ABSTRACT

ELLIOTT, STEVEN DANIEL. Exploring the Challenges and Opportunities of Implementing Software-Defined Networking in a Research Testbed. (Under the direction of Dr. Mladen A. Vouk.)

Designing a new network and upgrading existing network infrastructure are complex and arduous tasks. These projects are further complicated in campus, regional, and international research networks given the large bandwidth and unique segmentation requirements coupled with the unknown implications of testing new network protocols. The software-defined networking (SDN) revolution promises to alleviate these challenges by separating the network control plane from the data plane [208]. This allows for a more flexible and programmable network. While SDN has delivered large dividends to early adopters, it is still a monumental undertaking to re-architect an existing network to use new technology. To ease the transition burden, many research networks have chosen either a hybrid SDN solution or a clean-slate approach.

Unfortunately, neither of these approaches can avoid the limitations of existing SDN implementations. For example, software-defined networking can introduce an increase in packet delay in a previously low-latency network. Therefore, it is vital for administrators to have an in-depth understanding of these new challenges during the SDN transition. OpenFlow (OF) [209], the protocol many SDN controllers use to communicate with network devices, also has several drawbacks that network architects need to discern before designing the network. Therefore, care must be taken when designing and implementing a software-defined network.

This thesis takes an in-depth look at Stanford University, GENI, and OFELIA as case study examples of campus, national, and international research networks that utilize SDN concepts. Additionally, we detail the planning of the future MCNC SDN that will connect several North Carolina research institutions using a high-speed software-defined network. After dissecting the design and implementation of these software-defined research networks, we present common challenges and lessons learned. Our analysis uncovered some common issues in existing software-defined networks. For example, there are problems with the Spanning Tree Protocol (STP), switch/OpenFlow compatibility, hybrid OpenFlow/legacy switch implementations, and the FlowVisor network slicing tool. These potential issues are discussed in detail. Trends include implementation of OpenFlow version 1.3, use of commercial-quality controllers, and a transition to inexpensive network hardware through the use of software switches and NetFPGAs.

We hope the findings presented in this thesis will allow network architects to avoid some of the difficulties that arise in design, implementation, and policy decisions when campus and other research networks are transitioning to a software-defined approach.

Exploring the Challenges and Opportunities of Implementing Software-Defined Networking in a Research Testbed

by
Steven Daniel Elliott

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2015

APPROVED BY:

Dr. Yannis Viniotis	Dr. Khaled Harfoush

Dr. Mladen A. Vouk
Chair of Advisory Committee

DEDICATION

To my wife for all of her love, support, and understanding throughout this arduous process.

BIOGRAPHY

Steven Elliott was born in North Carolina and lived in Greenville, N.C. for most of his life. He received a Bachelor's degree in Computer Science from North Carolina State University in December 2012. He returned to N.C. State University for his Master's degree in Computer Science in the fall of 2013. After graduation, he plans to work for Sandia National Laboratories in Albuquerque, New Mexico. Steven has always had a passion for the outdoors and enjoys hiking and camping in his free time.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Mladen A. Vouk, for his mentorship, guidance, and tireless effort towards this thesis. I am grateful to Dr. Viniotis and Dr. Harfoush for serving on my thesis committee alongside Dr. Vouk as the chair. I would like to thank William Brockelsby and Dr. Rudra Dutta who provided valuable input and insight into the MCNC-SDN project. I would like to thank Sandia National Laboratories for funding my degree through the Critical Skills Masters Program. This work would not have been possible without the support and guidance of my colleagues at Sandia who introduced me to software-defined networking. I am also grateful for all of my family and friends that provided me with feedback on this thesis and support throughout all of my endeavors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACI	application-centric infrastructure
ACL	access control list
AL2S	Advanced Layer 2 Service
AM	aggregate manager
AMQP	Advanced Message Queuing Protocol
AP	access point
API	application programming interface
BGP	Border Gateway Protocol
CC*DNI	Campus Cyberinfrastructure - Data, Networking, and Innovation
CLI	command-line interface
COTN	California OpenFlow Testbed Network
CPU	central processing unit
DDoS	distributed denial-of-service
DNS	Domain Name Service
DOT	Distributed OpenFlow Testbed
DPID	data path ID
EDOBRA	Extending and Deploying OFELIA in Brazil
FIBRE	Future Internet Testbeds between Brazil and Europe
FOAM	FlowVisor OpenFlow Aggregate Manager
FPGA	field-programmable gate array
GENI	Global Environment for Network Innovations
GMOC	GENI Meta-Operations Center
GMPLS	Generalized Multi-Protocol Label Switching
GpENI	Great Plains Environment for Network Innovation
GRAM	GENI Rack Aggregate Manager
GRE	generic routing encapsulation
GUI	graphical user interface
ICN	information-centric networking
IP	Internet Protocol
IPSec	Internet Protocol Security
IPv6	Internet Protocol version 6
iRODS	integrated Rule Oriented Data System
ISP	Internet service provider
LDAP	Lightweight Directory Access Protocol
LLDP	Link Layer Discovery Protocol
MAC	media access control
MCNC	Microelectronics Center of North Carolina
MIH	Media Independent Handover
MITM	man-in-the-middle
MPLS	Multiprotocol Label Switching
NBI-WG	North Bound Interface Working Group
NCSU	North Carolina State University

NDA	non-disclosure agreement
NIC	Network Interface Card
NLR	National LambdaRail
NPU	network processing unit
NSF	U.S. National Science Foundation
NV	network virtualization
NVGRE	Network Virtualization using Generic Routing Encapsulation
OCF	OFELIA Control Framework
OF	OpenFlow
OF@TEIN	OpenFlow at the Trans-Eurasia Information Network
OFELIA	OpenFlow in Europe: Linking Infrastructure and Applications
OFV	Optical FlowVisor
ONF	Open Networking Foundation
ORCA	Open Resource Control Architecture
OSPF	Open Shortest Path First
OVS	Open vSwitch
PBB	provider backbone bridging
QoS	quality of service
RENCI	Renaissance Computing Institute
RISE	Research Infrastructure for large-Scale network Experiments
RM	resource manager
ROADM	reconfigurable optical add-drop multiplexer
RTT	round-trip time
SDN	software-defined networking
SNAC	Simple Network Access Control
SNMP	Simple Network Management Protocol
STP	Spanning Tree Protocol
TCAM	ternary content addressable memory
TCP	Transmission Control Protocol
TE	traffic engineering
TLS	Transport Layer Security
UNC	University of North Carolina at Chapel Hill
VLAN	virtual local area network
VM	virtual machine
VMOC	VLAN-based Multiplexed OpenFlow Controller
VPN	virtual private network
VXLAN	virtual extensible local area network
WAN	wide area network
xDPd	eXtensible DataPath Daemon

Chapter 1

Introduction

1.1 Motivation and Goals

The traditional network paradigm is rapidly shifting with the advent of software-defined networking (SDN). The SDN industry is expected to become an estimated 8 to 35 billion dollar market by 2018 [78], [166], [281], [282]. Furthermore, funding opportunities for integrating existing networks with SDN technologies are becoming plentiful. For example, the U.S. National Science Foundation (NSF) Campus Cyberinfrastructure - Data, Networking, and Innovation (CC*DNI) program will provide up to $\$1,000,000$ for proposals whose goal is to transition an existing research network to a SDN [230]. Furthermore, companies which process massive amounts of data, such as Facebook and Google, are moving their underlying networks to a software-defined approach [128], [172], [302], [320]. Even Internet service providers (ISPs), such as Verizon and AT&T, are viewing software-defined networking as a viable alternative to their existing infrastructure [74], [322]. While the expectations for the SDN market may vary widely, clearly this technology is an effective alternative to traditional networking and will continue to provide new research opportunities.

In terms of research, to adequately support new network experiments, researchers need a new generation of testbeds. Many network testbeds have already adopted SDN, including those used to originally test OpenFlow's [209] capabilities. However, each existing research network overcame a wide range of challenges before successful implementation. Furthermore, as the OpenFlow protocol and the SDN landscape evolves, existing testbeds need to adapt to new technologies to maximize their capabilities for cutting edge research. Although there are many examples of SDN implementations for architects to follow, there is little work that outlines the issues that may plague a future SDN testbed. This thesis reviews practical issues affecting software-defined networks with the intention of examining existing testbeds and informing future testbed designers about the common benefits and limitations of SDN in terms of the

design and implementation of a research network. We hope that this work will shape future architectural and implementation decisions for existing and developing SDN testbeds.

1.2 Software Defined Networking

Software-defined networking diverges from traditional network architecture by separating the control plane from the data plane [208]. A traditional router performs various hardware functions (i.e., forwarding, lookups, and buffering) while being controlled by routing protocols, such as the Border Gateway Protocol (BGP) and Open Shortest Path First (OSPF), as well as management software, such as the Simple Network Management Protocol (SNMP) or command-line interfaces (CLIs). However, separating the control intelligence from the data plane allows for a simpler switch design, thereby lowering costs and easing management responsibilities. Figure 1.1 illustrates the new logical abstractions that occur in a software-defined network.

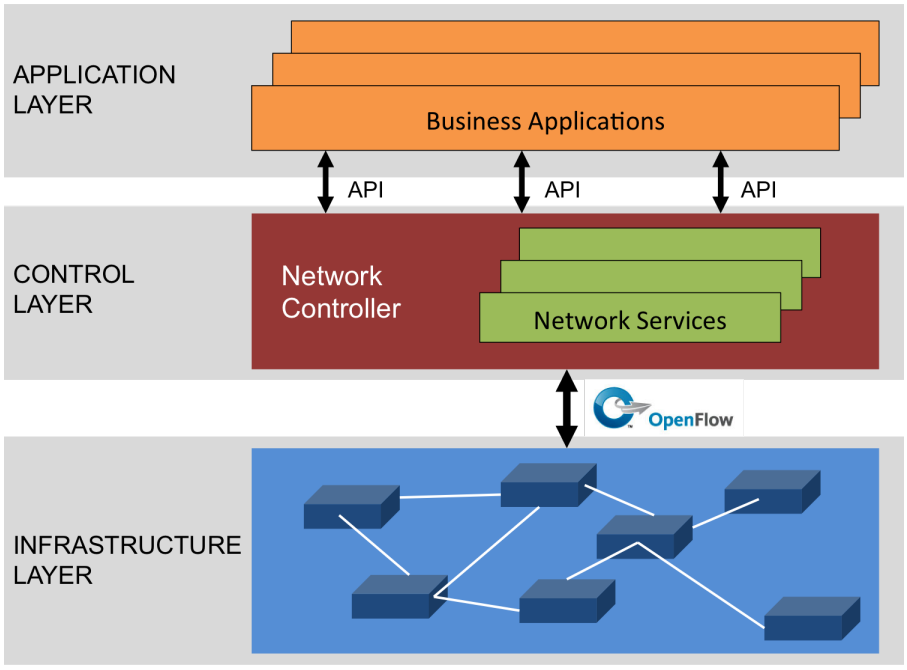


Figure 1.1: A representation of software-defined networking. (Adapted from [238].)

Throughout this thesis, we will refer to any SDN device as a switch, though any network hardware can be SDN compatible. Switches are located within the infrastructure layer and each switch maintains a flow table that determines packet forwarding rules. The controller communicates with each switch through an infrastructure-facing or southbound protocol, as

shown in Figure 1.1. While the typical southbound protocol used is OpenFlow (OF) [209], which has been standardized and endorsed by most major networking vendors [238], the concept of a software-defined network does not rely solely on this protocol. The entire network control plane is managed by a separate device called the network controller. The controller, which can be either a centralized or distributed system, is responsible for determining flow tables and transmitting them to switches in the network. By using the southbound protocol, the controller maintains a global view of the entire network and can facilitate control over forwarding paths. This is in contrast with the localized (e.g., within the router) decision making that occurs in current network architectures and routing/switching devices. Some popular OpenFlow controllers used in research networks include NOX [228], POX [264], Floodlight [100], and OpenDaylight [241]. Section 2.2.1 will provide additional information on controllers that are particularly relevant to this thesis. In addition to a southbound protocol, many controllers also have a northbound (or application-facing) application programming interface (API) which allows for interaction from the application layer. Applications can achieve a wide range of business objectives including collecting analytics, traffic engineering, or even enhancing network security. Through the application layer, an administrator can manage and control the SDN programmatically with fine-grained precision.

1.2.1 OpenFlow

The advent of OpenFlow [209] propelled software-defined networking from a concept into a movement. Originally developed at Stanford, the OpenFlow protocol is now managed by the Open Networking Foundation (ONF) [238]. The OpenFlow specification provides an interface between the controller and the switch as well as details how OpenFlow should be implemented within an OF compatible switch. The protocol outlines how flow tables are defined, the mandatory match fields, and possible actions a switch can take if a match occurs. An illustration of how the packet processing occurs within an OpenFlow switch is presented in Figure 1.2. Within each table the following steps occur: 1) locating a matching flow entry with the highest priority, 2) performing any instructions including packet modifications, updating action instructions, etc., and 3) passing the packet and action set to the flow table downstream.

While there are several iterations of the OpenFlow specification, there are two versions that are salient for this thesis, v1.0 [245] and v1.3 [247]. These versions are prominent for several reasons. The v1.0 specification was the first iteration that gained official hardware support among switch vendors and v1.3 is a more robust specification that provides features for a production network environment. There was enough time between these version changes for vendors to accurately implement most of the required features, unlike version v1.1 and v1.2, which were released within several months of each other. The latest OF specification is v1.5,

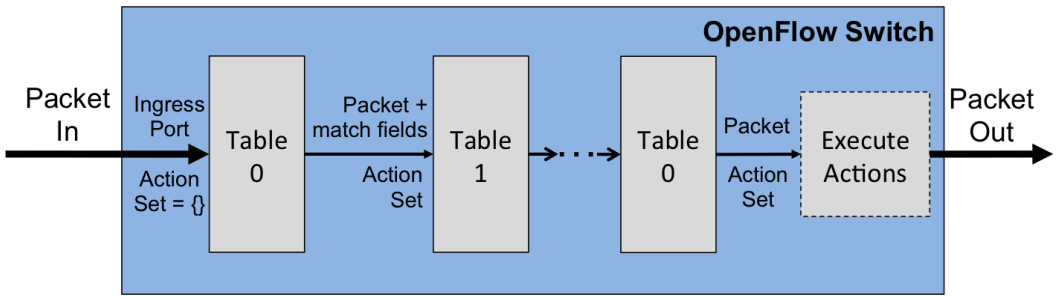


Figure 1.2: Packet flow within a switch's OpenFlow processing pipeline. (Adapted from [247].)

which was released in December 2014 [248]. Unlike most other network protocols, backwards compatibility is not a major goal for new OpenFlow versions. For example, the original required matching fields outlined in version 1.0 differ from those in the 1.3 version. Regardless of these variations, OpenFlow remains an integral part of most software-defined networks.

1.2.2 Application-Centric Infrastructure

While OpenFlow is almost synonymous with software-defined networking, some vendors, most notably Cisco and Plexxi, use a different approach to SDN termed application-centric infrastructure (ACI). This architecture focuses on data center applications and their needs within the network. As shown in Figure 1.3, ACI still uses a network controller to manage policies across the network. However, these policies are for each network application and do not affect route provisioning and forwarding calculations [171]. This architecture ensures that much of the existing router and switch intelligence remains within the hardware [171]. In further contrast, the southbound protocol used in Cisco's ACI is the Cisco-designed OpFlex protocol rather than OpenFlow [171]. While networks using ACI are technically software-defined, this approach is completely different from OpenFlow and is not often used by SDN researchers. Therefore, in the context of this thesis, networks that are software-defined should be viewed as using the architecture outlined by Figure 1.1 rather than application-centric networking, unless explicitly stated.

1.3 Research Networks

Like most areas of research, networking advancements need to be thoroughly tested and evaluated before implementation in a production environment. There are three strategies for evaluating networking research. These methods are simulation, emulation, and infrastructure-based testbeds. While there are various advantages to each of these approaches, this thesis focuses on the third option. Infrastructure testbeds allow experiments to be conducted across a large

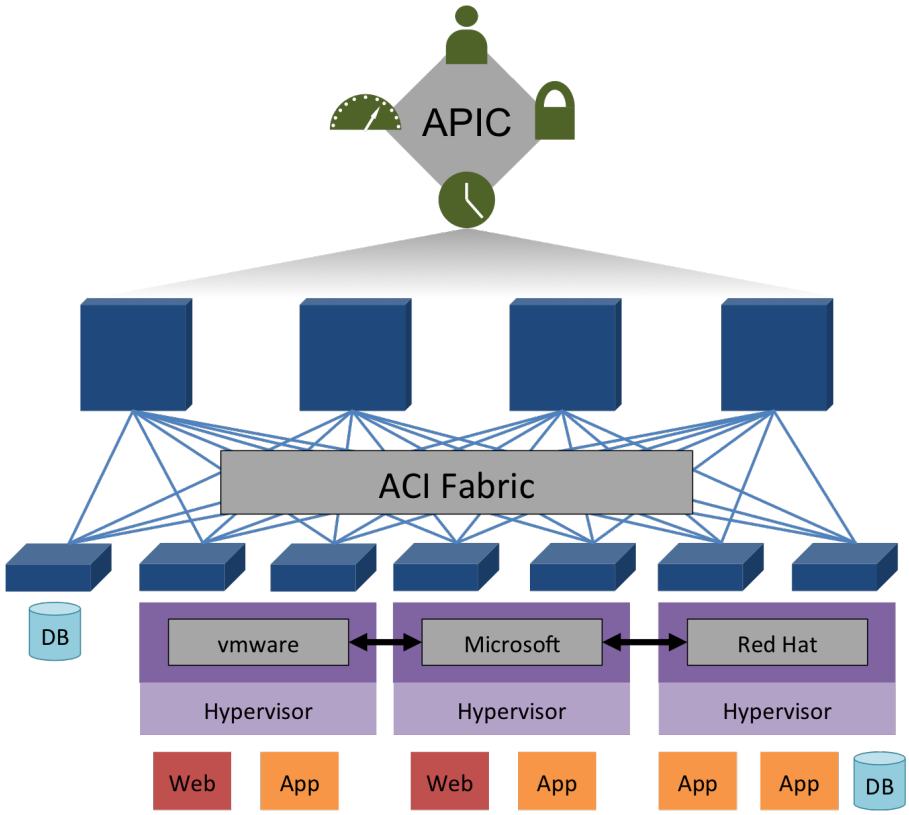


Figure 1.3: Cisco Application-Centric Infrastructure. (Adapted from [56].)

network containing a diverse set of hardware, which can improve fidelity, especially when conducting SDN research [52], [79], [152], [173]. Research networks may vary in size from a single campus to an international network with many partner institutions. Additionally, the number of new infrastructure testbeds is growing and existing testbeds are rapidly expanding [26], [205]. Furthermore, many of the newer testbeds are using SDN as a critical design component. Infrastructure-based testbeds have been and will continue to be a crucial component in expanding SDN capabilities and conducting future research. Therefore, it is vital that testbed architects have insight into the challenges and opportunities that arise when developing a new software-defined network testbed.

1.4 Outline

The remainder of this thesis is organized as follows. Chapter 2 discusses related work, including expanding upon various components required in a SDN testbed, contrasting experiment isolation techniques, outlining methods of testing SDN research, and introducing known SDN

issues. In Chapter 3, we present an in-depth examination of four research networks that use software-defined networking as a critical design component. In particular, we provide the reader with an understanding of how SDN technologies impact the network design and provide several examples of research conducted on the network. Next, Chapter 4 provides an analysis of the case studies, in which we discover commonalities and differences that may significantly impact future testbeds. We conclude by exploring future research directions in Chapter 5.

Chapter 2

Related Work

This chapter reviews state-of-the-art software-defined networking research with a particular focus on research networks. Although the foundation for SDN already exists, new SDN research increases the flexibility of existing testbed capabilities and offers new methods of testing next-generation Internet protocols and architectures. To understand the unique advantages provided by SDN, it is important to recognize the differences in seemingly similar ideas. Therefore, we explore the distinction between SDN and various other network virtualization techniques. Next, we discuss network controllers and SDN within a wide area network (WAN). Section 2.3 presents relevant research on testing SDN solutions using simulation, emulation, or infrastructure-based testbeds. Finally, we discuss known limitations of SDN in research networks.

2.1 Network Virtualization

In general, network virtualization (NV) allows multiple virtual networks to coexist in the same underlying physical infrastructure. Virtualization is a broad concept incorporating an array of technologies. A more accurate definition provided by Wang et al. [323] states, "network virtualization is any form of partitioning or combining a set of network resources, and presenting (abstracting) it to users such that each user, through its set of the partitioned or combined resources has a unique, separate view of the network." In addition to clarifying what network virtualization is, [323] provides an important guide to virtualization technologies. An in-depth research overview of NV is presented with survey papers such as [53] and [54]. It is important that readers note that network virtualization is different, although complimentary to, software-defined networking. This section provides an overview of how network virtualization differs from SDN and the ways in which these concepts are used within research networks.

2.1.1 Overlay Networks

Overlay networking, a network virtualization technology, enables virtual networks to exist on top of a physical network, typically through the application layer [54]. Overlays have been used to improve network quality of service (QoS) [294], enhance routing performance [6], or even enable anonymity [73]. Furthermore, some infrastructure testbeds, such as PlanetLab [55], use overlays as a critical design component resulting in dual use as a testbed and a deployment platform [256].

There are several differences between overlays and SDN. First, it is important to note that they have fundamentally different architectures. Overlays work on top of existing network infrastructure whereas SDN creates a new underlying infrastructure. Therefore, it is possible to use overlay concepts within a software-defined network. Second, SDN aims to solve large architectural challenges while overlays are incapable of doing so because they typically use the application layer [7]. Finally, SDN provides a global view of the entire network. In contrast, overlays are limited to their narrow purpose-built view, leaving them unable to view any coexisting overlays [7]. For these reasons, next-generation testbeds, such as Global Environment for Network Innovations (GENI), use overlays in conjunction with software-defined networking.

2.1.2 Network Slicing

One of most important components of research testbeds is the ability to host many simultaneous experiments without studies interfering with one another. This greatly increases the usability of the testbed while maintaining the fidelity of each experiment. The ability to isolate different experiments, known as network slicing, has been foundational to the creation of research testbeds since the early 2000s [256]. Early examples include Emulab [127], [329], PlanetLab [55], [256], and VINI & Trellis [22], [27]. Additionally, new SDN testbeds, like those presented in Chapter 3, are still using network slicing to maintain experiment isolation. Although slice definitions vary by testbed, most slices consist of the following: 1) a topology containing a set of network, storage, and computational resources, 2) a mapping between those resources and the underlying physical infrastructure, and 3) an aggregation of predicates that define which network packets may enter the slice [119]. Isolation between sets of traffic is not a new idea; therefore, we will distinguish the ways in which SDN differs from and can improve upon this concept.

Past Techniques

Virtual local area networks (VLANs) stipulate isolation between different layer 2 network segments. The guarantees provided by VLANs make them ideal for segregating experiments within a testbed environment. Additionally, they are an important part of many SDN testbeds because they separate OpenFlow traffic from non-OpenFlow traffic. This increases testbed flexibility by

enabling a hybrid mode that operates with both OpenFlow and legacy traffic coexisting. However, VLANs are time consuming to manage and have scalability limitations (i.e., typically only 4096 VLAN IDs). Therefore, while VLANs are complimentary to SDNs, an isolation solution using SDN may provide additional flexibility. For example, Lara et al. [198] improved upon statically configured VLAN tags by developing a dynamic traffic isolation solution. The tool Lara et al. developed implements dynamic VLAN creation, deletion, or modification without traffic disruption [198].

While VLANs are the most popular traffic isolation mechanism, there are other segmentation techniques, including Multiprotocol Label Switching (MPLS) [269], pseudowire [42], IEEE 802.1ad [164] (i.e., Q-in-Q tagging), virtual extensible local area networks (VXLANs) [203], and Network Virtualization using Generic Routing Encapsulation (NVGRE) [292]. MPLS is a QoS oriented protocol built for traffic engineering. By attaching a label to the packet header, MPLS improves performance by replacing complex Internet Protocol (IP) lookups with a quick label lookup. Though MPLS does not provide strong isolation guarantees, it creates a common platform to route packets without regard for any particular layer 2 technology. Given its disregard for the underlying architecture, MPLS can be used in conjunction with a software-defined network. Furthermore, the centralized network view provided by SDN can optimize MPLS services [286].

Pseudowire allows the emulation of point-to-point services across packet-switched networks. A pseudowire aims to deliver the minimum functionality of a "transparent wire" while maintaining the required degree of fidelity, which is dependent on the emulated service [270]. Because this emulation sits above packet-switched protocols like MPLS and IP, its end router configuration can be defined through OpenFlow [211].

Q-in-Q tagging was originally intended to allow service providers to encapsulate customer traffic and allows multiple VLAN tags for each frame. It can be used in research testbeds to ensure isolation between experiments without disrupting any intra-experiment VLANs. Additionally, Q-in-Q dramatically increases the number of available VLANs. This protocol can be used not only within a SDN environment, but it can be beneficial to OpenFlow testbeds by improving upon the limitations of traditional VLANs.

Finally, both VXLAN [203] and NVGRE [292] are new overlay protocols designed for addressing scalability within cloud environments. They overcome the scalability limitations of VLANs by allowing for about 16 million isolated networks. As with all overlay technologies, these protocols will not replace software-defined networking but can coexist with the architectural changes SDN creates. Ultimately, all of the above solutions provide some traffic segmentation but do not fundamentally change an existing network architecture, which differs from a SDN implementation. Therefore, software-defined networks are capable of using these existing protocols in conjunction with OpenFlow; this flexibility offers new solutions for research

networks.

SDN Slicing Solutions

Testbeds that use traditional isolation mechanisms, such as Emulab and PlanetLab, have three major limitations: speed, scalability, and ease of technology transfer [288]. Within these testbeds, all packet processing and forwarding is performed in software, thereby slowing down the performance of the network. In contrast, a SDN solution ensures that, while the controller sets the forwarding rules, hardware performs the processing and forwarding. Next, these testbeds need to support both the overlay mechanisms and underlying infrastructure; therefore it is cost prohibitive to scale to a global size. Finally, these testbeds perform packet processing on central processing units (CPUs) rather than network processing units (NPUs), creating a significant challenge in transferring the experimental results into vendor hardware. Sherwood et al. [288] aimed to solve these problems by leveraging SDN in a new slicing mechanism. They developed FlowVisor [287], which allows control plane message slicing, thereby enabling user-defined manipulation of packet forwarding [288]. In addition to this novel slicing mechanism, FlowVisor is transparent to both the control and data planes and provides guaranteed isolation between slices. However, there are some limitations to FlowVisor that are discussed in Section 4.2.2.

There are several approaches to extend or enhance FlowVisor, including ADVisor [275], VeRTIGO [75], and GiroFlow [10]. ADVisor overcomes FlowVisor's inability to create arbitrary virtual topologies, though it uses VLAN tags to distinguish between flowspaces [275]. VeRTIGO extends FlowVisor further by providing unique views of the network based on customer specifications [75]. That is, it can instantiate an arbitrary virtual network, allowing the user full control. For a simpler set of requirements, VeRTIGO abstracts the whole network into a single instance. GiroFlow compliments FlowVisor by providing a management model to manage network slices easily based on the applications used within the slice [10]. A similar alternative to FlowVisor is a simple slicing tool termed the FlowSpace Firewall [101]. It allows OF switch configuration via multiple controllers while prohibiting them from interfering with each other's flow rules.

Diverging from FlowVisor and similar tools, Gutz et al. [119] argue that slice abstraction should happen at a language level because isolation at this level allows for automatic optimizations within an implementation, reduced latency in the absence of a proxy-based solution like FlowVisor, and formal verification of an implemented policy [119]. Additionally, the authors present a tool that will take in a collection of slice definitions and output a set of OpenFlow rules [119].

Other research ideas offer entirely different approaches to slicing by aiming for full NV. One solution, OpenVirteX [5] maintains the proxy-based mechanism proposed by FlowVisor, but

adapts it to full network virtualization. OpenVirteX functions as a network hypervisor that provides NV, including a user-directed topology and full header space, to each tenant [5]. Bozakov and Papadimitriou [30] outline a new SDN hypervisor, which would create a virtual SDN, and suggest that this technique can be incorporated into OpenVirteX to increase its versatility. In contrast to OpenVirteX, which uses a single proxy, AutoVFlow [331] utilizes multiple proxies to provide robust flowspace virtualization on a wide area network without the need for a third party. However, not all NV solutions are proxy-based. For example, FlowN [77] provides a scalable network virtualization solution by using databases to provide mappings between physical and virtual networks. Additionally, FlowN uses a form container-based virtualization in which controller applications are run.

2.1.3 Software Switches and Routers

Software-based routers and switches are another network virtualization technology closely associated with SDN. While the idea of software routers, such as XORP [122], predate OpenFlow, SDN can improve the feasibility and flexibility of software-based networking devices. For example, one of the best known software switches, Open vSwitch (OVS) [240], uses a flow-oriented forwarding approach, which is based on ideas found in OpenFlow [257]. Furthermore, since its inception, OVS has supported the latest versions of the OpenFlow protocol [257]. Other virtual routers and switches include the OpenFlow 1.3 Software Switch [242], Cisco's Nexus 1000V [57], and Brocade's Vyatta [41]. Like other virtualization technologies, these virtual switches and routers are not inherently SDN technologies but rather tools used to facilitate SDN deployment and support. One example of the quick development speed is Pica8's support for OpenFlow v1.4, the latest OF version implemented by any vendor listed in Table A.1. This is largely because Pica8's switch software, PicOS, relies on Open vSwitch to provide OpenFlow support [260]. While SDN and software switches are becoming increasingly interrelated, a software switch does not guarantee SDN support. That is, regardless of whether a switch is implemented in hardware or software, its architecture fundamentally needs to be modified before supporting OpenFlow or other SDN mechanisms.

2.2 SDN Advancements

2.2.1 OpenFlow Controllers

Although the fundamental purpose of a network controller was outlined in Section 1.2, the reader will benefit from a familiarity with popular SDN controllers. This section focuses on open source projects; however many commercial alternatives are available [201].

NOX [228] was the first OpenFlow controller and was written in C++ by Nicira Neworks.

While it provided the research community with a starting point, the code base has not been maintained and has since been deprecated. It was replaced by POX [264], which is a python-based controller. POX is still used in many research projects as it is an easy platform on which to develop.

Beacon [81] and Floodlight [100] are Java based controllers. Beacon was developed at Stanford and serves as the code base for Floodlight, which is being maintained by Big Switch Networks. While Beacon began as a research project, Floodlight is being developed actively and guided Big Switch in developing a commercial SDN controller [28]. NOX, Beacon, Floodlight, and Maestro [43], a highly scalable OF controller, were compared in a performance evaluation in [285]. Subsequently, Shah et al. presented controller architecture guidelines based on the performance goals of the controller [285].

OpenDaylight [241], another Java based controller, is supported by a large consortium of companies including Cisco, Juniper, Microsoft, HP, and IBM. With this large support community, it is being developed rapidly to support new southbound protocols, such as Cisco's OpFlex [251], and new services, such as OpenStack. In addition to the controllers mentioned above, OF controllers have been developed with a wide range set of performance enhancements and features in numerous languages including Ruby [316], C [237], and JavaScript [25]. However, most of these systems function as a single controller that manages the entire network.

In contrast, the Onix [193], Hyperflow [315], and Devolved controllers [303] have distributed control planes which address the scalability and reliability issues of a traditional network controller. However, each of these solutions require that every controller has the same network-wide view. Kandoo [124] overcomes this limitation by proposing a hierarchical controller architecture in which each distributed local controller has a different network view and root controllers contain the network-wide view. Another solution is DISCO [258], which uses a distributed control plane for multi-domain networks. Built on top of Floodlight, DISCO uses Advanced Message Queuing Protocol (AMQP) to coordinate with neighboring controllers [258].

2.2.2 SDN in the WAN

Large-scale SDN testbeds allow researchers to experiment with the applicability of SDN in a WAN. One of the first software-defined WANs is B4 [172], which is the wide area network that links all of Google's data centers. Because of Google's unique and data intensive services, B4 has a variable traffic rate and an immense bandwidth. Most WAN links are overprovisioned to provide significant reliability at the considerable expense of additional bandwidth and equipment [172]. Using SDN, Google's traffic engineering (TE) algorithms ensure that average bandwidth utilization is maximized while balancing application priority with network capacity [172]. Additionally, because the TE mechanisms are located in the application layer

rather than the control layer, Google has a fail-safe mechanism. That is, network administrators can transition the network back to a shortest path algorithm should an issue with the TE application occur [172]. Because of the large unprecedented nature of B4, the Google team developed custom OpenFlow switches and control applications [172]. As one of the largest and oldest software-defined networks, B4 is considered a tremendous success because it has boosted many of the links to 100 percent utilization [172]. Additionally, it helped lay the foundation for large-scale software-defined networks including the testbeds described in Chapter 3.

One disadvantage of centralized network control is the loss of resilience, particularly in the event of a natural disaster. Nguyen et al. [225] identified the physical connection between switches and controllers, network modifications, and lack of disaster recovery plans as the three key problems of resilient software-defined WANs. The authors conducted two sets of experiments to evaluate a software-defined WAN and examine its ability to recover quickly in the event of a disaster [225]. The first set of experiments focused on finding the average and worst case latency between controllers and switches [225]. They found that even the worst case round-trip time (RTT) was smaller than many resiliency requirements for existing WANs [225]. Therefore, the authors suggested that one controller and a backup will be sufficiently fast for WAN disaster recovery time [225]. The second simulation demonstrated the impact of an earthquake on a software-defined WAN by interrupting communication from one of the nodes [225]. This study found that the OpenFlow network experienced some performance loss, but overall it restored traffic flow quickly [225]. The minimal added latency for controller-to-switch communication and the ability to update routing tables quickly for a given network demonstrates the importance of SDN in creating survivable networks [225].

2.3 Experimenting with SDN

2.3.1 Simulation Based Testing

The need to analyze new network protocols has resulted in the creation of simulation tools to automate this process. Although REAL [186] and ns-1 [206] have been surpassed by newer versions and other simulators, similar tools remain helpful in evaluating networks. For example, VINT [32] is a framework that provides simulation tools, specifically ns, to help researchers test wide area protocols. This approach alleviates some of the simulation limitations raised by Paxson and Floyd [255] by incorporating techniques like abstraction and emulation. However, even in the context of its shortcomings, simulation can be a useful tool to evaluate SDN research.

One modern simulator is ns-3 [126], which now includes support for OpenFlow [229]. The ns toolkit has already proved useful for evaluating a traffic engineering algorithm, which was optimized for software-defined networks [4]. Agarwal et al. performed two sets of experiments, the

first involves a static performance measurement and the second using a discrete event network simulator (ns-2) [4]. In both cases, a traffic matrix and a network topology were provided as input to the simulator. The static measurement tests were designed to examine the performance improvement while the ns-2 tests measured packet loss and network delay [4]. These simulations demonstrated that SDN routing produces less delay and packet loss than OSPF routing.

Another modern simulator is fs-sdn [118], which leverages POX and the fs [291] simulation platform. It evaluates SDN applications rigorously and at large scale so that they are translatable to a real SDN controller [118]. Jin and Nicol [173] implemented OpenFlow within S3F, a scalable simulation framework. This framework provides greater context to a simulation by also using emulation tools and a virtual-machine-based testing environment. The authors claim that their design improves the efficiency of a simulated OpenFlow controller; however, the overall performance results were not compared to that of a physical testbed nor to any emulation based tools [173]. There are also commercial OpenFlow simulation tools, such as Epoch by Northbound Networks [80]. While information about Epoch is sparse, the company claims it works with any major SDN controller and can run on both Linux and Mac OS X [80].

2.3.2 Emulation Based Testing

Emulation-based systems provide a balance between the shortcomings of simulation tools and the large cost of a physical testbed. One of the most popular early emulation tools is Dummynet [268]. It was originally intended as a link emulator, but has since been updated with new extensions and even has been integrated into the PlanetLab testbed [46].

Mininet [197] is the most popular SDN emulation tool currently available. Within a single virtual machine (VM), it quickly creates a realistic virtual network. It is being actively developed and has been used in many SDN research projects including [102], [266], [336], and [198]. Also, there are many extensions and adaptations to Mininet. For example Mininet-HiFi [121] uses Linux containers to add isolation and ensure fidelity within Mininet. This work, published by the same authors of Mininet, has been incorporated into the main Mininet source code [121], [213]. While Mininet is a popular platform, it has a few noticeable limitations. Huang et al. [152] demonstrates that, without emulating vendor-specific OpenFlow implementations, there are noticeable differences between the emulated environment and a physical implementation. This is due largely to the variation between OVS, the OpenFlow switch implemented in Mininet, and vendor implementations of the OpenFlow protocol.

Another important limitation of Mininet is its inability to generate large networks. Mininet CE (cluster edition) [8] overcomes this limitation and can emulate large scale networks by creating a cluster of Mininet instances. This work is extended further into a large scale simulation solution in [9]. Using a concept similar to Mininet CE, Roy et al. [271] developed a Distributed

OpenFlow Testbed (DOT) that also uses a clustering approach to emulate a large network. Similarly, Wette et al. [328] developed MaxiNet, which abstracts multiple Mininet installations, thereby distributing the emulated network across many physical machines. While Mininet CE, DOT, and MaxiNet take similar approaches to emulating large networks using Mininet, they have not yet been compared, indicating that future work is needed to determine the most effective solution.

In contrast to the open source solutions above, there are commercial SDN emulators available. EstiNet produced the EstiNet OpenFlow network simulator and emulator [326]. This combination of simulation and emulation tools mirrors [173] and offers increased flexibility. Additionally, Wang et al. [326] compared EstiNet to Mininet and ns-3 and found improved scalability and performance correctness.

2.3.3 Infrastructure Based Testing

There are many infrastructure-based SDN testbeds, including GENI, OpenFlow in Europe: Linking Infrastructure and Applications (OFELIA), MCNC, and Stanford. These cases will be presented in Chapter 3. Infrastructure testbeds differ in terms of a wide range of properties including size and scale, bandwidth, hardware, link type, and control software, among others. Though the main research focus of each testbed is different, all serve the specific goal of advancing SDN research. Several SDN testbeds that are not discussed in Chapter 3 are listed and described briefly below.

- **ESNet 100G SDN Testbed:** This high performance testbed supports a range of research topics including network protocols and high-throughput applications. Its main goal is to provide an environment that creates reproducible experiments [82].
- **California OpenFlow Testbed Network (COTN):** This testbed offers California researchers a breakable testbed with no restriction on traffic types. Additionally, it guarantees that experiments will be unhindered by any production traffic [59].
- **Future Internet Testbeds between Brazil and Europe (FIBRE):** FIBRE is a federated testbed containing both Brazilian and European islands. It uses three unique control frameworks to allow the concurrent allocation of various resource classes, including OpenFlow assets [272].
- **German Lab (G-Lab):** This German testbed offers some OpenFlow support through a few hardware switches at one site and OVS at other sites [277].
- **K-GENI:** Spanning both Korea and the US, this testbed's core is based on OpenFlow and is connected to GENI's backbone networks to support international GENI-based

experimentation [188].

- **Research Infrastructure for large-Scale network Experiments (RISE):** Functioning as an overlay to JGN-X, a Japanese wide-area testbed, RISE uses a hybrid network approach to enable OpenFlow experiments on an existing infrastructure [184].
- **OpenFlow at the Trans-Eurasia Information Network (OF@TEIN):** Similar to RISE, OF@TEIN is an effort to build a SDN testbed on an existing network. This international testbed uses SmartX Racks, similar to GENI racks, to provide OpenFlow support to many Eurasian researchers [189], [233].

2.4 Issues with SDN

Because SDN is a new technology, there are many issues with few definitive solutions. Learning about existing issues and potential solutions, rather than discovering them first-hand, provides administrators who plan to implement SDN technologies with a noticeable advantage. This thesis discusses SDN limitations specifically in the testbed setting; however, there are other general issues that may create difficulty for implementers. The most common software-defined networking concerns include interoperability, scalability, and security [284].

2.4.1 Interoperability

We address some interoperability concerns in Section 2.1.2 by discussing the way in which SDN works with existing isolation and traffic shaping technologies. However, interoperability between applications and the control plane continues to be a concern. While OpenFlow has been standardized as a southbound API, there is not yet a standard northbound API allowing each controller to specify its own unique interface. Without standardization, developers may need to customize their applications for different controllers. However, the Open Networking Foundation is working to standardize several northbound APIs through the North Bound Interface Working Group (NBI-WG) [236]. While the NBI-WG has not provided any guidelines to a standard northbound API yet, the formation of this working group demonstrates the dedication of the SDN community to resolving this issue. Additionally, in Section 4.3.1 we address the interoperability of divergent OpenFlow implementations by hardware vendors.

2.4.2 Scalability

In Sections 2.2.1 and 2.2.2, we discussed scalability solutions through the use of multiple controllers and large-scale WAN deployments. One remaining scalability concern is termed the "Controller Placement Problem" [125]. Heller et al. [125] conducted a series of experiments

to examine this issue. Findings from this study include: 1) a single controller typically fulfills latency requirements, 2) random controller placement within a topology is less than optimal, 3) in general, additional controllers have a less than proportional decrease in latency, and 4) in three-fourths of tested networks, one cannot optimize for both worst-case and average-case latency. Bari et al. [19] improved upon [125] by addressing the "Dynamic Controller Provisioning Problem." The authors developed a framework for dynamically deploying controllers based on their optimal location within a network while minimizing cost. Other scalability concerns are sufficiently debunked by Yeganeh et al. [335], including flow initiation overhead, resiliency to failures, and scalability within different network architectures.

2.4.3 Security

The added benefits that SDN offers are contrasted with the new threat vectors that are created in this type of network. Kreutz et al. outlined seven SDN threats [196]. The following list describes the security issues found by [196] as well as pertinent research related to resolving each threat.

1. **Forged or Faked Traffic Flows:** Although not entirely a SDN issue, the SDN architecture adds a new dimension to this classic problem. For example, a centralized controller causes an inherent bottleneck between the control and data planes as noted by [24] and [289]. This bottleneck is a prime target for distributed denial-of-service (DDoS) attacks. Braga et al. [31] proposed a DDoS detection method based on traffic flow that lacks a prevention mechanism. The findings from Brenton et al. [24] suggest that some DDoS attacks can be mitigated by using multiple controllers, though flow rules will have to be conscientiously crafted so that no one controller becomes overwhelmed. Shin et al. [289] proposed AVANT-GUARD, which is inspired by the mitigation of Transmission Control Protocol (TCP) `SYN` flooding attacks through the use of `SYN` cookies. AVANT-GUARD helps defeat control plane saturation attacks, which occur when an attacker maximizes the traffic sent between the controller and the switches [289]. Other solutions to prevent DDoS attacks include a method of leveraging SDNs within a cloud environment [324] and OF-GUARD [325], which aims to prevent control plane saturation attacks by using packet migration and a data plane cache.

2. **Vulnerabilities in Switches:** While this also is not solely a SDN threat, this issue can be adapted to the SDN environment. For instance, a malicious switch can ignore new OpenFlow requests, flood the controller with packets, or slow down network traffic. Some trust-based mechanisms may be used as a potential solution [332], but unforeseen software bugs and the lack of Transport Layer Security (TLS) support suggest the need

for a thorough review by switch vendors [24]. This issue is discussed in more depth in Section 4.3.1.

3. **Attacks on Control Plane Communications:** The lack of TLS support by switch vendors leaves many controllers vulnerable to man-in-the-middle (MITM) attacks [24]. Furthermore, even encrypted communications rely on a trusted root, which leaves a single point of failure. Therefore, there must be some mechanism to guarantee trust between the controller and its managed switches. One such mechanism could be an oligarchical model such as voting [123], [254], although this has not yet been applied to SDN.

4. **Vulnerabilities in Controllers:** Because the controller is the cornerstone of a software-defined network, vulnerabilities allow an attacker to gain full control over the network. Therefore, it is vital to provide integrity and secrecy guarantees for flow tables. A clean separation from applications and a formal verification of controller correctness are crucial aspects of these guarantees. Porras et al. [263] introduced FortNOX, which provides role-based authorization and security constraint enforcement in the NOX controller. However, the project has since been ported to FloodLight and renamed SE-Floodlight due to the discontinued development of NOX [261]. FortNOX provided important initial research into secure SDN controllers by preventing the circumvention of flow rules by malicious applications. Additions made to SE-Floodlight include the mediation of all communication between an application and the data plane [262]. Wen et al. [327] presented a short paper outlining a fine-grained permission system for OpenFlow. However, the paper lacks details and does not address how to retrofit this model into an existing controller. Future research should apply lessons from work in operating system security to ensure that access control and secure information flow are built into controllers [318].

5. **Controller and Application Trust:** This problem is interrelated with Threat 4 because malicious applications are a likely avenue for exploiting controller vulnerabilities [263]. However, the distinction is in how the certification of applications should occur. A controller with autonomic trust management assures that the application remains trusted throughout its lifetime [196]. Scott-Hayward et al. [279] secure northbound APIs by developing a permissions system, guaranteeing controller operations are only accessible via trusted applications.

6. **Vulnerabilities in Administrative Stations:** Many controllers are not stand-alone operating systems, including Floodlight [100], NOX [228], and Beacon [81]. Therefore, it is vital to protect the underlying systems on which these controllers operate. This is not a unique problem to SDN and traditional methods of isolation, updates, system hardening, and secure credentials can limit the impact of this threat [196], [318].

7. **Lack of Forensics and Remediation:** To assist in investigating security incidents, SDNs need stronger logging mechanisms. Furthermore, a strong security policy and protection mechanisms that ensure the integrity and authenticity of the data is vital [196]. However, SDN's central view of the network can be leveraged to improve its forensic capabilities [21].

In addition to Kreutz et al., there are several other comprehensive SDN and OpenFlow security surveys including [24], [190], [195], [280], and [318]. In particular, [190] provides an analysis of OpenFlow using the STRIDE methodology [314] and attack tree modeling. Regarding SDN within a testbed setting, Li et al. [199], [200] conducted a security evaluation of ProtoGENI [265], a prototype GENI implementation based largely on Emulab. The authors notified GENI engineers about the vulnerabilities found in the study, all of which were addressed immediately.

Chapter 3

Case Studies

The primary focus of this thesis is to compare and contrast currently deployed software-defined networks to discover successes and limitations, which will improve future SDN deployments. In particular, we examine campus and research networks. In this chapter, we discuss specific examples of SDN technologies utilized in research networks, the process of building a software-defined research network, and the challenges faced by network architects. We present an in-depth review of hardware, software, and design decisions in GENI, OFELIA, Stanford's production SDN, and the future Microelectronics Center of North Carolina (MCNC) SDN testbed. It should be noted that the MCNC network outlined in Section 3.5 is currently in the implementation phase and is subject to change. Regardless, this network is an informative example of a recently designed OpenFlow research network.

3.1 Research Methodology

To assess issues occurring in software-defined networks, we examine existing and future SDN deployments. Our case studies are testbeds that use SDN as an integral part of their implementation. We chose GENI, OFELIA, Stanford, and MCNC, from this pool of candidates based on the following criteria: longevity of testbed deployment, amount of publicly available documentation, diversity of SDN implementation, network size, and influence on SDN research.

The length of time a testbed has been active was a major factor. A longer deployment time allows the testbeds to mature and evolve. This provides valuable feedback on initial assumptions, thus preventing potential pitfalls for future designers. Available documentation was relied upon heavily to discover the nuances of each testbed's functionality as well as any obscure deployment issues. This thesis draws from publications, website documentation, design specifications, test plans, bug reports, and presentations to uncover details that will assist future testbed designers. GENI and OFELIA are federated testbeds; GENI uses rack-based nodes and

OFELIA utilizes islands consisting of heterogeneous sets of equipment. Stanford is a campus network and involves a production OpenFlow deployment. Lastly, MCNC is a regional network designed for OpenFlow v1.3, which is at the beginning of its implementation phase. The selection process also accounted for testbed size. The current study explores only testbeds that are "sufficiently large." That is, any testbed that is contained within a single research lab or is entirely virtual (i.e. using only OVS, Mininet, etc.) was eliminated from consideration. Larger networks provide greater opportunities for implementation challenges such as interoperability. Finally, the testbed's influence on SDN research was also considered. The chosen networks, except for MCNC, have produced extensive research on software-defined networking, thereby providing insight into limitations of the existing hardware, software, protocols, and network design.

3.2 GENI

The Global Environment for Network Innovations (GENI) is a virtual testbed sponsored by the NSF for the creation, implementation, and testing of innovative networking solutions [26]. GENI is unique among research testbeds for several reasons: 1) it provides researchers access to a large-scale infrastructure unavailable at many research institutions, 2) it offers fine-grained control over the networking stack by allowing researchers to create layer 2 connections and send arbitrary layer 3 packets over this infrastructure, 3) it permits researchers to program all elements of an experiment, including the network core, 4) it allows repeatable experiments through fine-grained control over an experiment's environment, and 5) it provides tools for implementing experiments and for measuring, visualizing, and analyzing research outcomes [1]. GENI is a national network that spans approximately fourteen research institutions and is planning to grow to include about 100 different campuses worldwide. Figure 3.1 is a map of the locations of existing and proposed GENI resources [26].

Software-defined networking is a deeply ingrained aspect of GENI's infrastructure. For example, the two key concepts that provide flexibility for experiments in GENI's environment are *sliceability* and *deep programmability* [26]. As explained in Section 2.1.2 *sliceability* is not a new concept, but it can be enhanced through the use of SDN tools like FlowVisor [288]. In addition to using SDN to create a slice, GENI experimenters can choose to incorporate SDNs within their slice [26]. *Deep programmability* allows a researcher to control almost all aspects of the experiment including the network, storage, and computational resources [26]. SDN technologies also have impacted this component of GENI and offer highly programmable and efficient networks using commodity hardware [26].

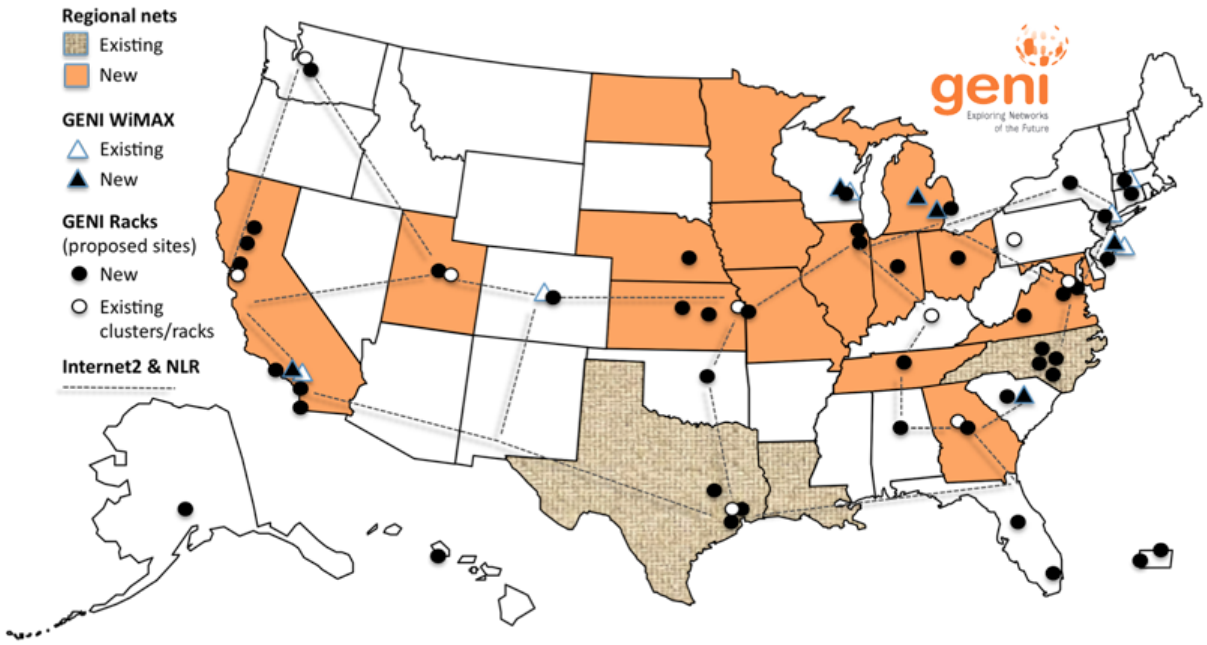


Figure 3.1: A map of current and proposed GENI sites [1].

3.2.1 GENI Design

The GENI network architecture is primarily composed of two components: the GENI rack and the GENI core. GENI racks contain compute, storage, and SDN resources. The GENI network core is a subset of Internet2 nodes. Some campuses also provide WiMAX base stations and, while this is beyond the scope of this thesis, an interested reader can find out more about WiMAX research using GENI in [239].

GENI Racks

GENI racks are the core units that contain the compute and network resources needed to connect to the GENI backbone. The rack (or racks in some deployments) will typically resemble Figure 3.2, though some variations exist. Currently, there are three rack types that implement all the GENI requirements: InstaGENI, OpenGENI, and ExoGENI. However, there is a fourth rack, the Cisco GENI Rack, that is undergoing the acceptance testing phase and, therefore, will also be discussed below.

InstaGENI Rack InstaGENI is an easily deployed and extensible GENI rack solution that typically is located outside of a campus network. It includes support for both OpenFlow and VLAN-based networking [60]. The OF switch used in the InstaGENI rack is an HP

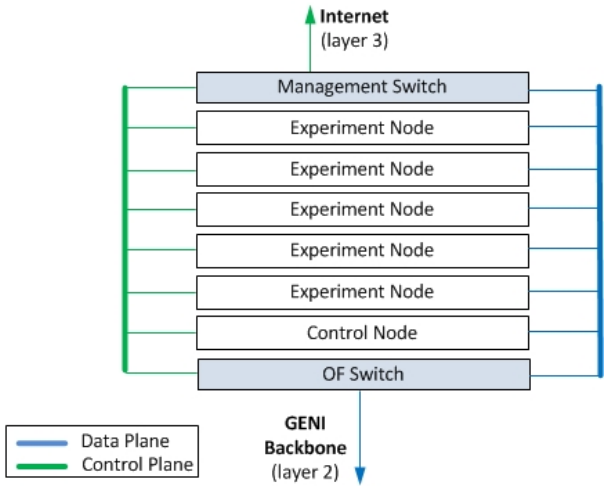


Figure 3.2: InstaGENI rack components [60].

ProCurve 6600 (now E-Series) that has forty-eight 1G ports and connects to the GENI core over a 10G link [207]. Administrators can manage rack hardware using several aggregate managers (AMs) including FlowVisor OpenFlow Aggregate Manager (FOAM) [249], ProtoGENI [265], and PlanetLab [55] [20]. InstaGENI offers OpenFlow support for connections within the rack as well as to external devices. For internal connections, InstaGENI takes advantage of the ProCurve's ability to enable OpenFlow on a per-VLAN basis, only using OpenFlow if an experimenter requests it for the slice [207]. However, slices must be based on either OpenFlow or VLANs, as hybrid solutions do not currently exist [207]. This VLAN configuration is enabled through SNMP, while OF-enabled VLANs will use FOAM as their controller [207]. By managing OF-enabled VLANs, FOAM maintains security and isolation between slices [207]. External OpenFlow connections use a single controller to manage all OF switches [207]. However, a new architecture being tested allows multiple controllers to be used in an environment, although only a single controller could be used for any given OF domain [207]. This is a major limitation of many OpenFlow testbeds and is discussed further in Section 4.3.2.

OpenGENI Rack Built to support cloud applications, OpenGENI is a Dell-centric rack that includes both OpenFlow and VLAN networking [60]. This newer rack is currently a provisional resource, indicating that it is connected to GENI for further testing [60]. Dell is the hardware vendor for all rack components including the OpenFlow enabled Force10 S4810, which supports forty-eight 10G and four 40G ports [60]. All of the OpenGENI compute nodes are managed by the control node using both OpenStack and the GENI Rack Aggregate Manager (GRAM) [250]. The four networks required to use this rack are the control plane, data plane, external plane, and management networks [250]. However, only the first two of these networks are relevant to this

thesis. The control plane network is not managed by OpenFlow and allows various management traffic, including OpenFlow commands, to pass between nodes. In contrast, the data plane network is controlled entirely by OpenFlow and sends traffic between virtual machines [250]. Both of these networks function similarly to traditional SDN control and data plane networks. OpenGENI uses parts of the POX OpenFlow controller toolkit to create OF functionality [250]. However, only packets sent by VMs on different compute nodes will pass through the OpenFlow switch, eliminating the OF switch's ability to impact all network traffic [250].

GRAM provides two key SDN components: `gram-ctrl`, a default OF controller that uses layer 2 learning and `gram-vmoc`, which is a VLAN-based Multiplexed OpenFlow Controller (VMOC). VMOC multiplexes on VLAN tags and switch data path ID (DPID)/ports allowing it to behave as a controller to switches and as a standard switch to any researcher specified controller [114]. Figure 3.3 depicts this unique behavior. Figure 3.4 provides a more in-depth description of how VMOC uses multiplexing to identify the correct flow of packets [114]. One

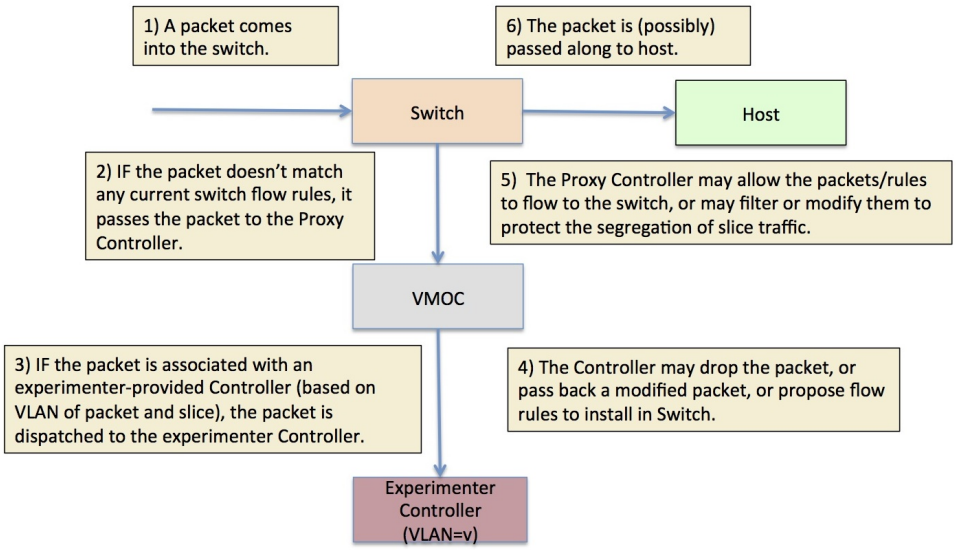


Figure 3.3: Managing controller/switch interactions using VMOC [114].

interesting aspect of VMOC occurs in step 5 of Figure 3.3. At this stage, the VMOC ensures that the controller response preserves the VLAN isolation of the slice [114]. VMOC's API allows GRAM to control the creation, deletion, or modification of network slices as well as other control commands [114].

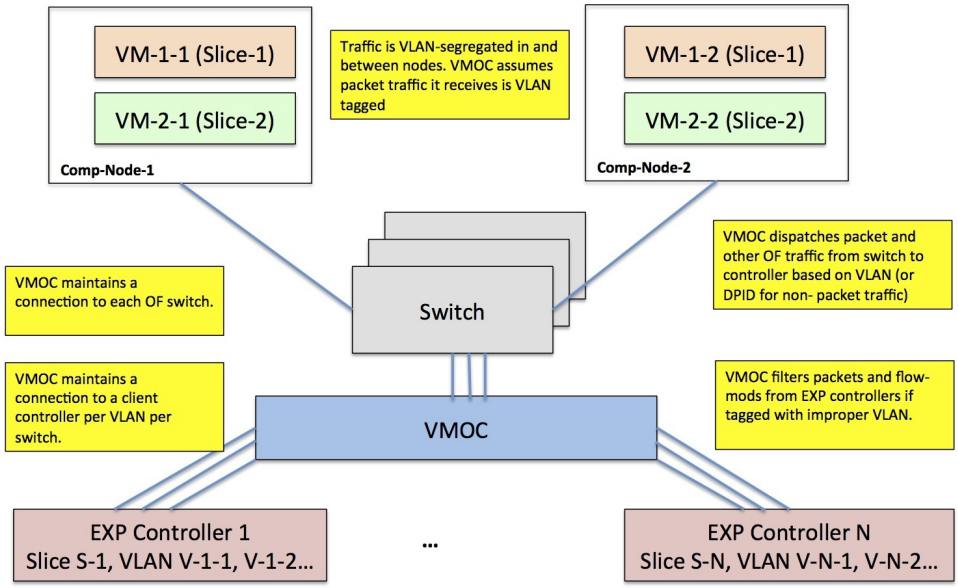


Figure 3.4: Multiplexing switches and controllers by VLAN using VMOC [114].

ExoGENI Rack ExoGENI is a more expensive rack that is assembled by IBM, which allows for multi-site cloud applications and can be integrated into an existing campus network [60]. The rack components include an IBM G8264R OpenFlow-enabled switch which has a 40G uplink to the GENI backbone as well as 10G Network Interface Cards (NICs) [60]. This OF switch can send experiment traffic to the GENI backbone or to an OpenFlow enabled campus [60]. Open Resource Control Architecture (ORCA) [109], the main control framework for ExoGENI, and FOAM interact with an instance of FlowVisor, located on the rack's head node, to create OpenFlow slices [85]. OpenFlow is a direct component of ExoGENI rack aggregates as opposed to other GENI rack implementations that regard OpenFlow as a separate aggregate [18]. An experimenter is free to designate an OF controller, which will be automatically authorized by the ExoGENI aggregates [18]. This controller manages traffic and flows on its specific network slice [18]. However, all network slices are created with VLANs using ORCA's coordination with the layer 2 switch [85].

Cisco GENI Rack There is a fourth GENI rack being developed by Cisco that, at the time of this writing, is undergoing GENI Acceptance Testing [60]. It will use Cisco UCS-B and C series servers, NetApp FAS 2240 storage capabilities, Cisco Catalyst switches for the management network, an ASA firewall for virtual private network (VPN) and Internet Protocol Security (IPSec), and a Cisco Nexus 3000 series switch to provide OpenFlow connectivity [60]. ExoGENI's software, ORCA, will be used as the main software stack for this rack. The Nexus

switch, NetApp storage, and UCS servers make this rack ideal for researchers that want to run data center, OpenFlow, or even multi-site cloud application experiments [60]. Furthermore, the current design allows existing racks to be easily expanded to provide more network or compute power [105]. Because the rack is still undergoing acceptance testing, it is possible that future versions of the rack will include the more powerful Nexus 7000 [105].

Rack Security While each of the above GENI racks implement their own security measures, the GENI Meta-Operations Center (GMOC) offers several suggestions for controlling experimental GENI traffic that may enter an internal campus network. Suggestions include: access control lists (ACLs), firewalls, device configuration changes on the data plane switch, and resource approval by an administrator [110]. However, the last two solutions are not ideal because they allow a local administrator to modify specific GENI rack functions [110]. Typically, experiment resource requests are automatically granted by the aggregate manager but, FOAM, the OF aggregate manager, can be configured to delay incoming requests until approved by a site administrator [110]. Additionally, only resources that use OpenFlow can be required to have local administrative approval for requests [110].

GENI Network Core

The GENI network core uses a set of OpenFlow compatible switches within Internet2's network [108]. Previously, the backbone was also connected to the National LambdaRail (NLR), but that section has been removed due to the closure of the NLR [108]. The backbone VLANs (3715, 3716, and 3717) are not extended beyond the backbone edge and are not interconnected within the core [108]. These VLANs connect New York City, Washington D.C., Atlanta, Houston, and Los Angeles using a broken ring topology, except for VLAN 3717 (shown in Figure 3.5), which uses a ring topology [108].

The entire GENI backbone integration network is shown in Figure 3.6. The Internet2 OF switches are able to use SDN to forward traffic at layer 1 [108]. That is, a core administrator can define and preserve pseudowire-like connections using various attributes such as DPID, VLAN ID, or subnet [107]. GENI provides its own layer1transport software that uses the Floodlight OpenFlow controller and its static flow pusher service [107]. This combination of software makes it easy to create, delete, or modify administrative flows that move traffic through the backbone OpenFlow switches [107].

3.2.2 GENI Use Cases

GENI's SDN capabilities make it an ideal testbed for new research experiments. This section will highlight a few of the projects that have utilized GENI OpenFlow equipment. Two stud-

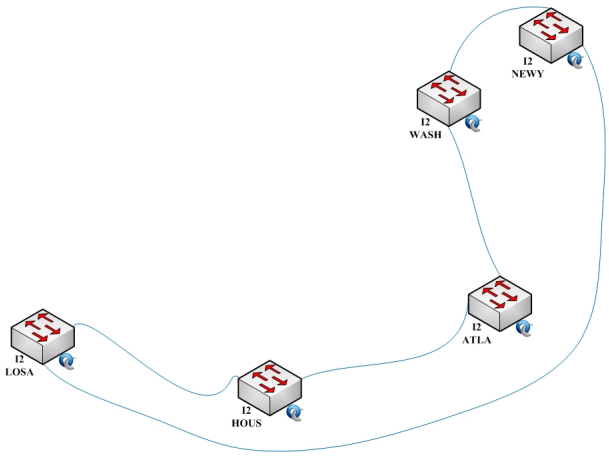


Figure 3.5: VLAN 3717 in the GENI backbone network [108].

ies looked into the management opportunities of OpenFlow. First, Huang et al. [153] presents a framework that unifies the policies that apply to integrated Rule Oriented Data Systems (iRODSs) and software-defined networks. GENI, specifically ExoGENI, provided the SDN resources necessary to develop and test this system [153]. Next, Malishevskiy et al. [204] created a new network manager that allows a user to visualize and manipulate OpenFlow parameters. GENI also serves as a useful platform for orchestrating virtual desktop resources as shown in [44]. By using a multi-domain OpenFlow environment in GENI, the researchers were able to repeatedly demonstrate that OpenFlow improves path selection and load balancing between thin-clients and virtual desktop clouds [44]. GENI-VIOLIN is a project that developed an in-network distributed snapshot by using OpenFlow [185]. This system allows for easy recovery of entire virtual infrastructures and minimizes modifications to any VM servers [185]. By using GENI, GENI-VIOLIN had access to the OpenFlow resources and networking infrastructure to verify its application with real-world hardware [185]. These examples illustrate the wide range of applications for which researchers can use the GENI testbed. Furthermore, they demonstrate the imperative nature of SDN resources to the future of networking research.

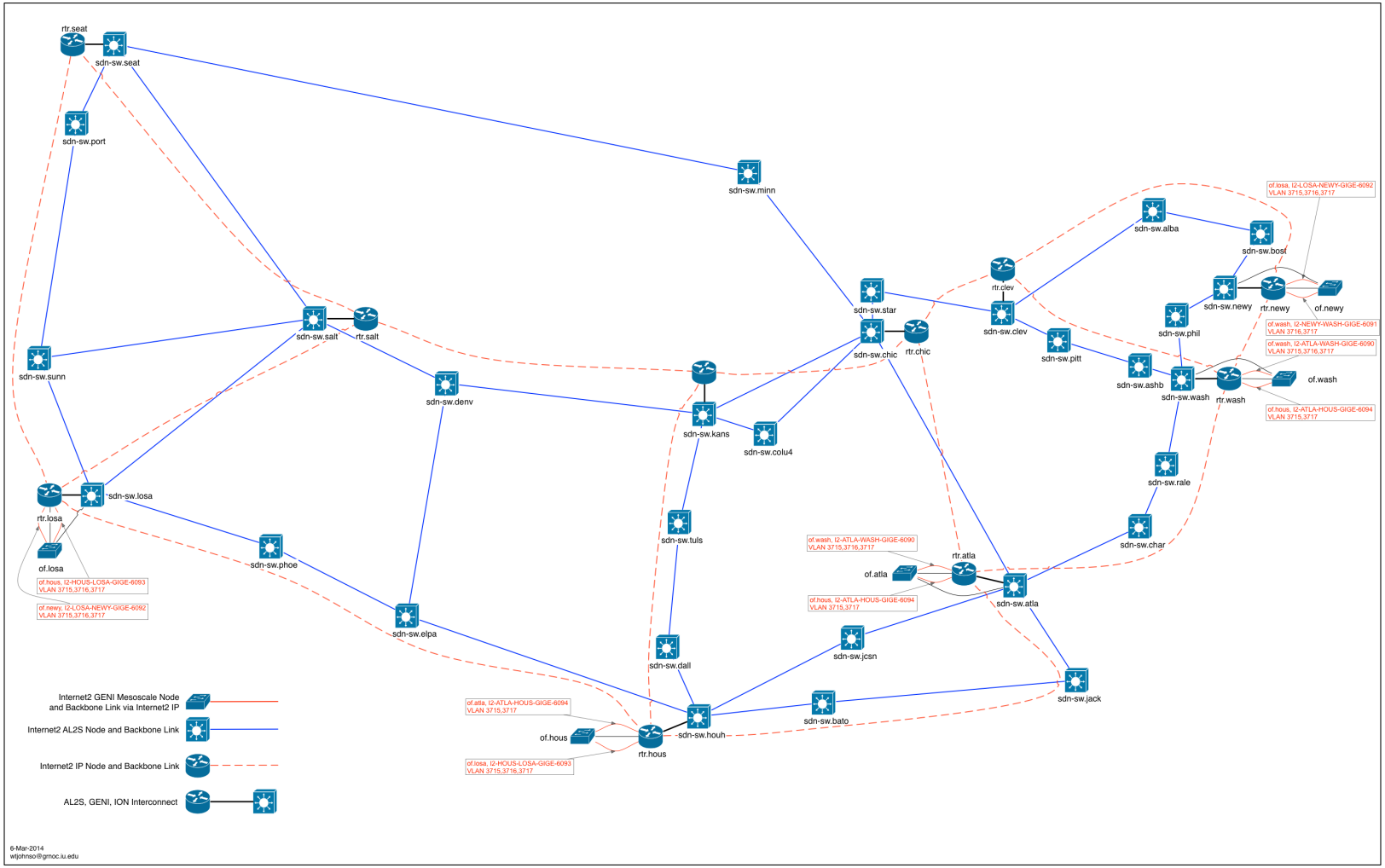


Figure 3.6: GENI backbone network [108].

3.3 OFELIA

OpenFlow in Europe: Linking Infrastructure and Applications (OFELIA) is an international testbed designed to bridge theoretical next-generation Internet architectures and the practical operation of that research [301]. A major focus for the OFELIA testbed was implementing software-defined networking and OpenFlow. OFELIA was completed in 2011 and remains operational and free to use for interested researchers [301]. This testbed, shown in Figure 3.7, has ten islands or individual research testbeds. These islands are interconnected at layer 2 using either VPN connections over the Internet or through GÉANT [120], a pan-European interconnection of research and educational networks [301].

Figure 3.7: Map of OFELIA islands [232].

Each island utilizes a different set of equipment and a unique topology, allowing OFELIA researchers to perform various experiments at multiple layers of the networking stack. The following is a list of the island locations, the testbed name, and a brief description of each testbed:

- **Berlin, Germany (TUB):** This mesh network uses four NEC IP8800/S3640 switches

and one HP 5400, all of which are OF capable [192], [307].

- **Ghent, Belgium (IBBT):** This testbed is built as the central hub for OFELIA and uses a combination of NEC OF-switches and OVS to specialize in large-scale emulation experiments [232], [306].
- **Zurich, Switzerland (ETH):** Three NEC IP8800/S3640 switches are used to create this mesh network. Additionally, there is an optional connection to Great Plains Environment for Network Innovation (GpENI) [113], a GENI project [312].
- **Barcelona, Spain (i2CAT):** This network uses both NEC and HP OF-Switches and has a reconfigurable optical add-drop multiplexer (ROADM) ring [232], [305].
- **Bristol, UK (UNIVBRIS):** This testbed includes four NEC IP8800/S3640 switches to manage the OpenFlow campus network, an optical testbed with equipment from ADVA and Calient, and a field-programmable gate array (FPGA) testbed. It is connected to Internet2, GÉANT, and JANET via three Extreme Black Diamond 12804R carrier grade switches [232], [309].
- **Catania, Italy (CNIT):** This is the first of two islands that use NetFPGAs with Open vSwitch to focus on information-centric networking (ICN) research [212], [232], [310].
- **Rome, Italy (CNIT):** The second of two islands that use NetFPGAs along with Open vSwitch, this testbed focuses on ICN research [212], [232], [310].
- **Trento, Italy (CREATE-NET):** This testbed is a city-wide distributed island that uses NEC switches and NetFPGAs for OpenFlow capabilities. It includes MPLS equipment and a wireless mesh network that can be used by researchers [232], [311], [317].
- **Pisa, Italy (CNIT):** This island uses NetFPGAs in combination with Open vSwitch to conduct research about cloud and data center environments [232], [310].
- **Uberlândia, Brazil (UFU):** This island uses Datacom switches, NetFPGAs, and Extending and Deploying OFELIA in Brazil (EDOBRA) switches [234]. EDOBRA is a software switch based on OpenWRT that is customized to use Media Independent Handover (MIH) and the OpenFlow protocol [234]. This testbed's research focuses on new network architectures, such as the interaction of multicast and mobility within heterogeneous networks [232], [308].

3.3.1 OFELIA Design

This section will focus on the three main SDN-related components of OFELIA's design. The first involves the overall network design of OFELIA and how it utilizes OpenFlow. The second component relates to the adaption of optical network equipment to use OpenFlow. Finally, we explore the software that defines OFELIA experiments and controls OF slices. It should be noted that because of the autonomy of the island typologies, Section 3.3.1 will not examine this topic. However, we include an example of an island topology in Figure 3.8, which illustrates the network design for the TUB testbed.

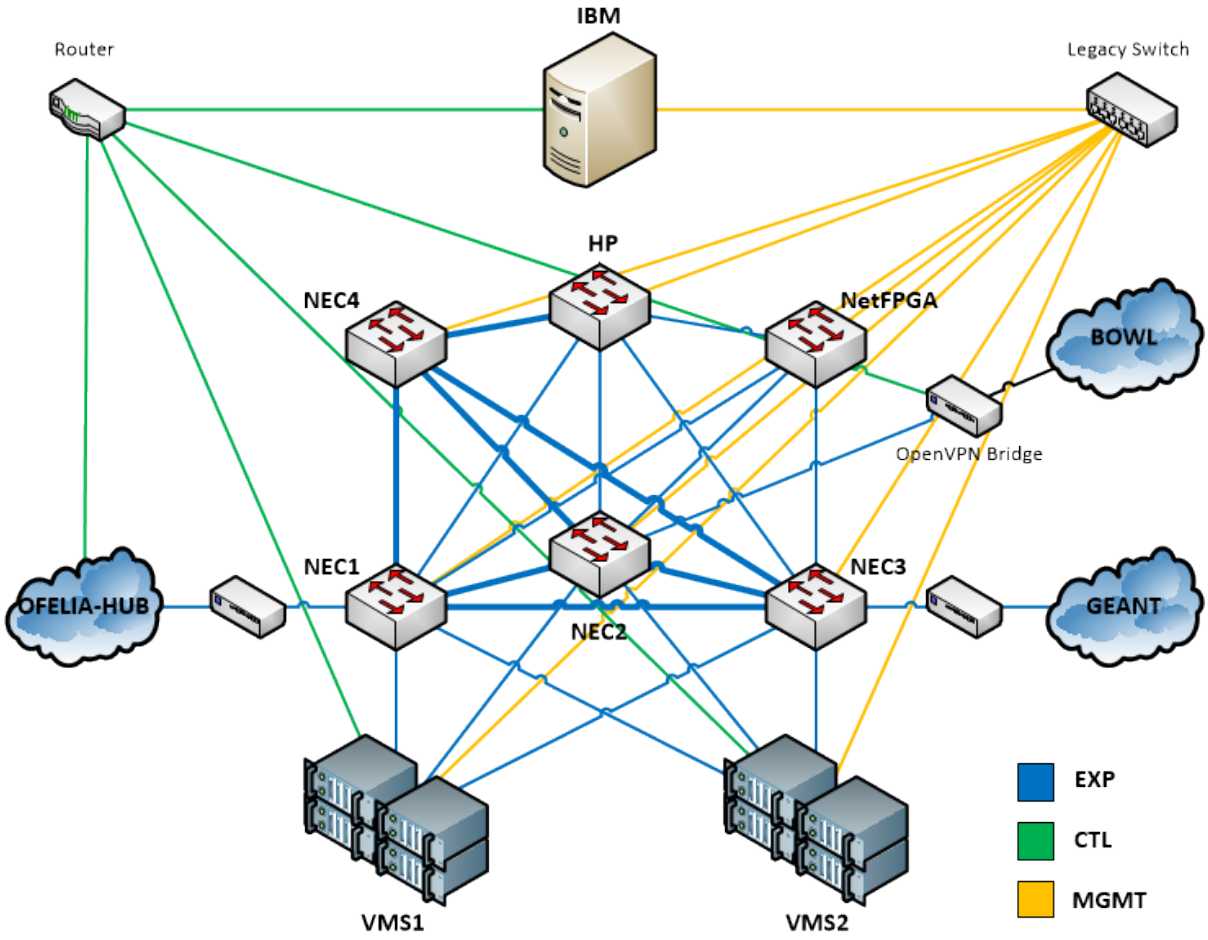


Figure 3.8: Physical network topology for the TUB island [307].

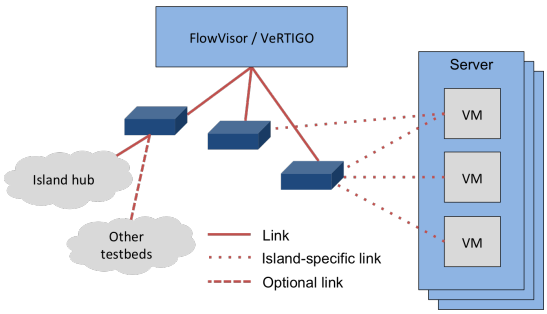
OFELIA Network Design

OFELIA operates with three networks, including an experimental, a control, and a management network. Researchers use the experimental network to control layer 2 OpenFlow network slices. Shown in Figure 3.9a, the experimental network allows access not only to VMs, but also to a diverse set of equipment supplied by each island. For example, access to optical equipment is offered by the UNIVBRIS island and a wireless mesh testbed is provided by CREATE-NET [301]. While each island's topology is independent, most are designed to accommodate arbitrary equipment allocation by using partial or full mesh typologies. In addition to a mesh of OpenFlow switches, many islands also connect the VM servers to multiple switches [301]. Both of these design decisions are illustrated in the TUB topology shown in Figure 3.8. Overall, the experimental network consists of a heterogeneous set of OpenFlow v1.0 enabled switches from NEC, HP, NetFPGA, and other vendors [301]. The islands are interconnected by using a dedicated 1G layer 2 circuit to the GÉANT network or VPN tunnels over the Internet.

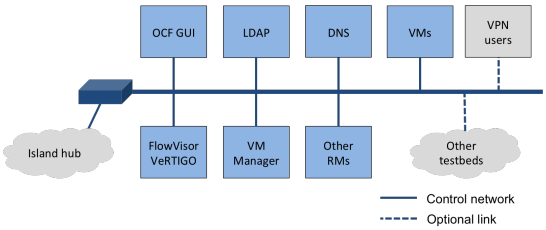
OFELIA uses slices to isolate experiments throughout the testbed and uses FlowVisor [287] and VeRTIGO [75] to assist in the network slicing. Currently, OFELIA uses VLAN ID-based slicing. However, designers assessed other slicing techniques such as media access control (MAC)-based slicing and flowspace slicing before choosing a method [301]. Experiments that need to use VLANs are assigned a group of VLAN IDs, which allow for VLAN use within the experiment while maintaining slice isolation [301]. Islands are also free to use a different slicing technique when intra-island experiments are conducted [301]. Furthermore, this design decision allows the flexibility to change from VLAN ID-based slicing to another technique in the future [301].

The control network is used by the experimenters and management staff to support a wide range of OFELIA services, including experimental resources, access to the OFELIA Control Framework (OCF), and internal networking services (i.e. Domain Name Service (DNS) and Lightweight Directory Access Protocol (LDAP)) and non-OFELIA resources, such as island-specific infrastructure [301]. Additionally, the control network is available through a VPN, allowing researchers access via the Internet [301]. The OFELIA control network is illustrated in Figure 3.9b. This network uses private IPv4 addressing, assigns a unique subnet to each island, and manages its link advertisements with the OSPF protocol [301].

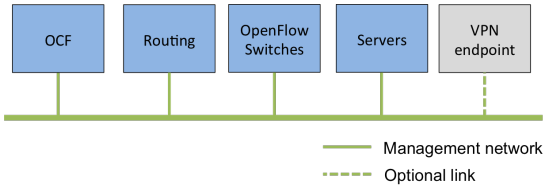
The management network, shown in Figure 3.9c, is only used by administrators for infrastructure setup and maintenance [301]. This network does not span the entire OFELIA topology and each island can provide a unique management network [301]. The OFELIA requirements indicate that the management network is not required and its duties can be subsumed by the control network. However, implementing a management network is highly recommended so that administrative and management traffic are properly segmented from user traffic.



(a) Available services in the experimental OpenFlow network.



(b) Available services in the control network.



(c) Typical resources administered in the management network.

Figure 3.9: Available services in the OFELIA testbed networks. (Adapted from [301].)

Optical Network Adaptations

A unique focus of the OFELIA testbed is using optical networking in conjunction with OpenFlow and SDN. All of the optical networking equipment in OFELIA has been adapted to use OpenFlow by implementing v0.3 of the circuit switch addendum to the OpenFlow specification [301]. To function with optical devices, OF was extended to match on a wide range of optical flow characteristics, such as wavelength, bandwidth, or power constraints [301]. Implementing OF on the optical switches required developing a software agent that abstracts SNMP functions and maps them onto OF commands. Next, an OpenFlow controller must be modified to use the new optical functions. The current controller is based on the software for ADVA's FSP 3000 ROADM equipment [2]. Once an OpenFlow controller has been modified to work with the optical equipment, it is able to perform topology discovery and manage both packet and circuit switched OF networks [301]. The extended controller can allocate network resources by using standard OpenFlow or a version of OpenFlow that has integrated Generalized Multi-Protocol Label Switching (GMPLS) within the control plane [301]. Finally, once the switches and controllers were modified, researchers created Optical FlowVisor (OFV) [14], which virtualizes an optical circuit network so that it can be controlled using the specialized OF controller. This virtualization enables users to slice the optical network based on wavelength and port [301].

OFELIA Control Framework

The OCF is based on the control functionality of other OF testbeds, such as GENI. The designers wanted researchers to be able to allocate resources, initialize experiment slices, and manage projects, while maintaining scalability, usability, island autonomy, and strong policy enforcement. The OCF currently has two architectures: v0.X and v1.X. At the time of this writing, v0.8.1 is in use but over the course of using the initial v0.X version of the framework, the designers made adjustments to the system. The OCF is an open source project and more information about its current feature set can be found in [231].

OCF v0.X The designers created this initial architecture based on an implementation-centric model that was designed for a single testbed deployment [301]. It has been updated to allow for a federation of different OCF testbeds. The architecture consists of two parts. First, the architecture includes a monolithic front-end, which contains a graphical user interface (GUI) that experimenters can use, as well as a "Clearinghouse" that contains state information, project information, and provides user authentication and authorization [301]. This front-end is based on Expedient [217], which was originally used in GENI, but has been replaced with FOAM. The second section is divided into resource managers (RMs) and aggregate managers (AMs). The RMs are used to allocate, manage, and monitor resources (VMs, OpenFlow switches, etc.). In contrast, the AMs are responsible for allocating, managing, and monitoring an amassed set of resources, known as an aggregate [301]. The segmentation between the front-end and the RMs/AMs allows for the federation of different instances. This occurs when connecting a front-end from one testbed to the resource and aggregate managers of another [301].

This initial implementation-centric architecture presents several issues. The monolithic front end does not allow for additional user interfaces to be utilized [301]. This lack of flexibility can be addressed by using a modular interface, which will separate the Clearinghouse from the GUI [301]. Furthermore, APIs should be formally defined to ensure clear communication among components, allowing for a flexible and extensible system [301]. Formal definitions of module duties are lacking also in the current system, leading to confusion about how resource aggregation is performed and where policies should be implemented [301].

OCF v1.X This new architecture is focused on fixing issues with the existing OCF as well as implementing ideas that have worked in other federated testbeds, such as GENI. The architecture splits the Clearinghouse from the GUI to create three formally defined layers [301]. In addition, all interactions between layers are clearly defined, alleviating the confusion of the v0.X software [301]. In this update, AMs/RMs can connect to other AMs/RMs in a hierarchical chain by using common, well-defined APIs [301]. This new feature allows for AM/RM interactions to

be translated easily to work with other testbed interfaces, such as the GENI API [301]. Finally, v1.X defines an accessible interface for administrators to add local policy rules to an AM or RM, if needed [301]. The OCF team is taking an agile, iterative development style and slowly implementing more features of this new architecture in each development cycle [301].

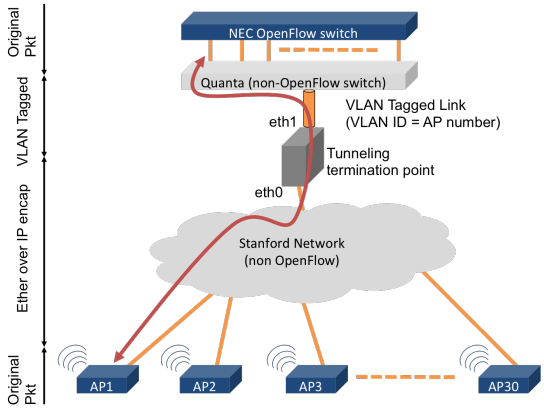
3.3.2 OFELIA Use Cases

OFELIA has provided not only a great environment for next-generation Internet architectures, but also for advancing OpenFlow research. Advancements in both network slicing and optical SDNs have been made with the help of OFELIA. First, researchers have developed VeRTIGO [75], an extension to FlowVisor previously discussed in Section 2.1.2, which allows more flexibility in choosing a network slice. In FlowVisor, a slice cannot use an arbitrary virtual topology and must conform to the existing physical topology [301]. This is a major limitation for researchers who want to use a topology that spans multiple islands that differs from the existing physical infrastructure. VeRTIGO removes this limitation by creating virtual links that can be used in topology creation [301]. Second, great strides in using SDN techniques with optical networks were made through OFELIA. This is outlined in Section 3.3.1 and in more depth in [13], [14], [50], and [51].

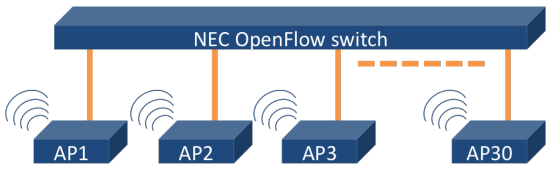
Information-centric networking (ICN) is a next-generation Internet idea that restructures the network into "content" elements with unique names and offers the ability to put and retrieve these elements within the network. OFELIA is a beneficial resource for researchers working on ICNs in combination with SDN. Veltri et al. [321] and Melazzi et al. [212] use OFELIA for this type of innovative research. Their foundational work demonstrates that SDNs allow for the creation of flexible ICN solutions, while using real network infrastructure. Work by Salsano et al. [273] demonstrates that SDNs can provide efficient content packet routing by using an OpenFlow controller to determine whether the packets go to a caching server or the content server. An in-depth look at various ICN/OpenFlow experiments is presented in [274].

3.4 Stanford

Stanford is credited with playing a significant role in the SDN revolution, beginning with their 2008 paper introducing OpenFlow [209]. Through developing and implementing OpenFlow, Stanford researchers gained invaluable knowledge about and experience with SDN technologies. Stanford researchers outline their OpenFlow journey in [191]. We will focus on the production deployment of OpenFlow as it incorporates later versions and alleviates the struggles of the initial protocol development. The Stanford production SDN is divided into two parts: an OpenFlow Wi-Fi network and a production OpenFlow network that spans several buildings [191].



(a) The physical topology of *ofwifi* tunneling.



(b) The logical topology, e.g. the controller's view, of *ofwifi* tunneling.

Figure 3.10: The *ofwifi* tunneling topology. (Adapted from [210].)

3.4.1 Stanford Design

Stanford OpenFlow Wi-Fi Network

Stanford's OpenFlow Wi-Fi network, called *ofwifi*, is deployed in the Gates Computer Science Building [191]. This network consists of many OF wireless access points (APs). Of these APs, four are directly connected to an OF switch [191]. The rest of the APs use Capsulator [45], a user space-based software tunneling program, to connect to an OF switch [210].

This tunneling was a critical part of the initial design and provides insight into future hybrid OF networks. One challenge of tunneling involves directing each AP's traffic into a different physical port of the OF switch [210]. Figure 3.10a illustrates Stanford's solution to this problem. The researchers terminated all tunnels at a computer that VLAN-tagged the packets using the access point ID as the VLAN ID [210]. The VLAN-encapsulated packets were then sent to a standard switch, which removed the VLAN encapsulation and forwarded the packets to different physical ports of the OF switch [210]. Figure 3.10b illustrates how the network controller views the access points. One minor issue with this approach was the intermediary switch (Quanta) dropping Link Layer Discovery Protocol (LLDP) packets that are needed by the OF controller [210]. This was remedied easily by using a non-standard LLDP destination MAC address [210].

The *ofwifi* network is utilized by as many as twenty users during peak hours, allowing for ample feedback [191]. Because there are other wireless networks in the building, any flaws found in *ofwifi* do not cause significant issues for users. This allows researchers to test updates and experiment with various components [191]. However, after many improvements to the software, the network has stabilized and remains a viable Wi-Fi option for users [191].

Stanford OpenFlow Production Network

The McKeown group at Stanford initially implemented the production network to monitor and enhance OpenFlow technologies [191]. Figure 3.11 shows the logical topology used in the Gates building during Stanford's initial SDN deployment. The network used a hodgepodge of OF v1.0 capable switches including HP, Pronto, NEC, and NetFPGA [191]. Originally, they controlled the OF switches with the Simple Network Access Control (SNAC) controller [290], based on NOX, and later began using the Big Network Controller [28], a commercial controller developed by BigSwitch [191]. After testing and improving the initial network, Stanford administrators deployed OpenFlow switches in two additional buildings. A full building deployment means that OpenFlow access points no longer need to use the tunneling work-around for connectivity to an OF switch.

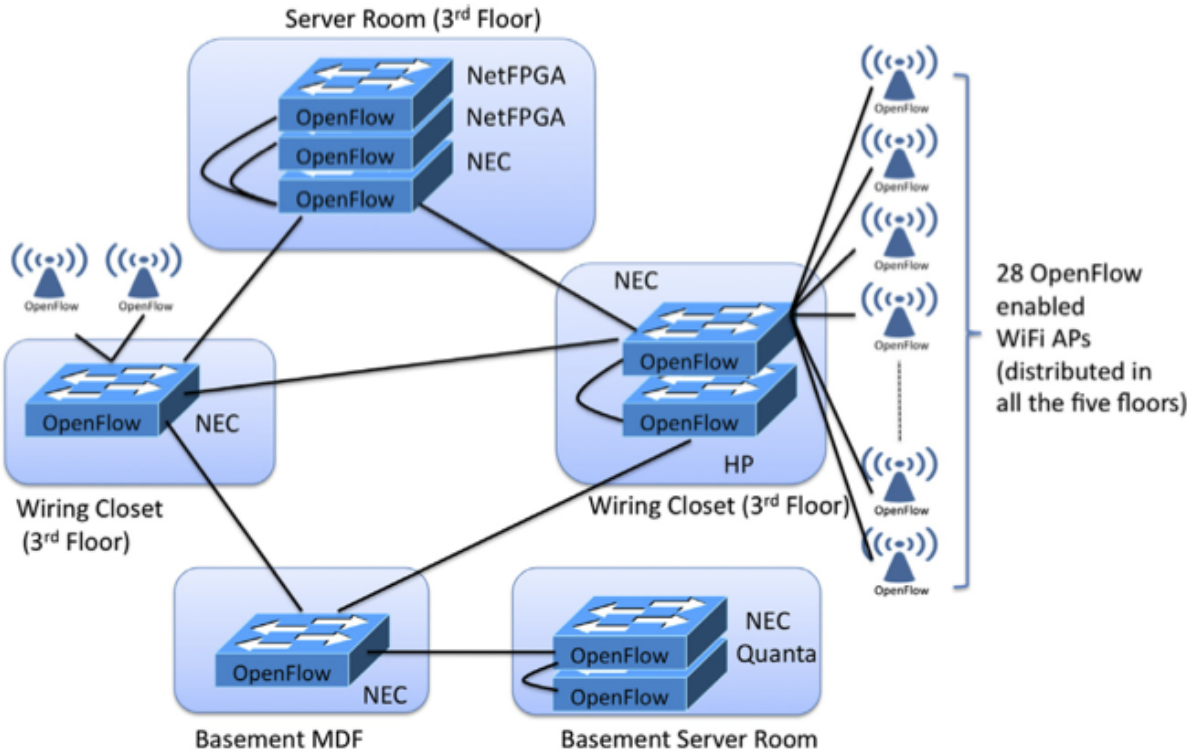


Figure 3.11: A logical view of the initial Gates building topology [191].

3.4.2 Stanford Use Cases

Through the many years of running experimental and production environments, Stanford produced a number of interesting studies. OpenRoads [333], [334] is built directly from *ofwifi* and allows researchers to slice wireless networks and, therefore, conduct experiments on a production wireless network. This innovation allows for creative use cases. For example, classes of students can develop different mobile management solutions and deploy them at the same time without interfering with one another [334]. OpenRoads provided a great trial for FlowVisor and as an incubator for the improvements that SDN and OpenFlow can bring to wireless networks.

The Stanford SDN did not serve only as an important resource to generate new ideas about how to use OpenFlow, but it also provided much needed feedback on many critical SDN projects. For instance, although FlowVisor was developed and designed at Stanford, there were many improvements made by analyzing how it was used in the production network [191]. Other contributions made by this network include recognizing compatibility issues with OpenFlow and Spanning Tree Protocol, finding limitations on flow table size, fixing bugs in NOX and SNAC, discovering problems with Wi-Fi handover, and offering a number of suggestions to improve the OpenFlow specification [191].

3.5 MCNC

MCNC is a non-profit organization in North Carolina that creates and manages network infrastructure for education, research, and other important institutions. In November 2013, MCNC started a SDN working group to develop and implement SDN infrastructure between Duke University, the University of North Carolina at Chapel Hill (UNC), North Carolina State University (NCSU), MCNC, and the Renaissance Computing Institute (RENCI) [319]. This project is ongoing and is slated for completion after the publication of this thesis. However, this SDN will provide valuable insight regarding making architectural decisions and the issues that arise in the planning and development phase of deployment. Furthermore, it should be noted that the other case studies presented above involve networks that were designed and implemented at least four years ago. Therefore, they do not account for improvements in various SDN technologies including OpenFlow, hardware switches, and controllers. This section provides a detailed look into the future MCNC software-defined network. It is important to note that some information is incomplete due to various non-disclosure agreements (NDAs).

3.5.1 MCNC Design

The network, shown in Figure 3.12, uses a star topology with the MCNC node as the hub. Each research institution will have a 10G link to MCNC. Additionally, the MCNC node will connect

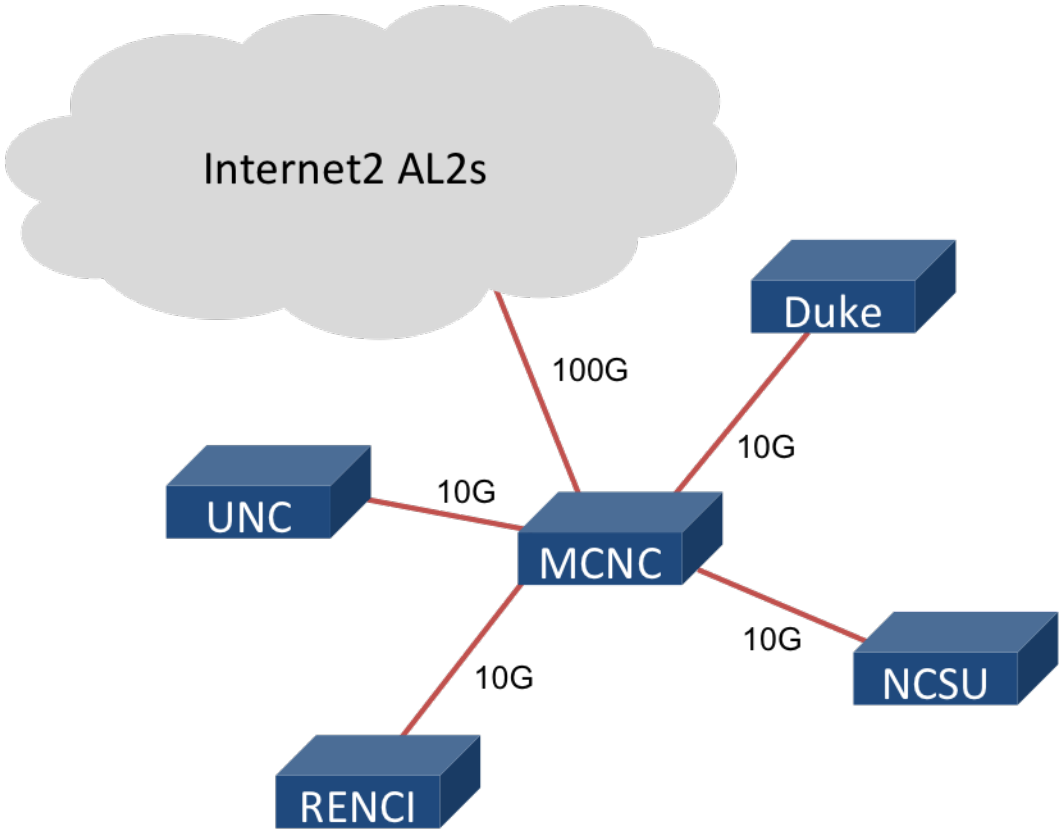


Figure 3.12: Future MCNC SDN topology.

to Internet2's Advanced Layer 2 Service (AL2S) via a 100G link. Each site is required to have a high-end OF compatible router and a VPN termination device. Currently, the designers are planning to use the VPN appliance to provide security for the network. However, ACLs will be used if the VPN appliance limits bandwidth significantly. In addition to the 10G experiment traffic, there will be a 1G management network, which connects the VPN appliance and site-specific router. Controller traffic will be sent over this management network to administer each router.

The MCNC network has the functionality to use both OpenFlow v1.0 and v1.3, depending on the use case. To provide slicing capabilities, designers are experimenting with FlowVisor and the FlowSpace Firewall. However, a significant limitation to both of these slicing mechanisms is the lack of OF v1.3 support. To manage this new network via OpenFlow, the designers are experimenting with several controllers including POX, Floodlight, OpenDaylight, and a commercial controller. However, this management controller will not impede researchers' ability to choose a unique controller for their experiment slices.

This project is currently behind schedule for several reasons. First, there were delays in receiving the routers from the hardware vendor. Also, the software update that allows OpenFlow compatibility was not delivered on schedule. Additional problems occurred when the equipment did not fit into the existing racks at several sites and required one site to undergo a power upgrade. The combination of these delays have caused the project to be delayed by several months.

3.5.2 Use Cases

The use cases presented below are examples of how researchers might use the new MCNC software-defined network. These experiments are in the early stages of development. However, they will demonstrate how the new SDN infrastructure can be tested leading to future opportunities for new research.

L2 VLAN Extension

This simple use case will provide campuses the ability to create VLAN circuits between two or more sites. This VLAN extension will be key in testing the new infrastructure. Additionally, it will be a foundation for more advanced experiments that will be developed in the future. This case is structured into two components: a point-to-point extension and a multi-point extension. The point-to-point extension will create a virtual circuit between two campuses through the MCNC infrastructure, whereas the multi-point extension will involve three or more campuses. During initial testing, a learning switch will be used to distribute traffic in the muti-point case. Other implementation details will be developed after the OF controller and slicing mechanisms are clarified.

SDN Tenant Networks

SDN can implement transport services in dynamic circuits and multi-point networks [16], [116]. These networks are known as tenant networks [16], [116]. To further define tenant networks, we can assume that they emulate Ethernet pseudowires, can cross multiple transport providers, and the tenant controls their layer 3 addressing and edge routing. This use case seeks to discover the full value of SDN controlling of a transport network. To do so, researchers will extend tenants' control over end hosts and edge networks by proposing new routing, filtering, and monitoring transport services to be used by the tenants. Though initially tenants will be able to install only static routes at specific exchanges, these services will eventually offer tenants full control over all layer 3 flows within their own tenant networks. Ultimately, this research will interconnect a virtual group's systems with a secure, private, bandwidth-provisioned IP network.

Chapter 4

Discussion

4.1 OpenFlow Switches

An abundance of OpenFlow compatible hardware exists, as depicted in Table A.1. However, most of these devices are not used within existing testbeds. This section compares and contrasts the hardware choices discussed in Chapter 3. Additionally, OF capable software switches, such as Open vSwitch or EDOBRA Switches, will be discussed briefly. This section focuses on the relevant (i.e., SDN capable) network equipment and disregards all other hardware including, servers, firewalls, and other middleboxes.

4.1.1 Hardware Switches

Table 4.1 presents the hardware that is used in each testbed described in Chapter 3. Within each case study, the table is subdivided by the location of the hardware due to the autonomy of the different OFELIA islands and the differences in GENI rack hardware. Additionally, the table displays the switch model, port bandwidth, supported OpenFlow version, any additional information, and a reference.

GENI used three vendors, IBM, HP, and Dell, to provide OpenFlow switches to each of its three rack types. All of these OF switches have at least 10G capability, which provides enough bandwidth to connect to the GENI backbone. The IBM and HP OF switches used in GENI are currently OpenFlow v1.3 capable, depending on the software implemented, while the Dell switch is expecting v1.3 support in early 2015 [70], [151], [163].

OFELIA is a federation of OpenFlow-capable islands that have autonomy in choosing their hardware. Some islands chose a single vendor for all of their OF switches, such as ETH, while others chose to combine switches from multiple vendors, including the TUB island. OFELIA used NEC, HP, NetFPGA, and Datacom for its OF hardware needs. Almost all islands use a 1G OpenFlow network. This is attributable to the widespread use of NetFPGA 1G switches

in many of the islands. Additionally, only a handful of the OF switches used in OFELIA are v1.3 compatible (with a software update), limiting the future capabilities of researchers in this testbed.

Stanford's influence on the development of the OpenFlow protocol explains the diverse set of switches in its networks that come from four vendors: NEC, NetFPGA, Pronto (now Pica8), and HP. Additionally, because this production network only covers a single building, a 1G switch provides adequate bandwidth. Stanford's involvement in the OpenFlow protocol specification allows them to ensure that the latest specification is applied to its switches, even if it is not officially supported by the vendor.

While the MCNC network hardware falls under a non-disclosure agreement, information can be provided about its future bandwidth and OpenFlow needs. The switches for the "spoke" sites will use 10G and 40G links while the main MCNC site also will include a 100G link for a high speed connection with Internet2. All of these switches will support OpenFlow v1.3 with an option to use v1.0, thereby providing considerable flexibility to researchers.

The vendors used in the case studies represent only a small fraction of currently available OpenFlow hardware. In total, seven vendors were used (excluding unknown MCNC hardware). Even more surprising, only nine unique switches were used and only HP provided more than one switch model. Additionally, four vendors, Pica8, Datacom, Dell, and IBM only supplied OF capable switches to one project. While this study does not account for the total number of OF switches deployed at each site, it is clear that NEC, HP, and NetFPGA were used in the most locations. Two main factors influenced these choices: compatibility and price.

Compatibility issues were a major concern for SDNs until recently, due to the fact that few companies produced OF switches. After developing the OpenFlow protocol, Stanford researchers worked with Cisco, HP, NEC, and Juniper to add OpenFlow support to their switches [191]. However, Juniper did not release a software update with OpenFlow support until 2014 [244]. Furthermore, Cisco has simultaneously embraced OpenFlow through its support of the ONF, the body that manages the OpenFlow standard, while, at the same time, delaying OpenFlow development in favor of its own SDN approach, which uses application-centric infrastructure (ACI) and onePK. Cisco does have OF v1.0 and v1.3 agents available for its Catalyst, ASR 9000, and Nexus platforms, but these agents are under limited release and include many restrictions that may impede OpenFlow usability [58], [165], [235], [267]. As opposed to Cisco and Juniper, NEC and HP have quickly embraced OpenFlow support within their product lines. While NEC's IP8800 is no longer available, NEC has three new OpenFlow v1.3 compatible switches (see Table A.1 in Appendix A) [300]. Additionally, HP offers an extensive line of OpenFlow v1.3 switches, indicating its continued support for the protocol. Many of the hardware vendors in Table 4.1 are working to improve their OpenFlow support to v1.3, which addresses concerns about the sustainability of the OpenFlow protocol.

Equipment cost is another important factor for researchers and campus information technology services, especially as many colleges have experienced budget cuts in recent years [313]. It is likely that this issue influences Stanford's use of less expensive switches in its production SDN. The proposed MCNC network is positioned to lead UNC, Duke, and NCSU to transition their campus networks to use SDN. Like Stanford and other campuses, these institutions will need to consider the cost of hardware when designing their networks due to the constraints of shrinking budgets. OpenFlow enables interoperability between switches from many vendors, opening the possibility for administrators to look beyond traditional switch vendors. For example, NetFPGA [222], an open source switching platform built for researchers, and Pica8, a white box switch vendor that leverages OVS [260], provide OpenFlow compatibility for a fraction of the cost of traditional networking hardware. These cost effective options likely will facilitate campus transitions to SDN in the future.

4.1.2 Software Switches

While hardware vendor support for OpenFlow was vital to the creation of the existing research networks, the advancements of software switches make them a competitive alternative to traditional switches. Software switches are an inexpensive way for researchers to use OpenFlow without purchasing expensive hardware. Furthermore, the ease of upgrading software switches provides administrators access to the latest SDN enhancements without waiting for official vendor support. Software switches not only play an important role in research labs, but they have become the software base of white box hardware switches [260]. Two software switches were used in the case study examples.

First, Open vSwitch [240] offers researchers and network designers a production quality virtual switch that is OpenFlow v1.4 compatible. OVS is used in the OFELIA and GENI testbeds. Several of OFELIA's islands use OVS, such as IBBT's Virtual Wall facility [306] that provides an Emulab environment in which OVS is used to support OpenFlow. Within GENI, the ExoGENI and InstaGENI racks use OVS to provide experimenters with OpenFlow capabilities without any hardware limitations [17], [112]. Additionally, within the OpenGENI rack, GRAM uses OpenStack's Quantum plugin for OVS [253] to manage inter-node network traffic [114], [250].

The second software switch mentioned within these case studies is the EDOBRA switch [234], used in the Brazil island of OFELIA. EDOBRA is a custom designed IEEE 802.21-enabled OpenFlow-capable switch that is based on OpenWRT [47]. These switches enable researchers to explore the interactions between OpenFlow and wireless technologies. For example, EDOBRA switches improve upon ICN techniques in wireless scenarios [117].

Currently, software switches are not a dominate force within the case study infrastructure

as they are only used in minor support roles and experimental cases. Nevertheless, the use of software switches will grow because they provide a low cost alternative to network hardware and provide a solution to the hardware limitations discussed in Section 4.3.1. Therefore, software switches will likely become a vital component of future SDN testbeds.

Table 4.1: Comparison of hardware used in case study examples.

Case Study	Equipment Location	OF Switch Model	Port Bandwidth	OF Version Supported	Additional Information	Reference
GENI	ExoGENI Racks	IBM RackSwitch G8264	Both 10G and 40G	v1.0 and v1.3	OpenFlow v1.3 support started with IBM Networking OS 7.8 [163].	[60], [84]
	InstaGENI Racks	ProCurve 6600 (now HP 6600)	1G with a few 10G ports	v1.0 and v1.3	OpenFlow v1.3 support started with HP's switch software K/KA/WB 15.14 [151].	[60], [207]
	OpenGENI Racks	Force10 S4810 (now Dell S-Series)	10G with a few 40G ports	v1.0	OpenFlow v1.3 support is expected in early 2015 [70].	[60], [250]
OFELIA	TUB	NEC IP8800/S3640	1G (a few 10G depending on model)	v1.0	Not currently offered on the open market [300].	[192], [307]
		NetFPGA 1G	1G	v1.0	–	[192], [307]
		HP 5400 zl	1G and 10G (depending on line cards)	v1.0 and v1.3	OpenFlow v1.3 support started with HP's switch software K/KA/WB 15.14 [151].	[192], [307]
	IBBT	NEC IP8800/S3640	1G (a few 10G depending on model)	v1.0	Not currently offered on the open market [300].	[232], [306]

Table 4.1: Continued.

Case Study	Equipment Location	OF Switch Model	Port Bandwidth	OF Version Supported	Additional Information	Reference
OFELIA	ETH	NEC IP8800/S3640	1G (a few 10G depending on model)	v1.0	Not currently offered on the open market [300].	[312]
	i2CAT	NEC IP8800/S3640	1G (a few 10G depending on model)	v1.0	Not currently offered on the open market [300].	[232], [305]
		HP 3500 yl	1G with a few 10G ports	v1.0 and v1.3	OpenFlow v1.3 support started with HP's switch software K/KA/WB 15.14 [151].	[232], [305]
	UNIVBRIS	NEC IP8800/S3640	1G (a few 10G depending on model)	v1.0	Not currently offered on the open market [300].	[232], [309]
	CNIT (Catania)	NetFPGA 1G	1G	v1.0	–	[212], [232], [310]
	CNIT (Rome)	NetFPGA 1G	1G	v1.0	–	[212], [232], [310]
	CREATE-NET	NEC IP8800/S3640	1G (a few 10G depending on model)	v1.0	Not currently offered on the open market [300].	[232], [311], [317]

Table 4.1: Continued.

Case Study	Equipment Location	OF Switch Model	Port Bandwidth	OF Version Supported	Additional Information	Reference
OFELIA	CREATE-NET	ProCurve 3500 (now HP 3500 yl)	1G with a few 10G ports	v1.0 and v1.3	OpenFlow v1.3 support started with HP's switch software K/KA/WB 15.14 [151].	[232], [311], [317]
		NetFPGA 1G	1G	v1.0	–	[232], [311], [317]
	CNIT (Pisa)	NetFPGA 1G	1G	v1.0	–	[232], [310]
	UFU	Datacom DM4000	1G with a few 10G ports	v1.0	–	[232], [308]
		NetFPGA 1G	1G	v1.0	–	[232], [308]
Stanford	Gates Building	NEC IP8800/S3640	1G (a few 10G depending on model)	v1.0	Not currently offered on the open market [300].	[191], [293]
		HP 5400 zl	1G and 10G (depending on line cards)	v1.0 and v1.3	OpenFlow v1.3 support started with HP's switch software K/KA/WB 15.14 [151].	[191], [293]
		NetFPGA 1G	1G	v1.0	–	[191], [293]

Table 4.1: Continued.

Case Study	Equipment Location	OF Switch Model	Port Bandwidth	OF Version Supported	Additional Information	Reference
Stanford	Gates Building	Pronto P-3290 (now Pica8)	1G with a few 10G ports	v1.0 and v1.4	OpenFlow v1.4 support depends on the software installed on the switch [260].	[191], [293]
MCNC	UNC	N/A	1G, 10G, 40G (depending on line cards)	v1.0 and v1.3	Manufacturer and model are subject to a NDA.	–
	NCSU	N/A	1G, 10G, 40G (depending on line cards)	v1.0 and v1.3	Manufacturer and model are subject to a NDA.	–
	Duke	N/A	1G, 10G, 40G (depending on line cards)	v1.0 and v1.3	Manufacturer and model are subject to a NDA.	–
	RENCI	N/A	1G, 10G, 40G (depending on line cards)	v1.0 and v1.3	Manufacturer and model are subject to a NDA.	–
	MCNC	N/A	1G, 10G, 40G, 100G (depending on line cards)	v1.0 and v1.3	Manufacturer and model are subject to a NDA.	–

4.2 Testbed Software

While hardware choices affect an administrator's budget, the management of the network affects his or her day-to-day responsibilities. OpenFlow and SDN theoretically simplify network management by using a centralized controller. Furthermore, many campus and research networks may choose to utilize network slicability more fully with a tool like FlowVisor. However, there are advantages and disadvantages to all available software. This section uses the case study examples to demonstrate limitations of existing software and present features to seek out when building a new software-defined network.

Table 4.2 presents a comparison of the software choices used in each of the case studies discussed in Section 3. The table specifies where the software is used, such as within a specific GENI rack type. Next, it lists any OpenFlow controllers that are used by the environment or by experiments running on top of the environment. In the next column, any other SDN specific software used within the environment is listed. Finally, the table outlines any additional information about the software and provides a reference.

The GENI racks allow experimenters to define which controller they would like to use within their slice. A sample of the controllers that experimenters have used within GENI include Beacon [81], Floodlight [100], Maestro [43], NodeFlow [25], NOX [228], POX [264], and Trema [316]. Within OpenGENI racks, POX and VMOC are leveraged as part of its rack aggregate manager [114]. In contrast to OpenGENI's use of GRAM as its aggregate management software, InstaGENI and ExoGENI use FOAM. The final major software difference between these racks is ExoGENI's use of ORCA to assist in controlling rack hardware.

OFELIA experimenters also can choose any OpenFlow controller, although both NOX and SNAC [290] were specifically mentioned as being used by an island. OFELIA also uses slicing mechanisms, including FlowVisor, Optical FlowVisor, and VeRTIGO, and the OFELIA Control Framework for testbed management.

In the Stanford production network, the SNAC controller initially was used before the network transitioned to the Big Network Controller [28], a commercial controller by Big Switch Networks. Additionally, Stanford utilizes FlowVisor for network slicing.

The MCNC network design plan includes a split environment with an experimenter segment and a control segment. Researchers will be able to define their own controllers within their network slice and both POX and Floodlight are being investigated for this purpose. Within the control segment, administrators will use either OpenDaylight [241] or a commercial controller to manage the network. Additionally, MCNC is investigating the possibility of utilizing either FlowVisor or the FlowSpace Firewall for slicing capabilities.

Table 4.2: Comparison of software used in case study examples.

Case Study	Site Specific (if any)	Controller	Other Software	Additional Information	Reference
GENI	ExoGENI Racks	Experimenter defined	ORCA, FOAM	–	[60], [84]
	InstaGENI Racks	Experimenter defined	FOAM	–	[60], [207]
	OpenGENI Racks	VMOC, POX, experimenter defined	GRAM	–	[60], [250]
	All	Beacon, Floodlight, Maestro, NodeFlow, NOX, POX, Trema	–	These are examples of OpenFlow controllers that researchers have used within GENI.	[243]
OFELIA	N/A	NOX, SNAC, experimenter defined	FlowVisor, VeRTIGO, OFV, OCF	Experimenters can use any OF-enabled controller for their slice.	[192], [301], [306], [309], [311]
Stanford	N/A	SNAC, Big Network Controller	FlowVisor	–	[191]
MCNC	Main Environment	OpenDaylight or a commercial controller	FlowSpace Firewall, FlowVisor	The environment will be set up so that a main controller will manage the network.	–
	Within an experimental slice	Floodlight, POX, experimenter defined	–	These are the controllers that are being evaluated.	–

4.2.1 Controller Type

In three testbeds, GENI, OFELIA, and MCNC, experimenters can define their own OpenFlow controller. POX, SNAC, and Floodlight have each been used in two case studies. However, SNAC was replaced by a commercial controller in the Stanford network and POX only supports OpenFlow v1.0. Additionally, MCNC is seeking a commercial controller for the management environment of the new SDN. The interest in a commercial controller by Stanford and MCNC indicates that a production quality network should be handled with a commercial quality controller. The initially developed OpenFlow controllers (i.e., NOX, POX, and SNAC) will soon be phased out of large deployments in favor of commercial controllers from Big Switch, Cisco, and others. However, smaller SDN deployments and experimenters will still use open source controllers that have an ongoing development community, such as Floodlight and OpenDaylight. It is also important to note that both of these controllers support OpenFlow v1.3, thereby increasing their appeal.

4.2.2 Slicing Tools

Slicing utilities comprise the majority of the other SDN-related software used in these testbeds. For instance, FlowVisor is used or being investigated for use in all the case studies presented. However, its monopoly on slicing OpenFlow networks creates limitations. First and perhaps most importantly, FlowVisor does not support OpenFlow v1.3. This is a significant problem for network designers that need to use important features like Internet Protocol version 6 (IPv6), MPLS, Q-in-Q tunneling, and extensible header support. Additionally, v1.3 addresses many of the scalability concerns of OpenFlow v1.0. Furthermore, lack of v1.3 support prohibits researchers from conducting experiments using the latest version of the protocol. This limitation alone may cause future designers to move away from FlowVisor. Other limitations of FlowVisor include a single point of failure for the entire test bed and increased packet latency. A failure of FlowVisor could ruin all ongoing experiments, depending on its implementation. While using OpenFlow adds some packet latency already, researchers may not want additional overhead on top of this existing delay. Finally, FlowVisor limits a researcher's ability to use arbitrarily large topologies [288]. That is, a single physical switch cannot be used more than once in a given slice's topology, disallowing a virtual topology that is larger than the underlying physical network [288].

MCNC, GENI, and OFELIA are currently looking for a FlowVisor alternative, indicating that it will not remain a prominent part of future networks without adaptation. At the 21st GENI Engineering Conference (GEC21), the GENI Operations group discussed the removal and replacement of FlowVisor as well as requirements for an alternative [106], [111]. Furthermore, it should be noted that OpenGENI operates without the use of FOAM and instead relies on

GRAM, which does not contain a FlowVisor component. OFELIA administrators plan to replace FlowVisor mainly due to researchers wanting the testbed to use OpenFlow v1.3 [276]. These researchers have developed another NV framework that uses eXtensible DataPath Daemon (xDPd), a new OpenFlow datapath building framework, to create a multi-platform datapath based on a scalable and robust NV architecture [76]. This framework allows researchers to use multiple versions of OpenFlow on many hardware platforms with minimal overhead, overcoming a key weakness of FlowVisor [76].

It is beyond the scope of this thesis to evaluate the many FlowVisor alternatives outlined above and in Section 2.1.2; therefore, we will leave this task to later work. However, it is clear that newly designed SDNs should be aware of the flaws in existing slicing solutions like FlowVisor and administrators should look for new methods of providing either full network virtualization or simplistic OpenFlow v1.3 network slicing.

4.3 Notable Issues

In this section, we examine issues that occurred within the case study examples presented in Chapter 3. These issues relate specifically to the design, implementation, or operation of software-defined networks and their associated technological components. While some issues listed below have been resolved, there are still open questions that will impact the design and implementation of a new SDN. Although each testbed experienced many nuanced individual site issues, the challenges described in this section are overarching problems that will affect all future SDNs.

4.3.1 Hardware Limitations

Section 4.1.1 briefly describes the limited hardware support for OpenFlow v1.3. However, there are other hardware limitations that create issues for the presented case studies and may remain a persistent challenge to future software-defined networks. The most prevalent hardware issues include flow table limitations, software bugs, incomplete or inaccurate OpenFlow support, low performance CPUs, and incompatibilities with hybrid OF networks.

The first of these challenges is the limited number of flow entries allowed on many switches. Designers of the OFELIA testbed and Stanford network had issues with this limit due to a shortcoming of the OpenFlow v1.0 specification, in which a single table must be able to match any of the twelve required fields [245], [301]. This encouraged switch manufactures to implement the flow tables within ternary content addressable memory (TCAM), which allowed the flow table to only contain a few thousand entries [202]. For example, Stanford noticed that their flow table was limited to approximately 1,500 entries [191]. This low number of entries dramatically

reduces the usability of OpenFlow in a production environment. While small flow tables are due, in part, to the switch/hardware limitations, this issue is largely a product of the initial OpenFlow specification. Many vendors have improved flow table size, but future administrators still need to ensure the maximum flow table size will fit their needs. Administrators that experience flow table limitations should use wildcard matching to assist in aggregate flows, thus reducing the total number of flow rules [191].

Another major issue involves interoperability bugs between the OF capable switch and its controller. For instance, the Stanford team identified a number of bugs within the NOX and SNAC controllers that were discovered after they were combined with an HP switch [191]. In another case, the researchers found the following race condition: when a packet arrived at the switch before the switch finishes processing a new flow rule for the arriving packet, it was dropped [191]. Like all software systems, network controllers are prone to bugs. However, tools, such as the SDN Troubleshooting System [278], may allow programmers to fix previously unknown controller bugs. Unfortunately, this system does not resolve issues within switch software. Therefore, while the problems mentioned above have been resolved, network designers should be aware that many switch vendors only recently have implemented OpenFlow and may not have discovered all reliability issues. Furthermore, while switches and controllers that implement the OpenFlow specification fully should theoretically communicate without issues, interoperability between switches and controllers remains a concern for administrators.

The concerns raised above are amplified by hardware vendor exceptions to full OF support. That is, although many vendors claim to fully support OpenFlow v1.X, many caveats exist within this support. For instance, Dell does not support emergency flows, TLS connections, or packet buffering [72]. Additionally, within Dell switches, VLAN IDs cannot be used for OpenFlow and legacy ports simultaneously and the flow priority for layer 2 flows is ignored [72]. Juniper OF switches have only one flow table, do not support the required `OFPP_IN_PORT` or `OFPP_TABLE` forwarding actions, do not support TLS connections between the switch and controller, and only allow for one active OF controller [183]. Furthermore, even though OF v1.3 adds support for provider backbone bridging (PBB), IPv6, and MPLS, none of these features are implemented in Juniper's switch software [183]. HP, one of the first vendors to support OpenFlow, still limits its support. These limitations include issues supporting the `OFPP_TABLE` forwarding action, some MPLS and PBB actions, and several port modification commands [151]. Even vendors that support only OpenFlow v1.0 include a long list of limitations in their user documents. For example, Extreme Networks fails to support the `IN_PORT`, `FLOOD`, `NORMAL`, and `TOS/DSCP` editing actions [97]. Dell, Juniper, HP, and Extreme Networks are a small, but representative sample of vendors that either fail to fully comply with the specification or implement only the minimum set of OpenFlow requirements. As these examples demonstrate, even when a switch is OF v1.3 compatible, it may not provide all the new features that the OpenFlow

specification outlines. Therefore, it is vital that future administrators consider these limitations when deciding on a hardware vendor for their network.

Though vendors are rapidly developing OpenFlow switch software, most hardware is not optimized for OpenFlow traffic. In particular, most switches have low-performance control plane CPUs. This makes sense in legacy network environments, but creates problems when these CPUs have to process large numbers of OpenFlow messages. Inadequate CPUs cause lengthy flow setups, poor response times to controller commands, and even network failures [191]. This behavior is especially prominent when a controller acts in reactive mode [191]. The clear solution is for switch vendors to increase CPU power and begin optimizing switch performance for OpenFlow and other SDN technologies. However, short-term solutions for administrators include placing the controller in proactive mode, rate-limiting OpenFlow control messages, and aggregating flow rules [191].

Switch Hybrid Mode

Hybrid switch mode, which uses both OpenFlow functions and legacy switch functions simultaneously, also causes several hardware limitations. The ONF Hybrid Working Group was created to address the challenges involved with *hybrid switches* [252]. The ONF found that there are two main forwarding models for hybrid OF switches: 1) Integrated and 2) Ships in the Night [252]. The Integrated approach involves incorporating OpenFlow into a legacy architecture, creating a single, unified environment [252]. In contrast, the Ships in the Night model defines an environment in which OpenFlow and legacy traffic are segregated [252]. The ONF suggests that vendors use the Ships in the Night model because it accurately addresses resource limitations as long as there is proper isolation between OF and non-OF functions [252]. Most vendors (Dell, Juniper, Extreme Networks, HP, Arista, Brocade, etc.) follow the Ships in the Night model by using OF-only VLANs to keep traffic properly isolated [12], [72], [97], [99], [149], [183]. Some vendors, including Arista and Brocade, also enforce isolation by implementing OpenFlow-only interfaces, in which all incoming traffic to that physical port is processed by the OpenFlow software stack [12], [99]. Though vendors use similar strategies, limitations of a particular hybrid implementation vary by vendor or even on a per-switch basis. For example, Juniper allows hybrid interfaces on its MX series of edge routers and EX9200 switches, but not on its EX4550 or QFX5100 switches [244].

The most common problem with hybrid switches involves the Spanning Tree Protocol. The Stanford production environment detected inconsistency issues when STP was enabled on OpenFlow VLANs [191]. The incompatibilities arise because the OF software stack has no knowledge that a port has been shut down by STP and still maintains that the port is in use [191]. Aside from Stanford, many switch vendors, including Dell, Juniper, Arista, and

Brocade, have specifically noted STP problems with hybrid ports [12], [72], [99], [183]. The recommended course of action is to disable STP for an OpenFlow deployment or, at minimum, on any ports that use both OpenFlow and legacy VLANs. Additionally, it is equally important that, when STP is disabled on the switch, the OF controller prevents loops within the network by monitoring flow entries.

Testbed control software should also be designed to work with hybrid switches. For example, ExoGENI's ORCA software recently introduced support that allows researchers to use the actual OF switch for an experiment rather than relying on OVS [17], [83]. However, even with this support, there are known issues with passing packets between the OpenFlow and VLAN sections of the switch [83]. This issue is due, in large part, to the segregation of OF and non-OF components in the Ships in the Night forwarding model described above.

4.3.2 Case Study Challenges

Many of the challenges faced by the case study examples have been discussed in previous sections. This section will address any additional issues that may be relevant for future SDN administrators.

Perhaps the most important problem with the current GENI environment is the lack of support for multiple controllers across multiple domains [207]. That is, currently, only a single OpenFlow controller manages all OF switches within an environment. This issue is specifically relevant to the InstaGENI rack design but applies to many other existing OF testbed implementations [207]. Lack of multi-controller support causes issues with scaling an OpenFlow network and introduces a single point of failure into the network. Furthermore, in multi-domain networks, the ability to view the entire topology is lacking. This issue affects research networks in multiple ways. First, it prevents researchers from experimenting with distributed OF controllers, such as HyperFlow [315] or Onix [193]. Secondly, it prevents multiple controllers from being connected to the same OpenFlow switch. This is due, in part, to the architecture design as well as the OF v1.0 specification. OpenFlow v1.2 and above [246] allow a switch to be multi-homed to multiple controllers; however, because GENI is currently running OF v1.0, this remains a major limitation.

Choosing an isolation mechanism was a major design challenge for OFELIA. This testbed uses VLAN ID-based slicing, due to the limited TCAM space described in Section 4.3.1. The OFELIA designers did not choose VLAN ID-based slicing only due to the scalability concerns with OpenFlow v1.0 but also because it works well with legacy switches and ensures layer 2 traffic isolation [301]. That is, VLAN-based slicing functions better in a hybrid-SDN environment. This argument is strengthened by the decision of many switch vendors to segregate OpenFlow and legacy traffic through the use of VLANs, as discussed in Section 4.3.1. While VLAN ID-

based slicing alleviates the concerns outlined above, some issues remain. First, there can be a maximum of only 4096 VLANs. In reality, the number of VLANs is less than the maximum capacity, especially within a hybrid SDN environment. Secondly, in the case of a federated testbed like OFELIA, inter-island connectivity becomes challenging and requires coordination between the islands as well as tunneling capabilities. In OFELIA, this issue was resolved through the use of generic routing encapsulation (GRE) tunnels, rather than Q-in-Q tunneling [194].

Stanford's deployment team noticed long delays in flow setup when the controller was in reactive mode. The root cause of this delay was Nagel's algorithm. This algorithm solves the small-packet problem, which occurs when packets contain a minimal amount of data (typically one byte), thereby greatly improving the efficiency of TCP [215]. In general, Nagel's algorithm improves TCP, but it has been known to cause performance issues in some applications [214]. The Stanford team noticed that Nagel's algorithm increased flow setup time during message exchange between the controller and switch [191]. OpenFlow control packets are small (e.g. only a few bytes) but are time sensitive and, therefore, should not wait to satisfy the conditions of Nagel's algorithm before transmission. Thus, it is recommended that network administrators manually disable the algorithm for control traffic if the controller does not do so by default [23], [191].

Chapter 5

Conclusion and Future Work

5.1 Summary

The goal of this thesis was to identify commonalities among issues occurring in four software-defined networks. We summarized the infrastructure hardware and the software systems responsible for controlling SDN components of these testbeds. This analysis will shape future SDN design decisions by steering administrators away from potential problems and towards extensible solutions.

First, we identified four research-based software-defined networks determined by the following factors: longevity of testbed deployment, amount of publicly available documentation, diversity of SDN implementation, size of the network, and influence on SDN research. Using these principles, we selected GENI, OFELIA, Stanford's SDN, and MCNC's future SDN. These diverse environments represent a cross-section of testbeds, providing valuable insight into the wide range of complications that can occur when designing and implementing SDN testbeds.

By examining the hardware equipment used in the case studies, we discovered that only a small portion of the available switching hardware was utilized in the networks we examined. Both price and compatibility played a large role in vendor choice. Moreover, software-based switches contributed to the available OpenFlow compatible equipment, further reducing the cost burden on the testbeds. However, more vendors are supporting OpenFlow version 1.3, providing increasing competition for future testbeds. Nevertheless, hardware vendors still have a multitude of issues with SDN support. Current hardware switches have flow table limitations, software bugs, low performance CPUs, and incompatibilities with hybrid OF networks. However, the most concerning issue for network administrators is the incompleteness or inaccuracy of OpenFlow implementations provided by some vendors.

In terms of control software for future testbeds, we discovered numerous issues with current slicing platforms. Most prominently, FlowVisor, the leading slicing solution, will be inadequate

to meet the needs of future SDN testbeds. Its limitations will force testbed designers to find or develop alternative solutions to meet their needs. Additionally, while most testbeds allow for experimenter defined controllers, there has been a shift away from primitive research controllers towards well-developed alternatives, such as OpenDaylight and other commercial products.

Finally, we offered other lessons learned to help architects avoid the common pitfalls of implementing SDN within a testbed. Our discussion ranged from disabling the Spanning Tree Protocol to methods of connecting islands in a federated environment. Through this thesis, readers gained an intimate understanding of how to build a successful and future-proof testbed that will support numerous SDN research projects.

5.2 Future Work

This thesis is an important step toward identifying weaknesses in existing SDN implementations. Additionally, we presented numerous pitfalls that future designers should avoid while building a new SDN testbed. This research should be extended in two directions. The first involves finding new solutions to address issues raised within this study. In particular, hardware vendors can work with the SDN community to resolve the shortcomings of their hardware implementations of OpenFlow. Secondly, available solutions that have similar goals can be evaluated to provide designers a clear implementation direction and details about various trade-offs. These two research directions will further distinguish the necessary steps for designing and implementing a software-defined research network.

REFERENCES

[1] About GENI, URL: http://www.geni.net/?page_id=2 (visited on 02/26/2015).

[2] ADVA optical FSP 3000 ROADM equipment, ADVA Optical Networking, URL: http://www.advaoptical.com/en/products/scalable-optical-transport/fsp-3000.aspx (visited on 12/08/2014).

[3] (May 10, 2012). ADVA optical networking pioneers OpenFlow in the optical domain, ADVA Optical Networking, URL: http://www.advaoptical.com/en/newsroom/press-releases-english/20120510 (visited on 01/15/2015).

[4] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *2013 Proceedings IEEE INFOCOM*, 2013, pp. 2211–2219.

[5] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, W. Snow, and G. Parulkar, "Openvirtex: A network hypervisor," in *Open Networking Summit*, Santa Clara, CA: USENIX, 2014.

[6] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, ser. SOSP '01, New York, NY, USA: ACM, 2001, pp. 131–145, ISBN: 1-58113-389-8.

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

[8] V. Antonenko and R. Smelyanskiy, "Global network modelling based on mininet approach.," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13, New York, NY, USA: ACM, 2013, pp. 145–146, ISBN: 978-1-4503-2178-5.

[9] V. Antonenko, R. Smelyanskiy, and A. Nikolaev, "Large scale network simulation based on hi-fi approach," in *Proceedings of the 2014 Summer Simulation Multiconference*, ser. SummerSim '14, San Diego, CA, USA: Society for Computer Simulation International, 2014, 4:1–4:8.

[10] T. Araujo and R. M. Salles, "Giroflow: OpenFlow virtualized infrastructure management tool," in *Proc. of the 10th International Conference on Network and Service Management*, vol. 1, Rio de Janeiro, Brazil, Nov. 2014, pp. 1–5.

[11] Arista - software driven cloud networking, URL: http://www.arista.com/en/products/software-driven-cloud-networking (visited on 01/15/2015).

[12] *Arista user manual - arista EOS version 4.14.6m*, Jan. 19, 2015. URL: http://www.arista.com/docs/Manuals/ConfigGuide.pdf (visited on 01/26/2015).

[13] 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," in *2011 37th European Conference and Exhibition on Optical Communication (ECOC)*, Sep. 2011, pp. 1–3.

[14] S. Azodolmolky, R. Nejabati, S. Peng, A. Hammad, M. Channegowda, N. Efstathiou, A. Autenrieth, P. Kaczmarek, and D. Simeonidou, "Optical FlowVisor: An OpenFlow-based optical network virtualization approach," in *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2012 and the National Fiber Optic Engineers Conference*, Mar. 2012, pp. 1–3.

[15] B1sdn OpenFlow switch series, URL: `http://www.znyx.com/products/hardware/b1sdn.jsp` (visited on 02/11/2015).

[16] K. Bakshi, "Considerations for software defined networking (SDN): Approaches and use cases," in *2013 IEEE Aerospace Conference*, Mar. 2013, pp. 1–9.

[17] Baldin. (Sep. 18, 2014). Running OpenFlow experiments with real rack switches, URL: `http://www.exogeni.net/2014/09/running-openflow-experiments-with-real-rack-switches/` (visited on 01/18/2015).

[18] I. Baldine, Y. Xin, A. Mandal, P. Ruth, C. Heerman, and J. Chase, "Exogeni: A multi-domain infrastructure-as-a-service testbed," in *Testbeds and Research Infrastructure. Development of Networks and Communities*, T. Korakis, M. Zink, and M. Ott, Eds., ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 44. Springer Berlin Heidelberg, Jan. 1, 2012, pp. 97–113, ISBN: 978-3-642-35575-2, 978-3-642-35576-9.

[19] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic controller provisioning in software defined networks," in *2013 9th International Conference on Network and Service Management (CNSM)*, IEEE, Oct. 2013, pp. 18–25.

[20] N. Bastin, A. Bavier, J. Blaine, J. Chen, N. Krishnan, J. Mambretti, R. McGeer, R. Ricci, and N. Watts, "The InstaGENI initiative: An architecture for distributed systems and advanced programmable networks," *Computer Networks*, Special issue on Future Internet Testbeds Part I, vol. 61, pp. 24–38, Mar. 14, 2014, ISSN: 1389-1286.

[21] A. Bates, K. Butler, A. Haeberlen, M. Sherr, and W. Zhou, "Let SDN be your eyes: Secure forensics in data center networks," in *Proceedings of the NDSS Workshop on Security of Emerging Network Technologies (SENT14)*, 2014.

[22] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI veritas: Realistic and controlled network experimentation," in *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '06, New York, NY, USA: ACM, 2006, pp. 3–14, ISBN: 1-59593-308-5.

[23] Beacon configuration, URL: https://openflow.stanford.edu/display/Beacon/Configuration (visited on 01/29/2015).

[24] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13, New York, NY, USA: ACM, 2013, pp. 151–152, ISBN: 978-1-4503-2178-5.

[25] G. Berger. (Jan. 8, 2012). Nodeflow: An OpenFlow controller node style, URL: http://garyberger.net/?p=537 (visited on 01/08/2015).

[26] 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," *Comput. Netw.*, vol. 61, pp. 5–23, Mar. 2014, ISSN: 1389-1286.

[27] S. Bhatia, M. Motiwala, W. Muhlbauer, Y. Mundada, V. Valancius, A. Bavier, N. Feamster, L. Peterson, and J. Rexford, "Trellis: A platform for building flexible, fast virtual networks on commodity hardware," in *Proceedings of the 2008 ACM CoNEXT Conference*, ser. CoNEXT '08, New York, NY, USA: ACM, 2008, 72:1–72:6, ISBN: 978-1-60558-210-8.

[28] Big network controller, Big Switch Networks, Inc. URL: http://bigswitch.com/products/SDN-Controller (visited on 12/11/2014).

[29] *BlackDiamond x8 data sheet*. URL: http://learn.extremenetworks.com/rs/extreme/images/BlackDiamond-X8-DS.pdf (visited on 01/15/2015).

[30] Z. Bozakov and P. Papadimitriou, "Towards a scalable software-defined network virtualization platform," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–8.

[31] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *2010 IEEE 35th Conference on Local Computer Networks (LCN)*, Oct. 2010, pp. 408–415.

[32] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and others, "Advances in network simulation," *Computer*, vol. 33, no. 5, pp. 59–67, 2000.

[33] *Brocade ICX 6450 data sheet*. URL: http://www.brocade.com/downloads/documents/data_sheets/product_data_sheets/brocade-icx-6430-6450-ds.pdf (visited on 01/26/2015).

[34] Brocade ICX 6450 switch, URL: http://www.brocade.com/products/all/switches/product-details/icx-6430-and-6450-switches/index.page (visited on 01/26/2015).

[35] *Brocade ICX 6610 data sheet*. URL: http://www.brocade.com/downloads/documents/data_sheets/product_data_sheets/brocade-icx-6610-ds.pdf (visited on 01/26/2015).

[36] Brocade ICX 6610 switch, URL: http://www.brocade.com/products/all/switches/product-details/icx-6610-switch/index.page (visited on 01/26/2015).

[37] Brocade MLX series, URL: http://www.brocade.com/products/all/routers/product-details/netiron-mlx-series/index.page (visited on 01/26/2015).

[38] Brocade NetIron CER 2000 series, URL: http://www.brocade.com/products/all/routers/product-details/netiron-cer-2000-series/index.page (visited on 01/26/2015).

[39] Brocade NetIron CES 2000 series, URL: http://www.brocade.com/products/all/switches/product-details/netiron-ces-2000-series/index.page (visited on 01/26/2015).

[40] Brocade software-defined networking (SDN) products, URL: http://www.brocade.com/solutions-technology/technology/software-defined-networking/products.page (visited on 01/26/2015).

[41] Brocade vyatta 5600 vRouter, URL: http://www.brocade.com/products/all/network-functions-virtualization/product-details/5600-vrouter/index.page (visited on 02/12/2015).

[42] S. Bryant, P Pate, and others, "Pseudo wire emulation edge-to-edge (PWE3) architecture," IETF, RFC 3985, Mar. 2005.

[43] Z. Cai, "Maestro: Achieving scalability and coordination in centralized network control plane," PhD thesis, Rice University, 2011.

[44] P. Calyam, S. Rajagopalan, A. Selvadhurai, S. Mohan, A. Venkataraman, A. Berryman, and R. Ramnath, "Leveraging OpenFlow for resource placement of virtual desktop cloud applications," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, May 2013, pp. 311–319.

[45] Capsulator, OpenFlow Wiki, URL: http://archive.openflow.org/wk/index.php/Capsulator (visited on 12/10/2014).

[46] M. Carbone and L. Rizzo, "Dummynet revisited," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 2, pp. 12–20, Apr. 2010, ISSN: 0146-4833.

[47] J. Castillo, F. Silva, A. Neto, F. Silva, P. Frosi, C. Guimaraes, D. Corujo, and R. Aguiar, "Evolving future internet clean-slate entity title architecture with quality-oriented control plane extensions," in *AICT 2014, The Tenth Advanced International Conference on Telecommunications*, 2014, pp. 161–167.

[48] Centec networks v330, URL: http://www.centecnetworks.com/en/SolutionList.asp?ID=42 (visited on 01/16/2015).

[49] Centec networks v350, URL: http://www.centecnetworks.com/en/SolutionList.asp?ID=43 (visited on 01/16/2015).

[50] M. Channegowda, P. Kostecki, N. Efstathiou, S. Azodolmolky, R. Nejabati, P. Kaczmarek, A. Autenrieth, J. Elbers, and D. Simeonidou, "Experimental evaluation of extended OpenFlow deployment for high-performance optical networks," in *2012 38th European Conference and Exhibition on Optical Communications (ECOC)*, Sep. 2012, pp. 1–3.

[51] M. Channegowda, R. Nejabati, M. Rashidi Fard, S. Peng, N. Amaya, G. Zervas, D. Simeonidou, R. Vilalta, R. Casellas, R. Martnez, R. Muoz, L. Liu, T. Tsuritani, I. Morita, A. Autenrieth, J. Elbers, P. Kostecki, and P. Kaczmarek, "Experimental demonstration of an OpenFlow based software-defined optical network employing packet, fixed and flexible DWDM grid technologies on an international multi-domain testbed," *Optics Express*, vol. 21, no. 5, pp. 5487–5498, Mar. 11, 2013.

[52] R. Chertov, S. Fahmy, and N. B. Shroff, "Fidelity of network simulation and emulation: A case study of TCP-targeted denial of service attacks," *ACM Trans. Model. Comput. Simul.*, vol. 19, no. 1, 4:1–4:29, Jan. 2009, ISSN: 1049-3301.

[53] N. M. M. K. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," *IEEE Communications Magazine*, vol. 47, no. 7, pp. 20–26, Jul. 2009, ISSN: 0163-6804.

[54] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, Apr. 8, 2010, ISSN: 1389-1286.

[55] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: An overlay testbed for broad-coverage services," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, Jul. 2003, ISSN: 0146-4833.

[56] *Cisco application centric infrastructure*, 2013. URL: http://www.cisco.com/c/dam/en/us/products/collateral/cloud-systems-management/aci-fabric-controller/at-a-glance-c45-729864.pdf (visited on 02/03/2015).

[57] Cisco nexus 1000v switch, Cisco, URL: http://cisco.com/c/en/us/products/switches/nexus-1000v-switch-vmware-vsphere/index.html (visited on 02/12/2015).

[58] Cisco plug-in for OpenFlow, Cisco, URL: http://cisco.com/c/en/us/td/docs/switches/datacenter/sdn/configuration/openflow-agent-nxos/cg-nxos-openflow.html (visited on 12/19/2014).

[59] Cotn: The california OpenFlow testbed network, URL: http://www.cenic.org/page_id=143/ (visited on 02/13/2015).

[60] Current GENI rack projects, URL: http://groups.geni.net/geni/wiki/GENIRacksHome (visited on 11/04/2014).

[61] Cyan blue planet SDN platform, URL: http://www.cyaninc.com/products/blue-planet-sdn-platform (visited on 01/15/2015).

[62] Cyan z-series packet-optical transport, URL: http://www.cyaninc.com/products/z-series-packet-optical (visited on 01/15/2015).

[63] Datacom DM4001, DATACOM, URL: http://www2.datacom.ind.br/new/?q=pt-br/node/205 (visited on 12/17/2014).

[64] Dell networking MXL blade switch, Dell, URL: http://www.dell.com/us/business/p/force10-mxl-blade/pd (visited on 01/22/2015).

[65] Dell networking n2000 series, Dell, URL: http://www.dell.com/us/business/p/networking-n2000-series/fs (visited on 01/22/2015).

[66] Dell networking n3000 series, Dell, URL: http://www.dell.com/us/business/p/networking-n3000-series/fs (visited on 01/22/2015).

[67] Dell networking n4000 series, Dell, URL: http://www.dell.com/us/business/p/networking-n4000-series/fs (visited on 01/22/2015).

[68] *Dell networking n4000 series data sheet*. URL: http://partnerdirect.dell.com/sites/channel/Documents/Dell-Networking-N4000-Series-Spec-Sheet.pdf (visited on 01/21/2015).

[69] Dell networking s-series, Dell, URL: http://www.dell.com/us/business/p/force10-s-series/pd (visited on 01/22/2015).

[70] Dell networking s4810, Dell, URL: http://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Dell_Force10_S4810_Spec_sheet.pdf (visited on 12/17/2014).

[71] Dell networking z-series, Dell, URL: http://www.dell.com/us/business/p/force10-z-series/pd (visited on 01/22/2015).

[72] *Dell OpenFlow deployment and user guide dell software-defined networking (SDN)*, Apr. 9, 2014. URL: https://www.force10networks.com/CSPortal20/KnowledgeBase/DOCUMENTATION/Deployment/SDN\%20Deployment\%20Guide\%20v2_Apr_09_2014.pdf (visited on 01/21/2015).

[73] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," DTIC Document, 2004.

[74] J. Dix. (Jan. 8, 2015). Inside AT&ts grand plans for SDN, Network World, URL: http://www.networkworld.com/article/2866439/sdn/inside-atts-grand-plans-for-sdn.html (visited on 02/03/2015).

[75] R. Doriguzzi Corin, M. Gerola, R. Riggio, F. De Pellegrini, and E. Salvadori, "Vertigo: Network virtualization and beyond," in *2012 European Workshop on Software Defined Networking (EWSDN)*, Oct. 2012, pp. 24–29.

[76] R. Doriguzzi-Corin, E. Salvadori, M. Gerola, M. Su, and H. Woesner, "A datapath-centric virtualization mechanism for OpenFlow networks," in *2014 Third European Workshop on Software Defined Networks (EWSDN)*, Sep. 2014, pp. 19–24.

[77] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *IEEE Internet Computing*, vol. 17, no. 2, pp. 20–27, Mar. 2013, ISSN: 1089-7801.

[78] J. Duffy. (Sep. 11, 2014). SDN market could hit $18b by 2018, Network World, URL: http://www.networkworld.com/article/2607095/sdn/sdn-market-could-hit-18b-by-2018.html (visited on 02/03/2015).

[79] S. Edwards, X. Liu, and N. Riga, "Creating repeatable computer science and networking experiments on shared, public testbeds," *SIGOPS Oper. Syst. Rev.*, vol. 49, no. 1, pp. 90–99, Jan. 2015, ISSN: 0163-5980.

[80] Epoch northbound networks, URL: http://northboundnetworks.com/epoch/ (visited on 02/13/2015).

[81] D. Erickson, "The beacon OpenFlow controller," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13, New York, NY, USA: ACM, 2013, pp. 13–18, ISBN: 978-1-4503-2178-5.

[82] ESNet 100g testbed, URL: https://es.net/network-r-and-d/experimental-network-testbeds/100g-sdn-testbed/ (visited on 02/13/2015).

[83] Exogeni, URL: http://www.exogeni.net/ (visited on 01/18/2015).

[84] ExoGENI hardware, URL: https://wiki.exogeni.net/doku.php?id=public:hardware:network:start (visited on 12/17/2014).

[85] ExoGENI software, URL: https://wiki.exogeni.net/doku.php?id=public:software:start (visited on 11/04/2014).

[86] Extreme networks BlackDiamond 8000 series, URL: http://www.extremenetworks.com/product/blackdiamond-8800-series (visited on 01/15/2015).

[87] Extreme networks BlackDiamond x8, URL: http://www.extremenetworks.com/product/blackdiamond-x-series (visited on 01/15/2015).

[88] Extreme networks e4g-200, URL: http://www.extremenetworks.com/product/e4g-200-cell-site-router (visited on 01/23/2015).

[89] Extreme networks e4g-400, URL: http://www.extremenetworks.com/product/e4g-400-cell-site-aggregation-router (visited on 01/23/2015).

[90] Extreme networks summit x430 series, URL: http://www.extremenetworks.com/product/summit-x430-series (visited on 01/15/2015).

[91] Extreme networks summit x440 series, URL: http://www.extremenetworks.com/product/summit-x440-series (visited on 01/15/2015).

[92] Extreme networks summit x460 series, URL: http://www.extremenetworks.com/product/summit-x460-series (visited on 01/15/2015).

[93] Extreme networks summit x480 series, URL: http://www.extremenetworks.com/product/summit-x480-series (visited on 01/15/2015).

[94] Extreme networks summit x670 series, URL: http://www.extremenetworks.com/product/summit-x670-series (visited on 01/15/2015).

[95] Extreme networks summit x770 series, URL: http://www.extremenetworks.com/product/summit-x770-series-2 (visited on 01/15/2015).

[96] *Extreme SDN architecture & development platform launch FAQ*, 2014. URL: http://learn.extremenetworks.com/rs/extreme/images/SDN-FAQ.pdf (visited on 01/15/2015).

[97] *ExtremeXOS 15.6 user guide*, Oct. 2014. URL: http://extrcdn.extremenetworks.com/wp-content/uploads/2014/10/EXOS_User_Guide_15_6.pdf (visited on 03/11/2015).

[98] EZchip technologies NP-4 network processor, URL: http://www.ezchip.com/p_np4.htm (visited on 01/15/2015).

[99] *FastIron ethernet switch SDN configuration guide (supporting FastIron software release 08.0.20a)*, Nov. 26, 2014. URL: http://www.brocade.com/downloads/documents/product_manuals/B_FastIron/FastIron_08020a_SDNGuide.pdf.

[100] Floodlight OpenFlow controller, Project Floodlight, URL: http://www.projectfloodlight.org/floodlight/ (visited on 04/28/2014).

[101] FlowSpace firewall, GlobalNOC - FSFW: FlowSpace Firewall, URL: http://globalnoc.iu.edu/sdn/fsfw.html (visited on 12/15/2014).

[102] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A network programming language," in *Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming*, ser. ICFP '11, New York, NY, USA: ACM, 2011, pp. 279–291, ISBN: 978-1-4503-0865-6.

[103] FSP 3000, URL: http://www.advaoptical.com/en/products/scalable-optical-transport/fsp-3000.aspx (visited on 01/15/2015).

[104] FSP network hypervisor, URL: http://www.advaoptical.com/en/products/scalable-optical-transport/fsp-network-hypervisor.aspx (visited on 01/15/2015).

[105] GEC16 public cisco GENI rack overview, URL: http://groups.geni.net/geni/attachment/wiki/GEC16Agenda/GENIRacks/GEC16\%20Public\%20Cisco\%20GENI\%20Rack\%20overview_v1.0.pptx (visited on 01/16/2015).

[106] GEC21 adgenda GENIOps, URL: http://groups.geni.net/geni/wiki/GEC21Agenda/GENIOps (visited on 01/09/2015).

[107] GENI layer1transport service, URL: http://groups.geni.net/geni/wiki/Layer1Transport (visited on 11/09/2014).

[108] GENI network core, URL: http://groups.geni.net/geni/wiki/NetworkCore (visited on 11/09/2014).

[109] GENI-ORCA, URL: https://geni-orca.renci.org/trac/wiki/orca-for-experimenter (visited on 11/04/2014).

[110] GENI rack security, URL: http://groups.geni.net/geni/wiki/GENIRacksHome/Security (visited on 11/09/2014).

[111] GENI slicer requirements, URL: http://groups.geni.net/geni/wiki/OpenFlow/Slicer/Requirements (visited on 01/09/2015).

[112] GENI tutorials: Intro to OpenFlow using OVS, URL: http://groups.geni.net/geni/wiki/GENIExperimenter/Tutorials/OpenFlowOVS/DesignSetup (visited on 01/27/2015).

[113] Gpeni, URL: https://wiki.ittc.ku.edu/gpeni/Main_Page (visited on 02/28/2015).

[114] GRAM network and software architecture description, URL: http://groups.geni.net/geni/wiki/GENIRacksHome/OpenGENIRacks/ArchitectureDescription (visited on 11/08/2014).

[115] D. Green. (Jul. 13, 2011). EZchip announces OpenFlow 1.1 implementations on its NP-4 100-gigabit network processor, EZchip, URL: http://www.ezchip.com/pr_110713.htm (visited on 01/15/2015).

[116] S. Gringeri, N. Bitar, and T. Xia, "Extending software defined network principles to include optical transport," *IEEE Communications Magazine*, vol. 51, no. 3, pp. 32–40, Mar. 2013, ISSN: 0163-6804.

[117] C. Guimaraes, D. Corujo, F. Silva, P. Frosi, A. Neto, and R. Aguiar, "IEEE 802.21-enabled entity title architecture for handover optimization," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2014, pp. 2671–2676.

[118] M. Gupta, J. Sommers, and P. Barford, "Fast, accurate simulation for SDN prototyping," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13, New York, NY, USA: ACM, 2013, pp. 31–36, ISBN: 978-1-4503-2178-5.

[119] S. Gutz, A. Story, C. Schlesinger, and N. Foster, "Splendid isolation: A slice abstraction for software-defined networks," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12, New York, NY, USA: ACM, 2012, pp. 79–84, ISBN: 978-1-4503-1477-0.

[120] (2012). GANT consortium, GANT FP7 Project, URL: `http://www.geant.net/` (visited on 11/20/2014).

[121] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '12, New York, NY, USA: ACM, 2012, pp. 253–264, ISBN: 978-1-4503-1775-7.

[122] M. Handley, O. Hodson, and E. Kohler, "Xorp: An open platform for network research," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 53–57, Jan. 2003, ISSN: 0146-4833.

[123] B. Hardekopf, K. Kwiaty, and S. Upadhyaya, "Secure and fault-tolerant voting in distributed systems," in *Aerospace Conference, 2001, IEEE Proceedings.*, vol. 3, 2001, 3/1117–3/1126 vol.3.

[124] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proceedings of the first workshop on Hot topics in software defined networks*, ACM, 2012, pp. 19–24.

[125] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12, New York, NY, USA: ACM, 2012, pp. 7–12, ISBN: 978-1-4503-1477-0.

[126] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. B. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 15, p. 17, 2008.

[127] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau, "Large-scale virtualization in the emulab network testbed," in *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, ser. ATC'08, Berkeley, CA, USA: USENIX Association, 2008, pp. 113–128.

[128] S. Higginbotham. (Apr. 2, 2014). Google launches andromeda, a software defined network underlying its cloud, URL: https://gigaom.com/2014/04/02/google-launches-andromeda-a-software-defined-network-underlying-its-cloud/ (visited on 02/03/2015).

[129] T. Hollingsworth. (Dec. 11, 2013). Dell n-series switches bring OpenFlow to campus networks, Network Computing, URL: http://www.networkcomputing.com/data-networking-management/dell-n-series-switches-bring-openflow-to/240164664 (visited on 01/22/2015).

[130] *HP 10500 switch data sheet*. URL: http://h20195.www2.hp.com/v2/GetDocument.aspx?docname=4AA3-6264ENW\&doctype=data\%20sheet\&doclang=EN_US\&searchquery=\&cc=us\&lc=en (visited on 03/11/2015).

[131] HP 10500 switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_10500_Switch_Series/index.aspx#tab=TAB2 (visited on 01/13/2015).

[132] HP 12500 switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_12500_Switch_Series/index.aspx (visited on 01/13/2015).

[133] HP 2920 switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_2920_Switch_Series/index.aspx (visited on 01/15/2015).

[134] HP 3500 and 3500 yl switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_3500_and_3500_yl_Switch_Series/ (visited on 12/17/2014).

[135] HP 3800 switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_3800_Switch_Series/index.aspx (visited on 01/15/2015).

[136] HP 5130 EI switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_5130_EI_Switch_Series/index.aspx (visited on 01/15/2015).

[137] HP 5400 zl switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_5400_zl_Switch_Series/index.aspx (visited on 12/17/2014).

[138] HP 5400r zl2 switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_5400R_zl2_Switch_Series/index.aspx (visited on 01/15/2015).

[139] HP 5500 EI switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_5500_EI_Switch_Series/index.aspx (visited on 01/15/2015).

[140] HP 5500 HI switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_5500_HI_Switch_Series/index.aspx (visited on 01/15/2015).

[141] HP 5900 switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_5900_Switch_Series/index.aspx (visited on 01/13/2015).

[142] HP 5920 switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_5920_Switch_Series/index.aspx (visited on 01/13/2015).

[143] HP 6600 switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_6600_Switch_Series/index.aspx (visited on 12/17/2014).

[144] HP 8200 zl switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_8200_zl_Switch_Series/index.aspx (visited on 01/13/2015).

[145] HP FlexFabric 11900 switch series, URL: http://h17007.www1.hp.com/docs/interop/2013/4AA4-6497ENW.pdf (visited on 01/13/2015).

[146] HP FlexFabric 12900 switch series, URL: http://h17007.www1.hp.com/docs/interop/2013/4AA4-6499ENW.pdf (visited on 01/13/2015).

[147] HP FlexFabric 5700 switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_FlexFabric_5700_Switch_Series/index.aspx (visited on 01/15/2015).

[148] HP FlexFabric 5930 switch series, URL: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_FlexFabric_5930_Switch_Series/index.aspx (visited on 01/13/2015).

[149] *HP OpenFlow protocol overview*, Sep. 2013. URL: http://h17007.www1.hp.com/docs/networking/solutions/sdn/devcenter/03_-_HP_OpenFlow_Technical_Overview_TSG_v1_2013-10-01.pdf (visited on 01/25/2015).

[150] HP SDN switches portfolio, URL: http://h17007.www1.hp.com/us/en/networking/solutions/technology/sdn/portfolio.aspx#infrastructure (visited on 01/13/2015).

[151] *HP switch software OpenFlow administrators guide for k/KA/WB 15.15*, Mar. 2014. URL: http://h20564.www2.hp.com/hpsc/doc/public/display?docId=c04217797 (visited on 01/24/2015).

[152] D. Y. Huang, K. Yocum, and A. C. Snoeren, "High-fidelity switch models for software-defined network emulation," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13, New York, NY, USA: ACM, 2013, pp. 43–48, ISBN: 978-1-4503-2178-5.

[153] S. Huang, H. Xu, Y. Xin, L. Brieger, R. Moore, and A. Rajasekar, "A framework for integration of rule-oriented data management policies with network policies," in *Research*

and Educational Experiment Workshop (GREE), 2014 Third GENI, Mar. 2014, pp. 71–72.

[154] *Huawei agile switch advertorial*, 2013. URL: http://enterprise.huawei.com/ilink/cnenterprise/download/HW_279180 (visited on 01/26/2015).

[155] *Huawei cloud fabric-cloud connect data center solution*, 2014. URL: http://enterprise.huawei.com/ilink/cnenterprise/download/HW_345117 (visited on 01/26/2015).

[156] Huawei CloudEngine 12800 series, URL: http://e.huawei.com/en/products/enterprise-networking/switches/data-center-switches/ce12800 (visited on 01/27/2015).

[157] Huawei CloudEngine 5800 series, URL: http://e.huawei.com/en/products/enterprise-networking/switches/data-center-switches/ce5800 (visited on 01/27/2015).

[158] Huawei CloudEngine 6800 series, URL: http://e.huawei.com/en/products/enterprise-networking/switches/data-center-switches/ce6800 (visited on 01/27/2015).

[159] Huawei CloudEngine 7800 series, URL: http://e.huawei.com/en/products/enterprise-networking/switches/data-center-switches/ce7800 (visited on 01/27/2015).

[160] Huawei s12700 series, URL: http://e.huawei.com/en/products/enterprise-networking/switches/campus-switches/s12700 (visited on 01/27/2015).

[161] Huawei s7700 series, URL: http://e.huawei.com/en/products/enterprise-networking/switches/campus-switches/s7700 (visited on 01/27/2015).

[162] Huawei s9700 series, URL: http://e.huawei.com/en/products/enterprise-networking/switches/campus-switches/s9700 (visited on 01/27/2015).

[163] (Jul. 15, 2014). IBM RackSwitch g8264, URL: http://www.redbooks.ibm.com/abstracts/tips0815.html (visited on 12/17/2014).

[164] "IEEE draft standard for local and metropolitan area networks, virtual bridged local area networks, amendment 4: Provider bridges," IEEE 802.1ad.

[165] Implementing OpenFlow agent, Cisco, URL: http://cisco.com/c/en/us/td/docs/routers/asr9000/software/asr9k_r5-1/sysman/configuration/guide/b-sysman-cg51xasr9k/b-sysman-cg51xasr9k_chapter_01110.html (visited on 12/19/2014).

[166] (Nov. 5, 2014). Infonetics forecasts carrier SDN and NFV market to reach $11 billion by 2018, Infonetics Research, URL: http://www.infonetics.com/pr/2014/Carrier-SDN-NFV-Market-Highlights.asp (visited on 02/03/2015).

[167] Intel ethernet switch FM5000 product brief, URL: http://www.intel.com/content/www/us/en/switch-silicon/ethernet-switch-fm5224-brief.html (visited on 01/15/2015).

[168] Intel ethernet switch FM5000/FM6000 series, URL: http://www.intel.com/content/www/us/en/switch-silicon/ethernet-switch-fm5000-fm6000-series.html (visited on 01/15/2015).

[169] Intel ethernet switch FM6000 product brief, URL: https://www-ssl.intel.com/content/www/us/en/switch-silicon/ethernet-switch-fm6000-series-brief.html (visited on 01/15/2015).

[170] *IP8800/s3640 software manual OpenFlow feature guide*, May 2010. URL: http://support.necam.com/kbtools/sDocs.cfm?id=fcbdcb3e-45fa-4ec4-9311-215bd9ab9f81 (visited on 01/15/2015).

[171] Is cisco application centric infrastructure an SDN technology? Cisco, URL: http://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-733456.html (visited on 02/03/2015).

[172] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hlzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, Aug. 2013, ISSN: 0146-4833.

[173] D. Jin and D. M. Nicol, "Parallel simulation of software defined networks," in *Proceedings of the 2013 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, ser. SIGSIM-PADS '13, New York, NY, USA: ACM, 2013, pp. 91–102, ISBN: 978-1-4503-1920-1.

[174] *Juiper EX9200 ethernet switch data sheet*. URL: http://www.juniper.net/assets/us/en/local/pdf/datasheets/1000432-en.pdf (visited on 01/16/2015).

[175] *Juiper QFX5100 ethernet switch data sheet*. URL: http://www.juniper.net/assets/us/en/local/pdf/datasheets/1000480-en.pdf (visited on 01/15/2015).

[176] Juniper networks EX4500 switch, URL: http://www.juniper.net/us/en/products-services/switching/ex-series/ex4500/ (visited on 01/16/2015).

[177] Juniper networks EX9200 switch, URL: http://www.juniper.net/us/en/products-services/switching/ex-series/ex9200/ (visited on 01/16/2015).

[178] Juniper networks MX240 3d universal edge router, URL: http://www.juniper.net/us/en/products-services/routing/mx-series/mx240/ (visited on 01/16/2015).

[179] Juniper networks MX480 3d universal edge router, URL: http://www.juniper.net/us/en/products-services/routing/mx-series/mx480/ (visited on 01/16/2015).

[180] Juniper networks MX80 3d universal edge router, URL: http://www.juniper.net/us/en/products-services/routing/mx-series/mx80/ (visited on 01/16/2015).

[181] Juniper networks MX960 3d universal edge router, URL: http://www.juniper.net/us/en/products-services/routing/mx-series/mx960/ (visited on 01/16/2015).

[182] Juniper networks QFX5100 switch, URL: http://www.juniper.net/tw/tc/products-services/switching/qfx-series/qfx5100/ (visited on 01/15/2015).

[183] *Junos OS OpenFlow feature guide*, Oct. 30, 2014. URL: http://www.juniper.net/techpubs/en_US/junos14.2/information-products/pathway-pages/junos-sdn/junos-sdn-openflow.pdf (visited on 01/23/2015).

[184] Y. Kanaumi, S.-i. Saito, E. Kawai, S. Ishii, K. Kobayashi, and S. Shimojo, "Deployment and operation of wide-area hybrid OpenFlow networks," in *2012 IEEE Network Operations and Management Symposium (NOMS)*, Apr. 2012, pp. 1135–1142.

[185] A. Kangarlou, D. Xu, U. Kozat, P. Padala, B. Lantz, and K. Igarashi, "In-network live snapshot service for recovering virtual infrastructures," *IEEE Network*, vol. 25, no. 4, pp. 12–19, Jul. 2011, ISSN: 0890-8044.

[186] S. Keshav, *REAL: A network simulator*. University of California, 1988.

[187] A. Khan and N. Dave, "Enabling hardware exploration in software-defined networking: A flexible, portable OpenFlow switch," in *2013 IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Apr. 2013, pp. 145–148.

[188] D. Kim, J. Kim, G. Wang, J.-H. Park, and S.-H. Kim, "K-GENI testbed deployment and federated meta operations experiment over GENI and KREONET," *Computer Networks*, Special issue on Future Internet Testbeds Part I, vol. 61, pp. 39–50, Mar. 14, 2014, ISSN: 1389-1286.

[189] J. Kim, B. Cha, J. Kim, N. L. Kim, G. Noh, Y. Jang, H. G. An, H. Park, J. Hong, D. Jang, and others, "OF@ TEIN: An OpenFlow-enabled SDN testbed over international SmartX rack sites," *Proceedings of the Asia-Pacific Advanced Network*, vol. 36, pp. 17–22, 2013.

[190] R. Kloti, V. Kotronis, and P. Smith, "Openflow: A security analysis," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, Oct. 2013, pp. 1–6.

[191] M. Kobayashi, S. Seetharaman, G. Parulkar, G. Appenzeller, J. Little, J. van Reijendam, P. Weissmann, and N. McKeown, "Maturing of OpenFlow and software-defined networking through deployments," *Computer Networks*, Special issue on Future Internet Testbeds - Part I, vol. 61, pp. 151–175, Mar. 14, 2014, ISSN: 1389-1286.

[192] M. Koerner, "The OFELIA TUB-island an europe-wide connected OpenFlow testbed," in *2013 IEEE 38th Conference on Local Computer Networks (LCN)*, Oct. 2013, pp. 320–323.

[193] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10, Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6.

[194] V. Kotronis, L. Bergesio, B. Puype, N. Efstathiou, and M. Channegowda, *OFELIA first report on implementation testing and operation of federated islands*, M. Channegowda, Ed., 2010. URL: `http://www.fp7-ofelia.eu/assets/Public-Deliverables/OELIAD2.4-First-report-on-implementation-testing-and-operation-of-Federated-Islandsfinal.pdf` (visited on 01/28/2015).

[195] D. Kreutz, F. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015, ISSN: 0018-9219.

[196] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13, New York, NY, USA: ACM, 2013, pp. 55–60, ISBN: 978-1-4503-2178-5.

[197] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX, New York, NY, USA: ACM, 2010, 19:1–19:6, ISBN: 978-1-4503-0409-2.

[198] A. Lara, A. Kolasani, and B. Ramamurthy, "Simplifying network management using software defined networking and OpenFlow," in *2012 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, 2012, pp. 24–29.

[199] D. Li, X. Hong, and J. Bowman, "Evaluation of security vulnerabilities by using ProtoGENI as a launchpad," in *2011 IEEE Global Telecommunications Conference (GLOBECOM 2011)*, Dec. 2011, pp. 1–6.

[200] D. Li, X. Hong, and D. Witt, "ProtoGENI, a prototype GENI under security vulnerabilities: An experiment-based security study," *IEEE Systems Journal*, vol. 7, no. 3, pp. 478–488, Sep. 2013, ISSN: 1932-8184.

[201] List of SDN controller vendors & SDN controllers, SDxCentral, URL: `https://www.sdxcentral.com/resources/sdn/sdn-controllers/sdn-controllers-comprehensive-list/` (visited on 02/12/2015).

[202] B. Lynch. (Jun. 15, 2013). OpenFlow - can it scale? SDNCentral, URL: `https://www.sdncentral.com/technology/openflow-sdn/2013/06/` (visited on 12/03/2014).

[203] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "Vxlan: A framework for overlaying virtualized layer 2 networks over layer 3 networks," *RFC 7348*, Aug. 2014, ISSN: 2070-1721.

[204] A. Malishevskiy, D. Gurkan, L. Dane, R. Narisetty, S. Narayan, and S. Bailey, "OpenFlow based network management with visualization of managed elements," in *Research and Educational Experiment Workshop (GREE), 2014 Third GENI*, Mar. 2014, pp. 73–74.

[205] J. Mambretti, J. Chen, and F. Yeh, "Creating environments for innovation: Designing and implementing advanced experimental network research testbeds based on the global lambda integrated facility and the StarLight exchange," *Computer Networks*, Special issue on Future Internet Testbeds Part I, vol. 61, pp. 118–131, Mar. 14, 2014, ISSN: 1389-1286.

[206] S. McCanne and S. Floyd, *NS network simulator*. 1995.

[207] R. McGeer, J. Blaine, N. Watts, N. Krishnan, Bavier, R. Ricci, and J. Mambretti, *InstaGENI design document*, Nov. 27, 2012. URL: `http://groups.geni.net/geni/raw-attachment/wiki/GENIRacksHome/InstageniRacks/InstaGENI\%20Design\%20Document\%20Rick\%202-15\%282\%29.pdf` (visited on 11/05/2014).

[208] N. McKeown, "Software-defined networking," *INFOCOM keynote talk*, vol. 17, no. 2, pp. 30–32, 2009.

[209] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008, ISSN: 0146-4833.

[210] McKeown group OpenFlow WiFi network (aka "ofwifi"), URL: `https://openflow.stanford.edu/pages/viewpage.action?pageId=3571812` (visited on 12/10/2014).

[211] J. Medved, D. Meyer, and A. McLachlan, "MPLS-TP pseudowire configuration using OpenFlow 1.3," 2012.

[212] N. Melazzi, A. Detti, G. Mazza, G. Morabito, S. Salsano, and L. Veltri, "An OpenFlow-based testbed for information centric networking," in *Future Network Mobile Summit (FutureNetw), 2012*, Jul. 2012, pp. 1–9.

[213] Mininet: An instant virtual network on your laptop (or other PC), URL: `http://mininet.org/` (visited on 02/13/2015).

[214] J. C. Mogul and G. Minshall, "Rethinking the TCP nagle algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 1, pp. 6–20, Jan. 2001, ISSN: 0146-4833.

[215] J. Nagle, "Congestion control in IP/TCP internetworks," 1984.

[216] J. Naous, D. Erickson, G. A. Covington, G. Appenzeller, and N. McKeown, "Implementing an OpenFlow switch on the NetFPGA platform," in *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ser. ANCS '08, New York, NY, USA: ACM, 2008, pp. 1–9, ISBN: 978-1-60558-346-4.

[217] J. Naous, P. Kazemian, R. Sherwood, S. Seetharaman, N. McKeown, G. Parulkar, and G. Appenzeller, "Expedient: A centralized pluggable clearinghouse to manage GENI experiments," Stanford University, Jan. 2010.

[218] NEC ProgrammableFlow PF5240 switch, URL: `http://www.necam.com/sdn/doc.cfm?t=PFlowPF5240Switch` (visited on 01/15/2015).

[219] NEC ProgrammableFlow PF5248 switch, URL: `http://www.necam.com/sdn/doc.cfm?t=PFlowPF5248Switch` (visited on 01/15/2015).

[220] NEC ProgrammableFlow PF5820 switch, URL: `http://www.necam.com/sdn/doc.cfm?t=PFlowPF5820Switch` (visited on 01/15/2015).

[221] NetFPGA 10g, URL: `http://netfpga.org/2014/#/systems/3netfpga-10g/details/` (visited on 01/15/2015).

[222] NetFPGA 1g, NetFPGA, URL: `http://netfpga.org/2014/#/systems/4netfpga-1g/details/` (visited on 12/17/2014).

[223] NetFPGA CML, URL: `http://netfpga.org/2014/#/systems/2netfpga-1g-cml/details/` (visited on 01/15/2015).

[224] NetFPGA SUME, URL: `http://netfpga.org/2014/#/systems/1netfpga-sume/details/` (visited on 01/15/2015).

[225] K. Nguyen, Q. T. Minh, and S. Yamada, "A software-defined networking approach for disaster-resilient WANs," in *2013 22nd International Conference on Computer Communications and Networks (ICCCN)*, 2013, pp. 1–5.

[226] NoviFlow - NoviSwitch, URL: `http://noviflow.com/products/noviswitch/` (visited on 01/15/2015).

[227] NoviFlow - NoviWare, URL: `http://noviflow.com/products/noviware/` (visited on 01/15/2015).

[228] NOX OpenFlow controller, URL: `http://www.noxrepo.org/` (visited on 04/28/2014).

[229] Ns-3 OpenFlow switch support, URL: `http://www.nsnam.org/docs/release/3.13/models/html/openflow-switch.html` (visited on 02/13/2015).

[230] NSF campus cyberinfrastructure - data, networking, and innovation program, URL: `https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=504748` (visited on 02/06/2015).

[231] OCF wiki, GitHub, URL: https://github.com/fp7-ofelia/ocf (visited on 12/08/2014).

[232] OFELIA facility and islands, URL: http://www.fp7-ofelia.eu/ofelia-facility-and-islands/ (visited on 11/20/2014).

[233] OF@TEIN, URL: http://trac.nm.gist.ac.kr/oftein (visited on 02/15/2015).

[234] F. de Oliveira Silva, D. Corujo, C. Guimaraes, J. H. d. S. Pereira, P. F. Rosa, S. T. Kofuji, A. Neto, and R. Aguiar, "Enabling network mobility by using IEEE 802.21 integrated with the entity title architecture," in *IV Experimental Research Workshop of the Future Internet*, Brazilian Computer Society, 2013, pp. 29–34.

[235] onePK FAQ, Cisco Communities, URL: https://communities.cisco.com/docs/DOC-53411 (visited on 12/19/2014).

[236] ONF NBI working group charter, URL: http://www.onfsdninterfaces.org/index.php/onf-nbi-leadership-roundtable-materials/6-onf-nbi-work-group-charter (visited on 02/15/2015).

[237] Open MuL foundation, Open MuL Foundation Home, URL: http://www.openmul.org/ (visited on 02/12/2015).

[238] Open networking foundation, Open Networking Foundation, URL: https://www.opennetworking.org/ (visited on 01/30/2015).

[239] Open virtualized WiMAX base station node for GENI wide-area wireless deployments, URL: http://groups.geni.net/geni/wiki/WiMAX.

[240] Open vSwitch, URL: http://openvswitch.org/ (visited on 12/16/2014).

[241] Opendaylight, URL: http://www.opendaylight.org/ (visited on 12/13/2014).

[242] OpenFlow 1.3 software switch by CPqD, URL: https://cpqd.github.io/ofsoftswitch13/ (visited on 02/12/2015).

[243] OpenFlow controllers in GENI, URL: http://groups.geni.net/geni/wiki/OpenFlow/Controllers (visited on 01/08/2015).

[244] (Oct. 30, 2014). OpenFlow support on devices running junos OS, Technical Documentation - Juniper Networks, URL: http://www.juniper.net/documentation/en_US/release-independent/junos/topics/reference/general/junos-sdn-openflow-supported-platforms.html (visited on 01/15/2015).

[245] *OpenFlow switch specification version 1.0*, Dec. 31, 2009. URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf (visited on 02/03/2015).

[246] *OpenFlow switch specification version 1.2*, Dec. 5, 2011. URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.2.pdf (visited on 01/16/2015).

[247] *OpenFlow switch specification version 1.3*, Jun. 25, 2012. URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf (visited on 02/03/2015).

[248] *OpenFlow switch specification version 1.5*, Dec. 19, 2014. URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf (visited on 02/03/2015).

[249] OpenFlow/FOAM GENI, URL: http://groups.geni.net/geni/wiki/OpenFlow/FOAM (visited on 11/04/2014).

[250] Opengeni, URL: http://www.opengeni.net/ (visited on 11/08/2014).

[251] Opflex: An open policy protocol, Cisco, URL: http://cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-731302.html (visited on 02/12/2015).

[252] *Outcomes of the hybrid working group*, Mar. 2013. URL: https://www.opennetworking.org/images/stories/downloads/working-groups/summary-hybrid.pdf (visited on 01/26/2015).

[253] OVS quantum plugin documentation, URL: http://www.openvswitch.org/openstack/documentation/ (visited on 03/03/2015).

[254] B. Parhami, "Voting algorithms," *IEEE Transactions on Reliability*, vol. 43, no. 4, pp. 617–629, Dec. 1994, ISSN: 0018-9529.

[255] V. Paxson and S. Floyd, "Why we don't know how to simulate the internet," in *Simulation Conference, 1997., Proceedings of the 1997 Winter*, Dec. 1997, pp. 1037–1044.

[256] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 59–64, Jan. 2003, ISSN: 0146-4833.

[257] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker, "Extending networking into the virtualization layer.," in *Hotnets*, 2009.

[258] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain SDN controllers," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–4.

[259] Pica8 1g/10g/40g open switches, URL: http://www.pica8.com/open-switching/1gbe-10gbe-40gbe-open-switches.php (visited on 12/17/2014).

[260] PicOS - pica8's switch operating system, URL: http://www.pica8.com/open-switching/open-switching-overview.php (visited on 12/17/2014).

[261] P. Porras. (2014). Toward a more secure SDN control layer - SRI international's view, SDNCentral, URL: http://www.sdncentral.com/education/toward-secure-sdn-control-layer/2013/10/ (visited on 04/19/2014).

[262] P. Porras, S. Cheung, M. Fong, K. Skinner, and V. Yegneswaran, "Securing the software-defined network control layer," in *Proceedings of the 2015 Network and Distributed System Security Symposium (NDSS)*, San Diego, California., Feb. 2015.

[263] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12, New York, NY, USA: ACM, 2012, pp. 121–126, ISBN: 978-1-4503-1477-0.

[264] POX OpenFlow controller, URL: http://www.noxrepo.org/pox/documentation/ (visited on 11/08/2014).

[265] Protogeni, URL: http://www.protogeni.net/ (visited on 02/26/2015).

[266] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13, New York, NY, USA: ACM, 2013, pp. 27–38, ISBN: 978-1-4503-2056-6.

[267] Release notes for cisco ASR 9000 series aggregation services routers for cisco IOS XR software release 5.2.0, Cisco, URL: http://cisco.com/c/en/us/td/docs/routers/asr9000/software/asr9k_r5-2/general/release/notes/reln_520a9k.html (visited on 12/19/2014).

[268] L. Rizzo, "Dummynet: A simple approach to the evaluation of network protocols," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 1, pp. 31–41, Jan. 1997, ISSN: 0146-4833.

[269] E. Rosen, A. Viswanathan, R. Callon, and others, "Multiprotocol label switching architecture," IETF, RFC 3031, Jan. 2001.

[270] M. Rouse. (Feb. 2008). What is pseudowire? URL: http://searchnetworking.techtarget.com/definition/pseudowire (visited on 02/11/2015).

[271] A. Roy, M. Bari, M. Zhani, R. Ahmed, and R. Boutaba, "Design and management of DOT: A distributed OpenFlow testbed," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–9.

[272] T. Salmito, L. Ciuffo, I. Machado, M. Salvador, M. Stanton, N. Rodriguez, A. Abelem, L. Bergesio, S. Sallent, and L. Baron, "FIBRE - an international testbed for future

internet experimentation," in *Simpsio Brasileiro de Redes de Computadores e Sistemas Distribudos-SBRC 2014*, 2014, p–969.

[273] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Mazza, G. Morabito, A. Araldo, L. Linguaglossa, and L. Veltri, "Supporting content networking in software defined networks," Technical Report, available at http://netgroup. uniroma2. it/TR/CONETSDN. pdf, 2012.

[274] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito, and L. Veltri, "Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed," *Computer Networks*, Information Centric Networking, vol. 57, no. 16, pp. 3207–3221, Nov. 13, 2013, ISSN: 1389-1286.

[275] E. Salvadori, R. Doriguzzi Corin, A. Broglio, and M. Gerola, "Generalizing virtual network topologies in OpenFlow-based networks," in *2011 IEEE Global Telecommunications Conference (GLOBECOM 2011)*, Dec. 2011, pp. 1–6.

[276] M. Santuari, R. Doriguzzi-Corin, M. Gerola, E. Salvadori, U. Toseef, A. Zaalouk, K. Dombek, D. Parniewicz, A. Hammad, M. Rashidi-Fard, E. Jacob, and J. Matias, "Leading the OFELIA facility beyond OpenFlow 1.0 experimentations," in *2014 Third European Workshop on Software Defined Networks (EWSDN)*, Sep. 2014, pp. 119–120.

[277] D. Schwerdel, B. Reuther, T. Zinner, P. Mller, and P. Tran-Gia, "Future internet research and experimentation: The g-lab approach," *Computer Networks*, Special issue on Future Internet Testbeds Part I, vol. 61, pp. 102–117, Mar. 14, 2014, ISSN: 1389-1286.

[278] C. Scott, A. Wundsam, B. Raghavan, A. Panda, A. Or, J. Lai, E. Huang, Z. Liu, A. El-Hassany, S. Whitlock, H. Acharya, K. Zarifis, and S. Shenker, "Troubleshooting blackbox SDN control software with minimal causal sequences," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14, New York, NY, USA: ACM, 2014, pp. 395–406, ISBN: 978-1-4503-2836-4.

[279] S. Scott-Hayward, C. Kane, and S. Sezer, "Operationcheckpoint: SDN application control," in *2014 IEEE 22nd International Conference on Network Protocols (ICNP)*, Oct. 2014, pp. 618–623.

[280] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "Sdn security: A survey," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, Nov. 2013, pp. 1–7.

[281] (Apr. 24, 2013). SDN market to reach $35b by 2018, Plexxi, URL: `http://www.plexxi.com/2013/04/sdn-market-to-reach-35b-by-2018/` (visited on 02/03/2015).

[282] (Aug. 21, 2014). SDN market worth $8 billion by 2018 says IDC, URL: `http://www.lightwaveonline.com/articles/2014/08/sdn-market-worth-8-billion-by-2018-says-idc.html` (visited on 02/03/2015).

[283] SDN/OpenFlow products, Open Networking Foundation, URL: https://www.opennetworking.org/sdn-openflow-products (visited on 02/17/2015).

[284] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, Jul. 2013, ISSN: 0163-6804.

[285] S. Shah, J. Faiz, M. Farooq, A. Shafi, and S. Mehdi, "An architectural evaluation of SDN controllers," in *2013 IEEE International Conference on Communications (ICC)*, Jun. 2013, pp. 3504–3508.

[286] A. R. Sharafat, S. Das, G. Parulkar, and N. McKeown, "MPLS-TE and MPLS VPNS with OpenFlow," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11, New York, NY, USA: ACM, 2011, pp. 452–453, ISBN: 978-1-4503-0797-0.

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

[288] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the production network be the testbed?" In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10, Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6.

[289] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13, New York, NY, USA: ACM, 2013, pp. 413–424, ISBN: 978-1-4503-2477-9.

[290] SNAC home, SNAC Home, URL: http://www.openflowhub.org/display/Snac/SNAC+Home (visited on 12/11/2014).

[291] J. Sommers, R. Bowden, B. Eriksson, P. Barford, M. Roughan, and N. Duffield, "Efficient network-wide flow record generation," in *2011 Proceedings IEEE INFOCOM*, Apr. 2011, pp. 2363–2371.

[292] 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.

[293] Stanford deployment, URL: https://openflow.stanford.edu/display/SDEP/Home (visited on 12/17/2014).

[294] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz, "Overqos: An overlay based architecture for enhancing internet QoS.," in *NSDI*, vol. 4, 2004, p. 6.

[295] *Summit x440 series data sheet*. URL: http://learn.extremenetworks.com/rs/extreme/images/Summit-X440x-DS.pdf (visited on 01/15/2015).

[296] *Summit x460 series data sheet*. URL: http://learn.extremenetworks.com/rs/extreme/images/Summit-x460-DS.pdf (visited on 01/15/2015).

[297] *Summit x480 series data sheet*. URL: http://learn.extremenetworks.com/rs/extreme/images/SummitX480-DS.pdf (visited on 01/15/2015).

[298] *Summit x670 series data sheet*. URL: http://learn.extremenetworks.com/rs/extreme/images/Summit-X670-DS.pdf (visited on 01/15/2015).

[299] *Summit x770 series data sheet*. URL: http://learn.extremenetworks.com/rs/extreme/images/Summit-X770-DS.pdf (visited on 01/15/2015).

[300] Support@necam.com, *IP8800 OpenFlow support*, E-mail, Jan. 12, 2015.

[301] M. Su, L. Bergesio, H. Woesner, T. Rothe, A. Kpsel, D. Colle, B. Puype, D. Simeonidou, R. Nejabati, M. Channegowda, M. Kind, T. Dietz, A. Autenrieth, V. Kotronis, E. Salvadori, S. Salsano, M. Krner, and S. Sharma, "Design and implementation of the OFELIA FP7 facility: The european OpenFlow testbed," *Computer Networks*, Special issue on Future Internet Testbeds Part I, vol. 61, pp. 132–150, Mar. 14, 2014, ISSN: 1389-1286.

[302] Y. Sverdlik. (Jun. 3, 2014). Facebook testing broadcom's open compute switches in production, Data Center Knowledge, URL: http://www.datacenterknowledge.com/archives/2014/06/03/facebook-testing-broadcoms-open-compute-switches-production/ (visited on 02/18/2015).

[303] A.-W. Tam, K. Xi, and H. Chao, "Use of devolved controllers in data center networks," in *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2011, pp. 596–601.

[304] B. Tarvin, *Dell SDN switches*, E-mail, Jan. 19, 2015.

[305] Testbed barcelona - OFELIA, URL: https://alpha.fp7-ofelia.eu/doc/index.php/Testbed_Barcelona (visited on 11/21/2014).

[306] Testbed belgium virtual wall - OFELIA, URL: https://alpha.fp7-ofelia.eu/doc/index.php/Testbed_Belgium_Virtual_Wall (visited on 11/21/2014).

[307] Testbed berlin campus - OFELIA, URL: https://alpha.fp7-ofelia.eu/doc/index.php/Testbed_Berlin_Campus (visited on 11/21/2014).

[308] Testbed brazil - OFELIA, URL: https://alpha.fp7-ofelia.eu/doc/index.php/Testbed_Brazil (visited on 11/21/2014).

[309] Testbed bristol - OFELIA, URL: https://alpha.fp7-ofelia.eu/doc/index.php/Testbed_Bristol (visited on 11/21/2014).

[310] Testbed catania - OFELIA, URL: https://alpha.fp7-ofelia.eu/doc/index.php/Testbed_Catania (visited on 11/21/2014).

[311] Testbed trento - OFELIA, URL: https://alpha.fp7-ofelia.eu/doc/index.php/Testbed_Trento (visited on 11/21/2014).

[312] Testbed zurich - OFELIA, URL: https://alpha.fp7-ofelia.eu/doc/index.php/Testbed_Zurich (visited on 11/21/2014).

[313] The 2013 national survey of computing and information technology, Campus Computing, URL: http://www.campuscomputing.net/item/2013-campus-computing-survey-0 (visited on 12/21/2014).

[314] The STRIDE threat model, URL: https://msdn.microsoft.com/en-us/library/ee823878\%28v=cs.20\%29.aspx (visited on 02/25/2015).

[315] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for OpenFlow," in *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, ser. INM/WREN'10, Berkeley, CA, USA: USENIX Association, 2010, pp. 3–3.

[316] Trema, URL: https://trema.github.io/trema/ (visited on 01/08/2015).

[317] Trentino TestBed, URL: http://www.create-net.org/research/testbed (visited on 11/21/2014).

[318] M. Tsugawa, A. Matsunaga, and J. A. B. Fortes, "Cloud computing security: What changes with software-defined networking?" In *Secure Cloud Computing*, S. Jajodia, K. Kant, P. Samarati, A. Singhal, V. Swarup, and C. Wang, Eds. Springer New York, Jan. 1, 2014, pp. 77–93, ISBN: 978-1-4614-9277-1, 978-1-4614-9278-8.

[319] (Apr. 21, 2014). University explores software-defined networking to provide research opportunities, URL: https://oit.ncsu.edu/news-releases/university-explores-software-defined-networking-to-provide-research-opportunities (visited on 02/17/2015).

[320] J. Vanian. (Nov. 14, 2014). Facebook shows the promise of SDN with new networking tech, URL: https://gigaom.com/2014/11/14/facebook-shows-the-promise-of-sdn-with-new-networking-tech/ (visited on 02/03/2015).

[321] L. Veltri, G. Morabito, S. Salsano, N. Blefari-Melazzi, and A. Detti, "Supporting information centric functionality in software defined networks," in *2012 IEEE International Conference on Communications (ICC)*, Jun. 2012, pp. 6645–6650.

[322] M. Wagner. (Nov. 20, 2014). Verizon advances SDN strategy with bare-metal bet, Light Reading, URL: http://www.lightreading.com/carrier-sdn/sdn-technology/verizon-advances-sdn-strategy-with-bare-metal-bet/d/d-id/712170 (visited on 02/03/2015).

[323] A. Wang, M. Iyer, R. Dutta, G. Rouskas, and I. Baldine, "Network virtualization: Technologies, perspectives, and frontiers," *Journal of Lightwave Technology*, vol. 31, no. 4, pp. 523–537, Feb. 2013, ISSN: 0733-8724.

[324] B. Wang, Y. Zheng, W. Lou, and Y. Hou, "DDoS attack protection in the era of cloud computing and software-defined networking," in *2014 IEEE 22nd International Conference on Network Protocols (ICNP)*, Oct. 2014, pp. 624–629.

[325] H. Wang, L. Xu, and G. Gu, "OF-GUARD: A DoS attack prevention extension in software-defined networks," Mar. 2, 2014.

[326] S.-Y. Wang, C.-L. Chou, and C.-M. Yang, "EstiNet openflow network simulator and emulator," *IEEE Communications Magazine*, vol. 51, no. 9, pp. 110–117, Sep. 2013, ISSN: 0163-6804.

[327] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, "Towards a secure controller platform for OpenFlow applications," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13, New York, NY, USA: ACM, 2013, pp. 171–172, ISBN: 978-1-4503-2178-5.

[328] P. Wette, M. Drxler, A. Schwabe, F. Wallaschek, M. Zahraee, and H. Karl, "Maxinet: Distributed emulation of software-defined networks," in *Networking Conference, 2014 IFIP*, Jun. 2014, pp. 1–9.

[329] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 255–270, SI Dec. 2002, ISSN: 0163-5980.

[330] J. Xiaofeng, "Huawei SDN open programmability system (OPS)," Huawei Enterprise Networking Congress 2013 (HENC 2013), Shanghai World Expo Center, China, Sep. 2, 2013,

[331] H. Yamanaka, E. Kawai, S. Ishii, and S. Shimojo, "Autovflow: Autonomous virtualization for wide-area OpenFlow networks," in *2014 Third European Workshop on Software Defined Networks (EWSDN)*, Sep. 2014, pp. 67–72.

[332] Z. Yan and C. Prehofer, "Autonomic trust management for a component-based software system," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 6, pp. 810–823, Nov. 2011, ISSN: 1545-5971.

[333] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown, "Openroads: Empowering research in mobile networks," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 125–126, Jan. 2010, ISSN: 0146-4833.

[334] K.-K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, and N. McKeown, "The stanford OpenRoads deployment," in *Proceedings of the 4th ACM International Workshop on Experimental Evaluation and Characterization*, ser. WINTECH '09, New York, NY, USA: ACM, 2009, pp. 59–66, ISBN: 978-1-60558-740-0.

[335] S. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, Feb. 2013, ISSN: 0163-6804.

[336] H. Zeng, P. Kazemian, G. Varghese, and N. McKeown, "Automatic test packet generation," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 554–566, Apr. 2014, ISSN: 1063-6692.

APPENDIX

Appendix A

Software-Defined Network Hardware Vendors

A.1 Discovering SDN Vendors

To assist further in the analysis of existing SDNs, we compiled an incomplete list of available SDN hardware. While this list is not exhaustive, it provides a representative sample of available hardware at the time of this writing. This list was compiled from a variety of sources including hardware used within each case study, suggestions from the Open Networking Foundation [283], products used in research papers, and hardware found through online searches. There are some limits to this methodology. For instance, this list is very US-centric due to the difficulties of discovering foreign SDN switch vendors. Other issues included the lack of documentation on OpenFlow support, uninformed sales engineers, and research prototypes that are unavailable to the public. Notably, Huawei was unresponsive to all attempted communications and, therefore, some supported products may be missing from this table.

A.2 Available Hardware

Table A.1 presents a semi-comprehensive list of available SDN hardware. The information provided is up-to-date as of January 2015. We provide the vendor, product name, type of device, latest OpenFlow version the device supports (if any), a brief product description, any other pertinent information, and a reference. The hardware type varies from typical switches and chassis to optical transport systems and network processors. We cannot guarantee that all SDN compatible products from a listed vendor are represented within the table due to the issues described in Section A.1.

Table A.1: Semi-comprehensive list of SDN hardware.

Vendor	Product	Type	Latest OF Version	Description	Additional Information	Reference
ADVA Optical Networking	FSP 3000	Optical Transport	N/A	Optical transport device.	Can be controlled through SDN technologies (including OpenFlow) using the FSP Network Hypervisor [104].	[3], [103]
Arista Networks	7050, 7050X Switch Series	Switch	v1.0	Data center Ethernet or OpenFlow switches.	–	[11]
Brocade	ICX 6450	Switch	v1.3	Enterprise class stackable switch.	OpenFlow support gained through a mixed stack with Brocade ICX 6610 [33].	[33], [34]
Brocade	ICX 6610	Switch	v1.3	Enterprise class stackable switch.	–	[35], [36], [40]
Brocade	MLX Series	Router	v1.3	Service providers and enterprise class routers.	–	[37], [40]
Brocade	NetIron CER 2000 Series	Router	v1.3	High performance edge router.	–	[38], [40]

Table A.1: Continued.

Vendor	Product	Type	Latest OF Version	Description	Additional Information	Reference
Brocade	NetIron CES 2000 Series	Switch	v1.3	Edge class hybrid Ethernet/OpenFlow switches.	–	[39], [40]
Centec Networks	V330 Series	Switch	v1.3	Top of rack Ethernet/OpenFlow switches.	OpenFlow support is gained by using Open vSwitch [48].	[48]
Centec Networks	V350 Series	Switch	v1.3.1	Top of rack Ethernet/OpenFlow switches.	OpenFlow support is gained by using Open vSwitch [49].	[49]
Cyan	Z-Series Packet-Optical Platform	Switch	N/A	Family of packet-optical transport platforms.	Provides SDN capabilities when combined with Cyan Blue Planet SDN platform [61].	[62]
Datacom	DM4000 Series	Switch	v1.0	Metro standalone Ethernet/ OpenFlow switches.	–	[63]
Dell	S-Series (S4810, S4820T, S5000, S6000)	Switch	v1.3	High performance data center top of rack Ethernet/ OpenFlow switches.	OF v1.3 support is provided by FTOS 9.7 and is expected in early 2015 [304].	[69], [72]

Table A.1: Continued.

Vendor	Product	Type	Latest OF Version	Description	Additional Information	Reference
Dell	Z-Series	Switch	v1.3	Network core/aggregation switch.	OF v1.3 support is provided by FTOS 9.7 and is expected in early 2015 [304].	[71], [72]
Dell	MLX blade	Blade Switch	v1.3	High performance Ethernet/OpenFlow blade switch.	OF v1.3 support is provided by FTOS 9.7 and is expected in early 2015 [304].	[64], [72]
Dell	N2000 Series	Switch	v1.3	Layer 2 Ethernet/ OpenFlow switches.	OF v1.3 support is expected in early 2015 [304].	[65], [129]
Dell	N3000 and N4000 Series	Switch	v1.3	Layer 3 Ethernet/ OpenFlow switches.	OF v1.3 support is expected in early 2015 [304].	[66], [67], [68], [129]
Extreme Networks	Summit X430 Series	Switch	v1.0	Ethernet/OpenFlow standalone switches.	–	[90], [96], [97]
Extreme Networks	Summit X440 Series	Switch	v1.0	Ethernet/OpenFlow access switches.	–	[91], [96], [97], [295]
Extreme Networks	Summit X460 Series	Switch	v1.0	Edge class Ethernet/OpenFlow switches.	–	[92], [96], [97], [296]

Table A.1: Continued.

Vendor	Product	Type	Latest OF Version	Description	Additional Information	Reference
Extreme Networks	Summit X480 Series	Switch	v1.0	Data center class Ethernet/OpenFlow switches.	–	[93], [96], [97], [297]
Extreme Networks	Summit X670 Series	Switch	v1.0	Data center top of rack Ethernet/ OpenFlow switches.	–	[94], [96], [97], [298]
Extreme Networks	Summit X770 Series	Switch	v1.0	High performance data center top of rack Ethernet/ OpenFlow switches.	–	[88], [89], [96], [97]
Extreme Networks	E4G-200 and E4G-400	Router	v1.0	Cell site router.	–	[96], [97], [95], [299]
Extreme Networks	BlackDiamond 8000 Series	Chassis	v1.0	Core Ethernet/ OpenFlow chassis.	OF demo version support only [97].	[86], [96], [97]
Extreme Networks	BlackDiamond X8	Chassis	v1.0	Cloud-scale Ethernet/OpenFlow switches.	OF demo version support only [97].	[29], [87], [96], [97]

Table A.1: Continued.

Vendor	Product	Type	Latest OF Version	Description	Additional Information	Reference
Ezchip Technologies	Ezchip NP-4	Chip	v1.3	High performance 100-Gigabit network processors.	NoviFlow's switch software, NoviWare, supports OF v1.3 software for this chip [227].	[98], [115], [227]
Hewlett-Packard	FlexFabric 12900 Switch Series	Chassis	v1.3	Data center core switch chassis.	–	[146], [150]
Hewlett-Packard	12500 Switch Series	Chassis	v1.3	Enterprise data center core switch chassis.	–	[132], [150]
Hewlett-Packard	FlexFabric 11900 Switch Series	Chassis	v1.3	Data center aggregation switch chassis.	–	[145], [150]
Hewlett-Packard	10500 Switch Series	Chassis	v1.3	Enterprise network core chassis.	OpenFlow support for Comware v7 only [130].	[130], [131], [150]
Hewlett-Packard	8200 zl Switch Series	Chassis	v1.3	Data center class chassis.	–	[144], [150]
Hewlett-Packard	6600 Switch Series	Switch	v1.3	Data center edge Ethernet/ OpenFlow switches.	–	[143], [150]

Table A.1: Continued.

Vendor	Product	Type	Latest OF Version	Description	Additional Information	Reference
Hewlett-Packard	5900, 5920, FlexFabric 5930 Switch Series	Switch	v1.3	Data center top of rack Ethernet/ OpenFlow switches.	–	[141], [142], [148], [150]
Hewlett-Packard	FlexFabric 5700 Switch Series	Switch	v1.3	Data center top of rack Ethernet/ OpenFlow switches.	–	[147], [150]
Hewlett-Packard	5500 HI, 5500 EI Switch Series	Switch	v1.3	Edge class Ethernet/OpenFlow switches.	–	[139], [140], [150]
Hewlett-Packard	5400 zl, 5400R zl2 Switch Series	Chassis	v1.3	Network edge enterprise class chassis.	–	[137], [138], [150]
Hewlett-Packard	5130 EI Switch Series	Switch	v1.3	Ethernet/OpenFlow access switches.	–	[136], [150]
Hewlett-Packard	3800 Switch Series	Switch	v1.3	Enterprise class Ethernet/OpenFlow switches.	–	[135], [150]
Hewlett-Packard	3500 and 3500 yl Switch Series	Switch	v1.3	Edge class Ethernet/OpenFlow switches.	–	[134], [150]

Table A.1: Continued.

Vendor	Product	Type	Latest OF Version	Description	Additional Information	Reference
Hewlett-Packard	2920 Switch Series	Switch	v1.3	Edge class Ethernet/OpenFlow switches.	–	[133], [150]
Huawei	CloudEngine 5800, 6800, and 7800 Series	Switch	v1.3	Data center class Ethernet/OpenFlow switches.	Huawei's Open Programmability System (OPS) includes OpenFlow support [330].	[155], [157], [158], [159]
Huawei	CloudEngine 12800 Series	Chassis	v1.3	Data center core switch chassis.	Huawei's Open Programmability System (OPS) includes OpenFlow support [330].	[155], [156]
Huawei	S7700 and S9700 Series	Chassis	v1.3	High performance switching chassis.	OpenFlow support gained through a line card [161], [162].	[154], [161], [162]
Huawei	S12700 Series	Chassis	v1.3	Enterprise data center core switch chassis.	–	[154], [160]
IBM	RackSwitch G8264	Switch	v1.3.1	Data center switches supporting Visual Fabric and OpenFlow.	–	[163]

Table A.1: Continued.

Vendor	Product	Type	Latest OF Version	Description	Additional Information	Reference
Intel	FM5000 and FM6000 Switch Series	Switch	v1.0	High performance Ethernet/OpenFlow switches.	–	[167], [168], [169]
Juniper	EX4550 Ethernet Switches	Switch	v1.0	Data center top of rack Ethernet/ OpenFlow switches.	Using Junos OS version 13.2X51-D20 [244].	[176], [244]
Juniper	EX9200 Ethernet Switches	Chassis	v1.3.1	Core Ethernet/ OpenFlow chassis.	Using Junos OS version 14.2R1 [244].	[174], [177], [244]
Juniper	MX80 3D Universal Edge Router	Router	v1.3.1	Edge router for enterprise networks.	Using Junos OS version 14.2R1 [244].	[180], [244]
Juniper	MX240, MX480, MX960 3D Universal Edge Routers	Chassis	v1.3.1	High performance routing chassis.	Using Junos OS version 14.2R1 [244].	[178], [179], [181], [244]
Juniper	QFX5100 Switches	Switch	v1.3.1	High performance data center switches.	Using Junos OS version 14.1X53-D10 [244].	[175], [182], [244]
NEC	Programmable-Flow PF5240 and PF5248	Switch	v1.3.1	Enterprise class Ethernet/OpenFlow switches.	–	[218], [219]

Table A.1: Continued.

Vendor	Product	Type	Latest OF Version	Description	Additional Information	Reference
NEC	Programmable-Flow PF5820	Switch	v1.0	Enterprise class Ethernet/OpenFlow switch.	–	[220]
NEC	IP8800/S3640	Switch	v1.0	Layer 3 switch.	Not currently offered on the open market [300].	[170]
NetFPGA	1G, 10G, CML, SUME	Card	v1.0	PCI Express board with I/O capabilities.	–	[221], [222], [187], [216], [223], [224]
NoviFlow	NoviSwitch 1132 and 1248	Switch	v1.3	High performance OpenFlow switch.	–	[226]
Pica8	P-3297	Switch	v1.4	Ethernet/OpenFlow switches.	OpenFlow support is gained by using Open vSwitch [260].	[259]
Pica8	P-3922	Switch	v1.4	Ethernet/OpenFlow switches.	OpenFlow support is gained by using Open vSwitch [260].	[259]
Pica8	P-3930	Switch	v1.4	Ethernet/OpenFlow switches.	OpenFlow support is gained by using Open vSwitch [260].	[259]

Table A.1: Continued.

Vendor	Product	Type	Latest OF Version	Description	Additional Information	Reference
Pica8	P-5101	Switch	v1.4	Ethernet/OpenFlow switches.	OpenFlow support is gained by using Open vSwitch [260].	[259]
Pica8	P-5401	Switch	v1.4	Data center class Ethernet/OpenFlow switches.	OpenFlow support is gained by using Open vSwitch [260].	[259]
ZNYX Networks	B1 SDN	Switch	v1.3	Top of rack Ethernet/OpenFlow switches.	–	[15]