer Commun. Review*, vol. 35, no. 5, pp. 41–54, 2005.

[55] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," *ACM SIGCOMM Computer Commun. Review*, vol. 37, no. 4, pp. 1–12, 2007.

[56] OpenFlow Switch Consortium, "OpenFlow Switch Specification Version 1.0," 2009.

[57] "OpenFlow Configuration and Management Protocol." [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config1dot0-final.pdf

[58] B. Pfaff and B. Davie, *The Open vSwitch Database Management Protocol*, 2013.

[59] "OVS Official." [Online]. Available: http://www.openvswitch.org

[60] "NS Wiki." [Online]. Available: https://en.wikipedia.org/wiki/Ns_(simulator)

[61] R. Xie, F. R. Yu, H. Ji, and Y. Li, "Energy-efficient resource allocation for heterogeneous cognitive radio networks with femtocells," *IEEE Trans. Wireless Commun.*, vol. 11, no. 11, pp. 3910 –3920, Nov. 2012.

[62] H.-W. Kim and A. Kachroo, "Low power routing and channel allocation of wireless video sensor networks using wireless link utilization," *Ad Hoc and Sensor Wireless Networks*, vol. 30, no. 1-2, pp. 83–112, 2016.

[63] S. Bu, F. R. Yu, Y. Cai, and P. Liu, "When the smart grid meets energy-efficient communications: Green wireless cellular networks powered by the smart grid," *IEEE Trans. Wireless Commun.*, vol. 11, pp. 3014–3024, Aug. 2012.

[64] C. Wang and Y. Zhang, "Time-window and voronoi-partition based aggregation scheduling in multi-sink wireless sensor networks," *Ad Hoc and Sensor Wireless Networks*, vol. 32, no. 3-4, pp. 221–238, 2016.

[65] Z. Li, F. R. Yu, and M. Huang, "A distributed consensus-based cooperative spectrum sensing in cognitive radios," *IEEE Trans. Veh. Tech.*, vol. 59, no. 1, pp. 383–393, Jan. 2010.

[66] Y. Wei, F. R. Yu, and M. Song, "Distributed optimal relay selection in wireless cooperative networks with finite-state Markov channels," *IEEE Trans. Veh. Tech.*, vol. 59, no. 5, pp. 2149 –2158, June 2010.

[67] "Emulab Official." [Online]. Available: http://emulab.net/

[68] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordström, and C. Tschudin, "A large-scale testbed for reproducible ad hoc protocol evaluations," *Proc. IEEE WCNC'02*, vol. 1, pp. 412–418, 2002.

[69] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Commun. Review*, vol. 33, no. 3, pp. 3–12, 2003.

[70] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: A wireless sensor network testbed," in *Proc. the 4th Int'l Symposium on Information Processing in Sensor Networks*, 2005.

[71] N. S. Rao, W. R. Wing, S. M. Carter, and Q. Wu, "Ultrascience net: Network testbed for large-scale science applications," *IEEE Commun. Magazine*, vol. 43, no. 11, pp. S12–S17, 2005.

[72] T. Miyachi, K.-i. Chinen, and Y. Shinoda, "StarBED and SpringOS: Large-scale general purpose network testbed and supporting software," in *Proc. 1st Int'l Conf. Performance Evaluation Methodolgies and Tools*, 2006.

[73] "LISP-LAB official website." [Online]. Available: http://lisplab.openlisp.org/

[74] D. Medhi, B. Ramamurthy, C. Scoglio, J. P. Rohrer, E. K. Çetinkaya, R. Cherukuri, X. Liu, P. Angu, A. Bavier, C. Buffington *et al.*, "The GpENI testbed: Network infrastructure, implementation experience, and experimentation," *Computer Networks*, vol. 61, pp. 51–74, 2014.

[75] B. Davie and Y. Rekhter, *MPLS: Technology and Applications*. Morgan Kaufmann Publishers Inc., 2000.

[76] L. Martini, E. Rosen, N. El-Aawar, and G. Heron, "Encapsulation methods for transport of ethernet over mpls networks," *RFC4448, April*, 2006.

[77] D. Farinacci, P. Traina, S. Hanks, and T. Li, "Generic routing encapsulation (GRE)," *RFC 2784*, 1994.

[78] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "Virtual extensible local area network (VXLAN): A framework for overlaying virtualized layer 2 networks over layer 3 networks," *Internet Req. Comments*, 2014.

[79] M. Sridharan, A. Greenberg, N. Venkataramiah, Y. Wang, K. Duda, I. Ganga, G. Lin, M. Pearson, P. Thaler, and C. Tumuluri, "NVGRE: Network virtualization using generic routing encapsulation," *IETF draft*, 2011.

[80] "KVM Official." [Online]. Available: http://www.linux-kvm.org/page/Main_Page

[81] "Xen Official." [Online]. Available: http://xenproject.org/

[82] "OpenDaylight Virtual Tenant Network." [Online]. Available: https://github.com/opendaylight/vtn

[83] "ONOS ONOSFW." [Online]. Available: https://wiki.opnfv.org/display/onosfw/ONOS+Framework+Homen

[84] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, pp. 1–13, 2009.

[85] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, G. Parulkar, E. Salvadori, and B. Snow, "OpenVirteX: make your virtual SDNs programmable," in *Proc. the Third Workshop on Hot Topics in Software Defined Networking*, 2014.

[86] "FSFW." [Online]. Available: http://globalnoc.iu.edu/sdn/fsfw.html

[87] R. Doriguzzi Corin, M. Gerola, R. Riggio, F. De Pellegrini, and E. Salvadori, "Vertigo: Network virtualization and beyond," in *Proc. Software Defined Networking (EWSDN), 2012 European Workshop on*, 2012.

[88] S. Das, "Extensions to the OpenFlow protocol in support of circuit switching," *Addendum to OpenFlow Protocol Specification (v1. 0)Circuit Switch Addendum v0*, vol. 3, 2010.

[89] "Open Networking Foundation." [Online]. Available: https://www.opennetworking.org/

[90] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, no. 4, pp. 34–41, 2005.

[91] J. S. Turner and D. E. Taylor, "Diversifying the Internet," in *Proc. IEEE Globecom'05*, 2005.

[92] "GENI Wiki Spiral Three." [Online]. Available: http://groups.geni.net/geni/wiki/SpiralThree

[93] "GENI Wiki Spiral Six." [Online]. Available: http://groups.geni.net/geni/wiki/SpiralSix

[94] "Open-networking-summit-explore-software-defined-networking." [Online]. Available: http://engineering.stanford.edu/news/open-networking-summit-explore-software-defined-networking

[95] "GENI Portal." [Online]. Available: https://portal.geni.net/

[96] "GENI Federation Software Architecture Document." [Online]. Available: http://groups.geni.net/geni/attachment/wiki/GeniArchitectTeam/GENI\%20Software\%20Architecture\%20v1.0.pdf

[97] "Clearinghouse." [Online]. Available: http://groups.geni.net/geni/wiki/GeniClearinghouse

[98] "GENI Aggregate Manager API Version 3." [Online]. Available: http://groups.geni.net/geni/wiki/GAPI_AM_API_V3

[99] "SFA Official." [Online]. Available: http://groups.geni.net/geni/wiki/SliceFedArch

[100] "The ExoGENI Official." [Online]. Available: http://www.exogeni.net/

[101] "The InstaGENI Group Page." [Online]. Available: http://groups.geni.net/geni/wiki/INSTAGENI

[102] "GeniApi." [Online]. Available: http://groups.geni.net/geni/wiki/GeniApi

[103] "ExecuteExperiment." [Online]. Available: http://groups.geni.net/geni/wiki/GENIExperimenter/Tutorials/PortalOmniExample/ExecuteExperiment

[104] "ConnectivityOverview." [Online]. Available: http://groups.geni.net/geni/wiki/ConnectivityOverview

[105] D. Li, X. Hong, and J. Bowman, "Evaluation of security vulnerabilities by using ProtoGENI as a launchpad," in *Proc. IEEE Globecom'11*, 2011.

[106] "Understanding Q-in-Q Tunneling and VLAN Translation." [Online]. Available: http://www.juniper.net/techpubs/en_US/junos13.1/topics/concept/qinq-tunneling-qfx-series.html#jd0e34

[107] "National LambdaRail." [Online]. Available: http://www.nlr.net

[108] "Floodlight Official." [Online]. Available: http://www.projectFloodlight.org

[109] "OpenFlow Stanford Deployment." [Online]. Available: http://www.openflow.org/wp/stanford-deployment/

[110] "Internet2." [Online]. Available: http://www.internet2.edu

[111] "NetworkCore." [Online]. Available: http://groups.geni.net/geni/wiki/NetworkCore

[112] "Testbed networks: Provided by nlr." [Online]. Available: http://www.nlr.net/testbeds.php

[113] "GENI OpenFlow Backbone Deployment at Internet2." [Online]. Available: http://groups.geni.net/geni/wiki/OFI2

[114] F. Yu and V. C. M. Leung, "Mobility-based predictive call admission control and bandwidth reservation in wireless cellular networks," in *Proc. IEEE INFOCOM'01*, Anchorage, AK, Apr. 2001.

[115] F. Yu and V. Krishnamurthy, "Optimal joint session admission control in integrated WLAN and CDMA cellular networks with vertical handoff," *IEEE Trans. Mobile Computing*, vol. 6, no. 1, pp. 126–139, Jan. 2007.

[116] "Infinera." [Online]. Available: http://www.infinera.com/

[117] I. Monga, C. Guok, W. E. Johnston, and B. Tierney, "Hybrid networks: Lessons learned and future challenges based on esnet4 experience," *IEEE Commun. Magazine*, vol. 49, no. 5, pp. 114–121, 2011.

[118] A. Sadasivarao, S. Syed, P. Pan, C. Liou, A. Lake, C. Guok, and I. Monga, "Open transport switch: a software defined networking architecture for transport networks," in *Proc. the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, 2013.

[119] "The OFELIA Project and Testbed Federation." [Online]. Available: http://www.csg.ethz.ch/education/lectures/ATCN/hs2013/material/08_OFELIA_slides

[120] The EU FP7 Project and The European OpenFlow Experimental Facility. [Online]. Available: https://github.com/fp7-ofelia

[121] "Opt-In Manager software website." [Online]. Available: http://www.openflow.org/wk/index.php/OptIn_Manager

[122] W. Tavernier, *First Year Report on Planning Development Testing and Operation of Individual Islands*. [Online]. Available: www.fp7-ofelia.eu/publications-and-presentations/public-deliverables

[123] "OpenWRT Official." [Online]. Available: https://openwrt.org/

[124] "GEANT Official." [Online]. Available: www.geant.net

[125] R. Riggio, F. De Pellegrini, E. Salvadori, M. Gerola, and R. Doriguzzi Corin, "Progressive virtual topology embedding in OpenFlow networks," in *Proc. IFIP/IEEE Int'l Symposium Integrated Network Management*, 2013.

[126] S. Azodolmolky, R. Nejabati, S. Peng, A. Hammad, M. P. Channegowda, N. Efstathiou, A. Autenrieth, P. Kaczmarek, and D. Simeonidou, "Optical FlowVisor: An OpenFlow-based optical network virtualization approach," in *Proc. National Fiber Optic Engineers Conference*, 2012.

[127] The EU FP7 Project and The European OpenFlow Experimental Facility, "Publications-and-PresentationsOFELIAFebruary2013." [Online]. Available: http://www.fp7-ofelia.eu/assets/Publications-and-Presentations/OFELIAFebruary2013.pdf

[128] "Specialties and capabilities of each islands." [Online]. Available: http://www.fp7-ofelia.eu/ofelia-facility-and-islands/

[129] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-centric networking: seeing the forest for the trees," in *Proc. the 10th ACM Workshop on Hot Topics in Networks*, 2011.

[130] C. Fang, F. R. Yu, T. Huang, J. Liu, and Y. Liu, "A survey of green information-centric networking: Research issues and challenges," *IEEE Comm. Surveys Tutorials*, vol. 17, no. 3, pp. 1455–1472, Thirdquarter 2015.

[131] L. Veltri, G. Morabito, S. Salsano, N. Blefari-Melazzi, and A. Detti, "Supporting information-centric functionality in software defined networks," in *Proc. IEEE ICC*, 2012.

[132] A. Detti, N. Blefari Melazzi, S. Salsano, and M. Pomposini, "CONET: a content centric inter-networking architecture," in *Proc. ACM SIGCOMM Workshop on Information-Centric Networking*, 2011.

[133] "JGN-X." [Online]. Available: http://www.jgn.nict.go.jp/english/index.html

[134] "User access to RISE." [Online]. Available: http://www.jgn.nict.go.jp/rise/ja/procedures/index.html

[135] S. Ishii, E. Kawai, Y. Kanaumi, S.-i. Saito, T. Takata, K. Kobayashi, and S. Shimojo, "A study on designing OpenFlow controller RISE 3.0," in *Proc. IEEE ICON'13*, 2013.

[136] "IEEE standard for local and metropolitan area networks, virtual bridged local area networks, amendment 4: Provider bridges." [Online]. Available: http://www.ieee802.org/1/pages/802.1ad.html

[137] E. Kawai, "Experience of the RISE Testbed Deployment." [Online]. Available: http://meetings.internet2.edu/media/medialibrary/2015/04/01/20150401-kawai-RISETesbedDeployment.pdf

[138] "RISE: SDN Testbed on JGN-X -RISE Orchestrator." [Online]. Available: http://www.jgn.nict.go.jp/english/reports/presentation/documents/apii_ws-2014_03.pdf

[139] L. Martini, E. Rosen, N. El-Aawar, and G. Heron, "Encapsulation Methods for Transport of Ethernet over MPLS Networks," *IETF RFC 4448*, 2006.

[140] B. Davie and Y. Rekhter, *MPLS: Technology and Applications*. Morgan Kaufmann Publishers Inc., 2000.

[141] "RISE 3.0: The Design and Implementation of SDN/OpenFlow Testbed Considering Node Capacity and Inflexible Topology." [Online]. Available: https://www.apan.net/meetings/Nantou2014/Sessions/FIT/apan38th_itoh.pdf

[142] Y. Kanaumi, S. Saito, and E. Kawai, "Toward large-scale programmable networks: Lessons learned through the operation and management of a wide-area openflow-based network," in *Proc. Network and Service Management (CNSM)*, 2010.

[143] "Broadcasting Operation Experiment over Multiple SDN Switching Network Succeeds in Sapporo Snow Festival." [Online]. Available: http://www.nict.go.jp/en/press/2013/03/18-1.html

[144] "Worlds First Successful Ultra-High-Speed Transmission of Uncompressed 8K Streaming Video of Sapporo Snow Festival." [Online]. Available: http://www.nict.go.jp/en/press/2014/03/12-1.html

[145] T. Miyazawa, H. Furukawa, N. Wada, H. Harai, H. Otsuki, and E. Kawai, "Experimental demonstrations of interworking between an optical packet and circuit integrated network and openflow-based networks," in *Proc. IEEE Globecom Workshops*, 2013.

[146] H. Harai, "Optical Packet and Circuit Integrated Network for Future Networks," *IEICE Trans. on Commun.*, vol. E95-B, no. 3, pp. 714–722, 2010.

[147] T. Miyazawa, H. Furukawa, K. Fujikawa, N. Wada, and H. Harai, "Development of an Autonomous Distributed Control System for Optical Packet and Circuit Integrated Networks," *Optical Commun. and Networking*, vol. 4, no. 1, p. 2537, 2012.

[148] H. Furukawa, S. Shinada, T. Miyazawa, H. Harai, W. Kawasaki, T. Saito, K. Matsunaga, T. Toyozumi, and N. Wada, "A Multi-Ring Optical Packet and Circuit Integrated Network with Optical Buffering," *Optics Express*, vol. 20, no. 27, pp. 28 764–28 771, 2012.

[149] T. Miyazawa, H. Furukawa, N. Wada, and H. Harai, "Development of a common path control-plane for interoperating hybrid optical packet- and circuit-switched ring networks and WSONs," in *Proc. Int'l Conf. Photonics in Switching*, 2013.

[150] N. Kim, J. Kim, C. Heermann, and I. Baldine, "Interconnecting international network substrates for networking experiments," *Testbeds and Research Infrastructure. Development of Networks and Communities*, vol. 90, pp. 116–125, 2012.

[151] M. Sridharan, A. Greenberg, N. Venkataramiah, Y. Wang, K. Duda, I. Ganga, G. Lin, M. Pearson, P. Thaler, and C. Tumuluri, "NVGRE: Network virtualization using generic routing encapsulation," *IETF draft*, 2011.

[152] "SmartX Automation Center for OF@TEIN Multi-point International OpenFlow Islands." [Online]. Available: http://scent.gist.ac.kr/downloads/2014hpcss_6.pdf

[153] T. Na and J. Kim, "Inter-connection automation for of@tein multi-point international openflow islands," in *Proc. The Ninth Int'l Conference on Future Internet Technologies*, 2014, p. 12.

[154] "Onelab Portal." [Online]. Available: https://onelab.eu/services

[155] J. Auge *et al.*, "Control plane extension Status of the SFA deployment," 2013. [Online]. Available: http://www.ict-openlab.eu/publications/deliverables.html

[156] "FITeagle Github." [Online]. Available: https://github.com/FITeagle

[157] "The openflow experimentation of nitlab." [Online]. Available: http://nitlab.inf.uth.gr/NITlab/index.php/testbed/openflow-experimentation

[158] "PlanetLab Europe." [Online]. Available: www.planet-lab.eu

[159] G. Lettieri and L. Rizzo, *OpenFlow enhancements for PLE*, 2013. [Online]. Available: http://www.ict-openlab.eu/publications/deliverables.html

[160] S. Salsano, N. Blefari-Melazzi *et al.*, *EXPRESS final evaluation and overall report*, 2014. [Online]. Available: http://www.ict-openlab.eu/publications/deliverables.html

[161] R. Serral-Graci, G. Riera-Prez *et al.*, *PSP-SEC - Testbed Setup and Interdomain routing Security Evaluation*, 2013. [Online]. Available: http://www.ict-openlab.eu/publications/deliverables.html

[162] "OPENER: An open tool for managing experimentation with SDN Applications." [Online]. Available: http://www.craax.upc.edu/opener.html

[163] R. Serral-Graci, G. Riera-Prez *et al.*, *PSP-SEC - Final experiment report*, 2013. [Online]. Available: http://www.ict-openlab.eu/publications/deliverables.html

[164] S. Ishii, E. Kawai, T. Takata, Y. Kanaumi, S.-i. Saito, K. Kobayashi, and S. Shimojo, "Extending the RISE controller for the interconnection of RISE and OS3E/NDDI," in *2012 18th IEEE International Conference on Networks (ICON)*, 2012.

[165] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[166] "Trema: An Open Source modular framework for developing OpenFlow controllers in Ruby/C." [Online]. Available: https://github.com/trema/trema

[167] "SmartFire Project." [Online]. Available: http://projects.sigma-orionis.com/smartfire/project-key-facts/

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

[169] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Commun. Review*, vol. 38, no. 2, pp. 17–29, 2008.

[170] N. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE INFOCOM'09*, 2009.

[171] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Commun. Review*, vol. 41, no. 2, pp. 38–47, 2011.

[172] "OSMIS Official." [Online]. Available: http://www.ict-openlab.eu/technologies/testbeds/osims.html

[173] Y.-J. Liu, T. Huang, J. Zhang, J. Liu, H.-P. Yao, and R.-C. Xie, "Service customized networking," *Tongxin Xuebao/Journal on Communications*, vol. 35, no. 12, pp. 1–9, 2014.

[174] "Application Driven Network: providing On-Demand Services for Applications, author=Wang, Yi and Lin, Dong and Li, Changtai and Zhang, Junping and Liu, Peng and Hu, Chengchen and Zhang, Gong," in *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*, 2016.