Software-Defined Networking: Challenges and research opportunities for Future Internet

Article *in* Computer Networks · December 2014

DOI: 10.1016/j.comnet.2014.10.015

CITATIONS
209

READS
11,014

5 authors, including:

Akram Hakiri
Institut Supérieur des Sciences Appliquées et de Technologie de Mateur
48 PUBLICATIONS **790** CITATIONS

SEE PROFILE

Aniruddha Gokhale
Vanderbilt University
371 PUBLICATIONS **6,206** CITATIONS

SEE PROFILE

Pascal Berthou
Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS)
132 PUBLICATIONS **1,096** CITATIONS

SEE PROFILE

Douglas Schmidt
Vanderbilt University
819 PUBLICATIONS **25,331** CITATIONS

SEE PROFILE

Software-defined Networking: Challenges and Research Opportunities for Future Internet

Akram Hakiri[a,b], Aniruddha Gokhale[c], Pascal Berthou[a,b], Douglas C. Schmidt[c], Gayraud Thierry[a,b]

[a] *CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France*
[b] *Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France*
[c] *Institute for Software Integrated Systems, Dept of EECS*
Vanderbilt University, Nashville, TN 37212, USA

Abstract

Currently many aspects of the classical architecture of the Internet are etched in stone – a so called ossification of the Internet – which has led to major obstacles in IPv6 deployment and difficulty in using IP multicast services. Yet, there exist many reasons to extend the Internet, e.g., for improving intra-domain and inter-domain routing for high availability of the network, providing end-to-end connectivity for users, and allowing dynamic QoS management of network resources for new applications, such as data center, cloud computing, and network virtualization. To address these requirements, the next-generation architecture for the Future Internet has introduced the concept of Software-Defined Networking (SDN). At the core of this emerging paradigm is the separation and centralization of the control plane from the forwarding elements in the network as opposed to the distributed control plane of existing networks. This decoupling allows deployment of control plane software components (e.g., OpenFlow controller) on computer platforms that are much more powerful than traditional network equipment (e.g., switches/routers) while protecting the data and intellectual property of the vendors of such equipment.

A critical understanding of this emerging paradigm is necessary to address the multiple challenges in realizing the Future Internet and to resolve the ossification problem of the existing Internet. To address these requirements, this paper surveys existing technologies and the wide range of recent and state-of-the-art projects on SDN followed by an in-depth discussion of the major challenges in this area.

Keywords: Future Internet, Software-Defined Networking (SDN), OpenFlow, Network Function Virtualization, Forwarding Plane, Control Plane.

1. Introduction

1.1. The Need for a New Network Architecture.

The capacity of the current Internet is rapidly becoming insufficient to cater to the large volumes of traffic patterns delivered by the new services and modalities (e.g., mobile devices and content, server virtualization, cloud services, big data), which is generated due to a large number of users, sensors and applications [1, 2].

Existing networks built with multiple tiers of static Ethernet switches arranged in a tree structure are ill-suited for the dynamic computing and storage needs of today's and future enterprise hyper-scale data centers, campuses, and carrier environments. Instead, new networking infrastructures are desired that will provide high performance, energy efficiency, and reliability. Moreover, they should improve the network speedup, scalability and robustness

Email addresses: hakiri@laas.fr (Akram Hakiri), a.gokhale@vanderbilt.edu (Aniruddha Gokhale), berthou@laas.fr (Pascal Berthou), d.schmidt@vanderbilt.edu (Douglas C. Schmidt), gayraud@laas.fr (Gayraud Thierry)

with the effective creation and delivery of versatile digital services that provide stringent quality of service (QoS) guarantees. Meeting these requirements is impossible with existing network equipment due to their limited capabilities. Additionally, today's protocols tend to be defined in isolation and are meant to solve a specific problem without the benefit of any fundamental abstractions.

In addition, to implement network-wide policies and to support any new services, managers today have to configure thousands of network devices and protocols, which makes it difficult to apply a consistent set of QoS, security, and other policies. Networks become vastly more complex with the addition of thousands of network devices that must be configured and managed. These devices have their control and forwarding logic parts both integrated in monolithic, closed, and mainframe-like boxes. Consequently, only a small number of external interfaces are standardized (e.g., packet forwarding) but all of their internal flexibility is hidden. The internals differ from vendor to vendor, with no open software platform to experiment with new ideas.

A lack of standard open interfaces limits the ability of network operators to tailor the networks to their individual environments and to improve either their hardware or software. Hence, there is a need for a new network equipment architecture that decouples the forwarding and control planes of the routers to dynamically associate forwarding elements and control elements.

1.2. Software-Defined Networking.

Software-Defined Networking (SDN) [3, 4] has emerged as the network architecture where the control plane logic is decoupled from the forwarding plane. SDN is a new approach for network programmability, which refers to the ability to control, change, and manage network behavior dynamically through software via open interfaces in contrast to relying on closed boxes and proprietary defined interfaces. The SDN framework enables centralized control of data path elements independently of the network technology used to connect these devices that can originate from different vendors. The centralized control embeds all the intelligence and maintains a network-wide view of the data path elements and links that connect them. This centralized up-to-date view makes the controller suitable to perform network management functions while allowing easy modifications to the networking functions through the centralized control plane.

Figure 1 depicts the SDN architecture illustrating the separation between the applications, control plane and data plane. Applications use the northbound API supported by the control plane to enforce their policies in the data plane without directly interacting with the data plane. The interface between the control and data plane is supported by southbound APIs, where a SDN controller will use these APIs to communicate with the network equipments in the data plane. These equipments are required to support the standardized APIs at this level.

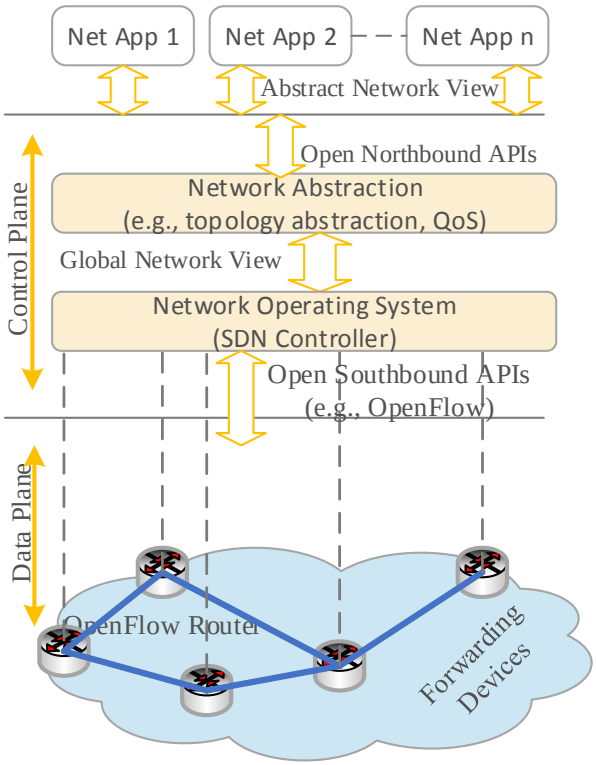

Figure 1: Simplified view of an SDN architecture

SDN makes it possible to manage the entire network through an intelligent orchestration and provisioning system that enables on-demand resource allocation, self-service provisioning, truly virtualized networking, and secure cloud services. Thus, the static network can evolve into an extensible vendor-independent service delivery

platform capable of responding rapidly to changing business, end-user, and market needs, which greatly simplifies the network design and operation. Consequently, the devices themselves no longer need to understand and process thousands of protocol standards but merely accept instructions from the SDN controllers.

A concrete realization of the SDN approach is OpenFlow (OF) [5, 6]. OpenFlow aims to allow providers to reengineer their traffic to test out new protocols in existing networks without disrupting production applications. The key elements of this technology consists of three parts: (i) flow tables installed in OpenFlow-aware switches, (ii) a controller installed in a remote host machine, and (iii) an OpenFlow protocol for the controller to talk securely with switches. The OpenFlow approach of splitting the control logic from the forwarding behavior provides a flexible capability for on-the-fly addition and update of several forwarding roles in the data plane.

1.3. Paper Objectives and Organization

The goal of this paper is to understand the challenges and research opportunities for SDN in defining the Future Internet. The rest of the paper is organized as follows: Section 2 outlines the key use cases of SDN and standardization efforts, which help to understand the realm of SDN and the spectrum of avenues for research and standardization; Sections 3 through 8 outline the key areas where more SDN research is needed to solve a number of unresolved challenges. Finally, Section 9 provides concluding remarks.

2. SDN Use Cases and Standardization Efforts

This section highlights the important use cases of SDN and current standardization activities. The goal of this section is to help the reader to situate their interests and research investigations in the context of ongoing standardization activities and use cases of the SDN technology.

2.1. SDN Use Cases in Service Provider Networks

The industry has recently undergone a significant transformation of their network infrastructure to support both current and future business models of SDN. This transformation offers a means to reduce the costs of service delivery and increase the service velocity using SDN technologies. In this section we present a number of SDN use cases.

2.1.1. Data Center Interconnects Use Case

Modern data centers are composed of tens of thousands of resources (e.g., processors, memory, storage, high-speed network interfaces), which in turn are packaged into racks and allocated as clusters consisting of thousands of hosts that are tightly connected with a high bandwidth network. These clusters are orchestrated to exploit thread-level parallelism to handle many Internet-based workloads. The traffic in the data center networks often exhibits bursty behavior where a large number of packets get injected in the network over a short time, which in turn induces a transient load imbalance that can affect other flows, and thereby significantly degrade the performance of the entire network [7].

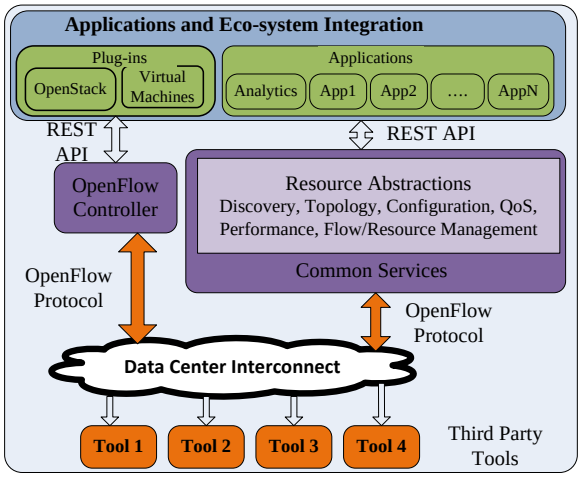

Figure 2: The use of OpenFlow for Cloud and Data-Center

The value of SDN in a data center interconnect lies specifically in its ability to provide network virtualization, abstraction and automation. In Figure 2, a centralized OpenFlow controller is used to provide dynamic traffic steering between applications that can be located in virtual machines or physical computers in data centers. Those applications use RESTful programming interfaces to expose their requirements to the underlying network over large-scale networks. The RESTful APIs en-

able them to perform a wide range of advanced network operations, such as topology discovery, QoS allocation, and load balancing. They also enable flexible network management with deterministic behavior by eliminating the need for resource over provisioning. Moreover, SDN-compliant third party tools can be easily plugged into data centers to perform faster recovery from link and node failures. That is, the OpenFlow controller reflects a unified view of the data center and simplifies the control of the entire network.

2.1.2. *Network Slicing Use Case*

Network slicing is a mechanism to divide the available network infrastructure into different partitions to allow multiple instances to coexist. Each slice controls its own packet forwarding without interfering with other slices even if they share the same underlying physical network. Figure 3 shows a multi tenant infrastructure connected through OpenFlow switches over a virtual overlay network to provide private or even public cloud services. This use case shows how slicing the network resources into multiple partitions provides tenants access to their virtual L2 slices without interfering with others. The SDN controller can achieve policy-based network management and flexible resource allocation, and at the same time can continue to support per-tenant/per-slice instantiation. This means that the controller can honor its rule to provide robustness, stability and scalability in terms of number of tenants, support for concurrent experiments and number of managed resources.

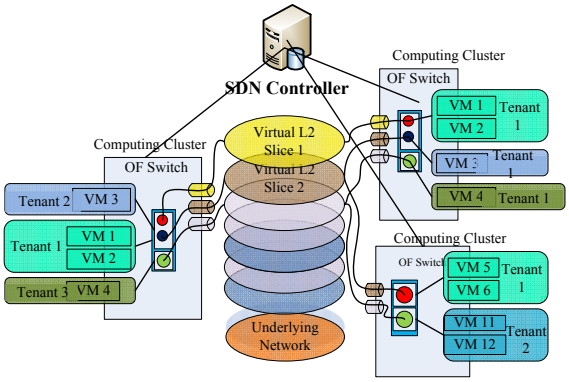

Figure 3: OpenFlow Network Slicing use case

2.1.3. *Wireless Settings Use Case*

Wireless networks require specific features like mobility management, dynamic channel configuration, and rapid client re-association. As depicted in Figure 4, the value of SDN in wireless networks lies specifically in its ability to provide new capabilities, such as network slicing, and the creation of new services on top of the virtualized resources in secure and isolated networks.

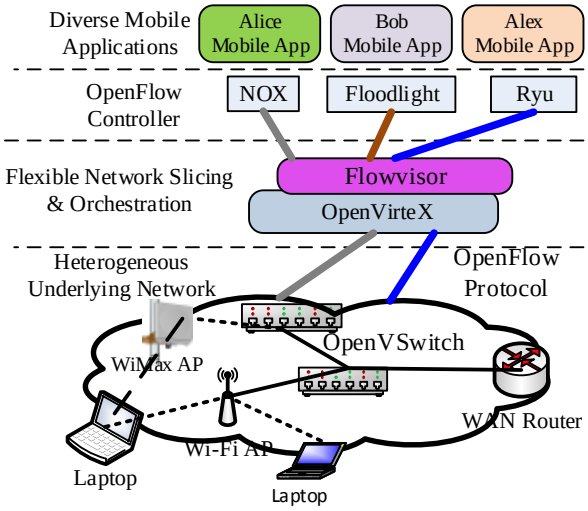

Figure 4: Wireless OpenFlow Infrastructure

In Figure 4 Flowvisor and OpenVirteX are OpenFlow-based proxy layers that allow creating slices based on multiple parameters such as bandwidth, flow-space (src/dst MAC, src/dst IP, and src/dst TCP ports), and CPU switch load. Each slice is independent, e.g., traffic from Alice's mobile application does not alter Bob's and Alex's traffic in the other slices. Moreover, wireless virtualization enables the migration of the switch configuration that manages Alice's application to another device seamlessly without disrupting the active network traffic of Bob and Alex. Such a configuration was introduced by the Odin framework [8], which is a programmable wireless data plane with modular and declarative programming interfaces across the wireless stack. This decoupling provides virtual access point abstractions to simplify network management for a wide range of enterprise requirements (e.g., an airport, a restaurant, public library) [9].

Additionally, one of the most interesting complemen-

tary technologies that can benefit from SDN are smart home gateways, which aim to interconnect residential home-gateway to their network providers. It also offers seamless and mobile connectivity without compromising security. For example, [10] introduced a Virtual Home-gateway Control (VHC) to improve seamless mobility, service delivery and home energy management.

2.1.4. Network Traffic Engineering Use Case

Traffic Engineering is a soft method for optimizing the performance of the ISP networks. It offers IP-based L2 or L3 enterprise VPN services and enables transmitting traffic to non IP networks like ATM and frame relay networks. Network providers use traffic engineering to support high transmission capacity and resilient communication by dynamically analyzing, predicting and regulating the behavior of the transmitted data. In traditional networks, some protocols such as OSPF and BGP allow the nodes to share their control information between their immediate neighbors and in a limited way to avoid network congestion. This means that there is no global view of the network available. If the users need to control or modify a particular path for a particular flow, the administrator has to test with parameters and priorities to achieve the expected behavior of the network. Each modification in the network policy requires individual configuration directly or remotely from each device.

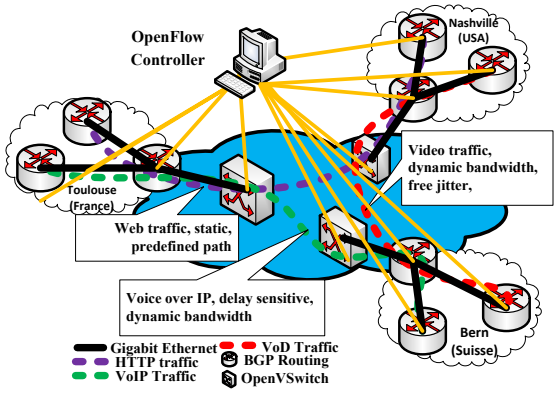

Figure 5: Traffic engineering with OpenFlow-based SDN

The added value of SDN is to provide flexible and programmable network devices to optimize and enable fine-grained control to different customers flow data with respect to the services they provide.

Figure 5 shows how a centralized SDN controller can manage three different traffic streams (HTTP traffic, VoIP, and VoD) over different underlying network technologies. For example, at the Toulouse access network, the user applications send their data through an OpenFlow-enabled router to the Bern network through the core network, which can be a circuit-based ATM technology. The centralized controller is able to control, manage and supervise the overall network equipment in the path between end-users. It performs traffic engineering with common control over packet and circuit networks using OpenFlow as a multi-layer Unified Control Plane (UCP). Thus, instead of using traditional distributed routing protocols like BGP, the centralized controller installs all routing decisions in the network equipment without using the traditional full-mesh of packet links[11], thereby enhancing the network flexibility and the scalability. Further, the controller can support application-specified routing as well as traffic aggregation based on IP source/destination addresses.

2.2. Standardization Activities around SDN

Driven by their attractive features and potential advantages, the development and deployment of SDN-based infrastructures have gained tremendous momentum in the industry and research community in recent years. In this section, we briefly discuss the standardization trends in SDNOpenFlow.

2.2.1. Clean Slate Design Initiative

The Internet has evolved to this day based on incremental changes to the protocols and by retrofitting the architecture to solve the current problems. However, it has not been possible to introduce any major changes to the deployed base of the Internet. The small and incremental changes that solve the current problems have introduced other problems so that the incremental approaches have arguably stretched the current design to its limit – a so called ossification of the Internet [12].

A new architectural design called the Clean Slate Design [13] considered how the Internet could be redesigned from a “clean slate” without being restrained by the accumulated complexity of the incremental approaches. The

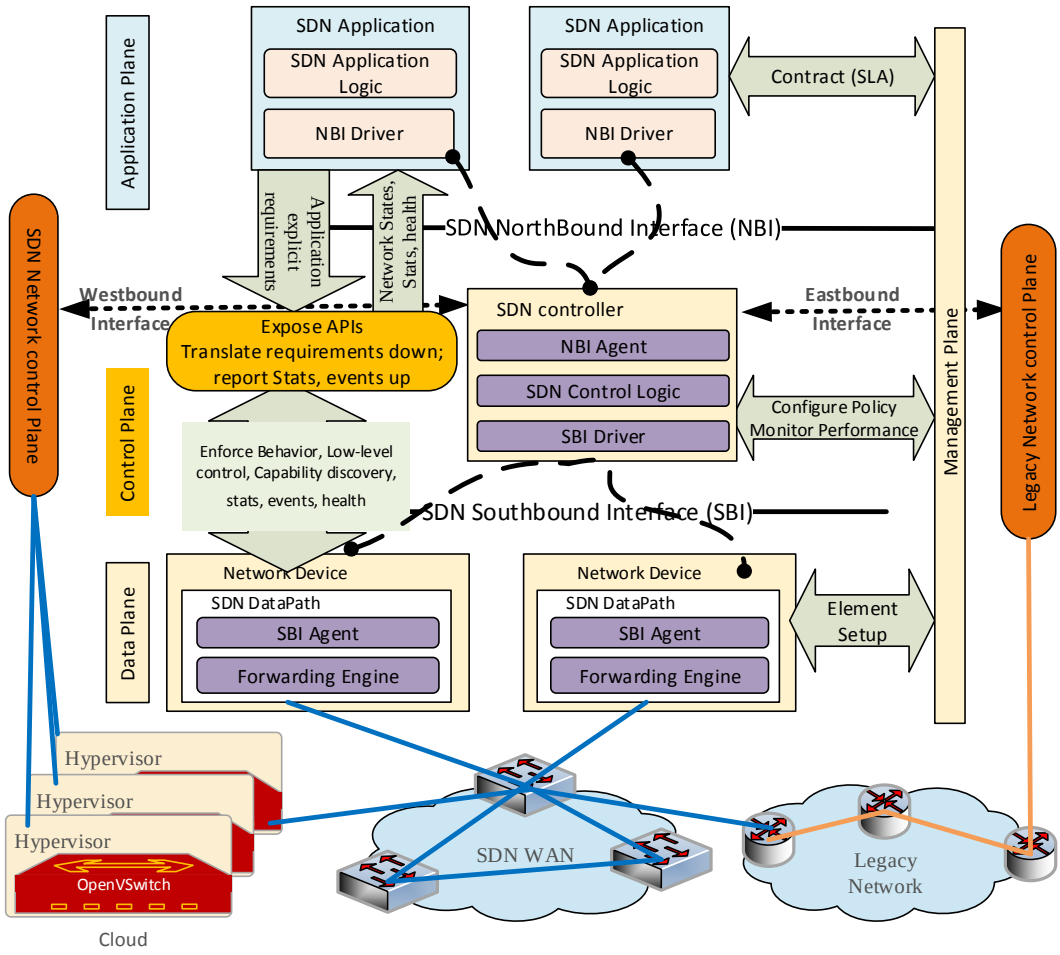

Figure 6: ONF Reference Architecture for SDN

research funding agencies all over the world are supporting the world-wide efforts to develop the next generation Internet [14].

The United States National Science Foundation (NSF) was among the first to announce the GENI (Global Environment for Networking Innovations) [15] program for developing and testing new Internet architectural designs as part of its FIND (Future Internet Design) program. This effort was followed by the FIRE (Future Internet Research and Experimentation) [16] program, the OFELIA (OpenFlow in Europe: Linking Infrastructure and Applications) program [17, 18], and the SPARC (Split Architecture Carrier Grade Networks) program [19] to support numerous next generation networking projects under the 7th Framework Program of the European Union, the AKARI Architecture Design Project [20] and the RISE (Research Infrastructure for large-Scale network Experiments) [21] in Japan.

2.2.2. *Open Networking Foundation for SDN*

The Open Networking Foundation [22] (ONF) was the first organization dedicated to the growth and success of SDN. Its mission was to evolve the OpenFlow protocol from its academic roots to a commercially viable

substrate for building networks and networking products. The ONF has formed working groups including carrier operators, software vendors, switch chip vendors, network equipment vendors, and system virtualization vendors to conduct the technical standardization tasks of SDN/OpenFlow. These groups continue to analyze SDN requirements, evolve the OpenFlow Standard to address the needs of commercial deployments, and research new standards to expand SDN benefits that cover the standardized protocols and technology points shown in Figure 6. The example scenario described in this Figure consist of interconnecting three different networks: a conventional IP network (legacy network), SDN-based core network and a SDN-enabled data center (cloud). The different SDN interfaces shown in the figure are as follows:

- ***Northbound Interface:*** It enables the exchange of data between the SDN controller and the application running on top of the network – a so called application-driven network. The kind of information that is exchanged, its form, and its frequency depends on each network application. There is no standardization for this interface.

- ***Southbound Interface:*** It refers to the APIs exposed to lower layers that allow the externalization of the SDN control plane to the data plane. OpenFlow and the Network Configuration Protocol (NetConf) are the southbound APIs used for a majority of SDN implementations to date.

- ***Eastbound Interface:*** It allows interconnecting conventional IP networks with SDN networks. Standardization of this interface is not provided and its implementation depends on the technology used by the non-SDN network. Usually, a translation module between SDN and legacy technology is required. For example, the SDN domain should be able to use legacy routing protocol to react to message requests (e.g., Path Computation Element protocol, (PCE), MPLS).

- ***Westbound Interface:*** They serve as the information conduit between multiple SDN control planes of different SDN domains. They help to achieve a global network view and influence routing decisions of each controller. They also allow seamless setup of network flow across heterogeneous SDN domains. Some conventional protocols like BGP can be used between remote SDN domains as westbound interface.

2.2.3. The Open Daylight Framework

The OpenDaylight Project [23] is a collaborative open source project hosted by The Linux Foundation. It aims to create platform-neutral, and open-source SDN technology. The OpenDaylight project provides a common controller infrastructure, protocol plug-ins, SDN applications, virtual overlay networks and standards-based northbound interfaces. It also provides support for a variety of southbound protocols, including OpenFlow, I2RS, Path Computation Element (PCE), Network Configuration (NetConf) and Application Layer Traffic Optimization (ALTO), as well as a Topology Manager module based on most of the active YANG data models for network topologies.

2.2.4. IETF and ITU-T Standardization Efforts for SDN

The IETF has recently begun to extend their specifications to support SDN principles [24]. In the IETF, the ForCES (Forwarding and Control Element Separation) project [25] defines a new architecture for network devices by specifying a framework [26] and a well-defined communication protocol (using a XML-based formal modeling language) to standardize the information exchange between the control and forwarding plane in a ForCES network element (NE) [27]. The ForCES NE follows a master-slave mode in which the forwarding elements (FE) are slaves and the control elements (CE) are masters.

ForCES physically separates the control and the forwarding plane by breaking the closed box of existing network equipment (i.e., routers) and replaces them with two separate network elements (FE and CE) each of which can connect to existing routers transparently, for example based on MPLS LSP (Label Switch Path) [28]. Despite ForCES being a mature standard solution with published RFCs and drafts [29, 30, 31, 32, 27, 33], it was not defined for SDN, and a limited number of vendor implementations exist. However, it can be used to design a new protocol in SDN [34].

Additionally, since many research and industrial SDN communities provided different SDN applications, con-

trollers and routers, it became hard to inter-operate them with standardized interfaces. Therefore, the IETF defined the Interface to the Routing System (I2RS) [35] with two major goals. The first is to standardize network-wide, multilayer topologies that include both virtual and real elements, network overlays and underlays. The second is to standardize the routing information base (RIB) programming of a device (virtual or real). Another goal of I2RS was to program Network Features Virtualization (NFV) service chains. The NFV aims to provide modern programmatic interfaces that offer fast, interactive access and can easily be manipulated by modern applications and programming methods.

Additionally, the Focus Group On Next Generation Networks (FGNGN) proposed the concept of "Softrouter" [36]. Softrouter advocates disaggregating the control and forwarding plane of a router. It separates the control plane processing functions (e.g., routing protocol processing) from the packet forwarding plane. These functions are implemented in outsourced dedicated servers that communicate with the forwarding elements via specific standard interfaces, i.e., Softrouter protocol.

2.2.5. Object Management Group efforts for SDN

The Object Management Group (OMG) is recently looking to provide its own specification to support northbound SDN ecosystem for middleware and related platform [37]. In the OMG, the Software Defined Networking (SDN) Working Group was established to investigate the opportunities for the development of open specification to support standard Information Model that represents the observable and controllable state of SDN network elements.

One of the issues being considered at the OMG on SDN is the use of the Data Distribution Service [38] (DDS) as a mechanism to monitor and configure SDN controllers. The OMG envisions DDS as a transport mechanism used to carry the OpenFlow commandsconfigurations as well as to observe the statestatistics of the SDN switches.

3. Challenges and Opportunities in the Evolution of SDN Architecture

SDN supports both centralized and distributed controller models. Each model has different infrastructure elements and requirements to consider. This section describes each SDN model along with a discussion about their advantages and drawbacks. Finally, we introduce the hybrid SDN models which combines the benefits of both approaches.

3.1. Pros and Cons of the Centralized SDN Model

The centralized SDN model is based on a single centralized controller that manages and supervises the entire network. This model is supported by the Open Networking Foundation (ONF). The network intelligence and states are logically centralized inside a single decision point. OpenFlow is the official protocol for use by the centralized controller to make global management and control operations.

Since only one centralized controller is used to program the entire network, it must have a global vision about the load on each switch across the routing path. It must also keep track of which flow inside which router presents a bottleneck on certain links between the remote SDN nodes. Additionally, the controller communicates with OpenFlow switches to collect statistics, errors and faults from each network device, and sends these data to the management plane. The latter is often a software composed of a database module and analytic algorithms that can detect the switch overloads and predict the future loads that may occur in the network.

Although the centralized control plane promises a single point of management and better control over the consistency of the network state, it incurs several key limitations. First, the controller needs to update OpenFlow switches more frequently than traditional routers. Thus, the topology discovery produces higher overload because all ports must be scanned linearly, which increases the response time and may impose a higher overload. For example, the controller may classify flows with different priorities into multiple classes, where each class requires a specific QoS setting that should be approved individually at setup time for every new flow received by OpenFlow switches [39]. Such an approach can incur substantial flexibility and robustness challenges for large-scale networks.

Second, the simplicity of the centralized model can come at the cost of control plane scalability. That is, grouping all the functionalities in a single node requires more computation power, data storage and throughput

to deliver the traffic causing its response time to be degraded. For example, as the size of data centers and cloud computing networks keeps increasing, neither the over-provisioning mechanisms nor load-balancing solutions can solve the scalability problems. Also, with respect to the hardware limitations, the switches may impose greater scalability bottlenecks and quickly hit real-life limits.

Third, in the centralized model, the first packet of every new flow that is introduced in the system must first be forwarded to a centralized SDN controller for inspection. The controller determines the future path for the flow hop-by-hp and programs the flow entries into every switch on the path including both the aggregation and core switches. Thus, when a new flow is to be programmed, the controller has to contact all the switches in the path, which is a scalability challenge for large networks and might result in an explosion in the number of forwarding states in the flow tables if fine-grained flow matching is required. The consequence is extra latency and the possibility of network failure as the number of new flows programmed increases. The centralized controller could also represent a single point of failure which makes the network highly vulnerable to disruptions and attacks. Also, the time required by the controller to setup all the properties for the flow will add to the latency. Any failure at any step may result in instability and convergence problems in the network.

Finally, SDN networks are becoming more complex and heterogeneous since they are designed to support versatile communication services and provide diverse functionalities such as security enforcement, firewall, network virtualization, and load balancing. These services need to coordinate their activities in the control plane to realize complicated control objectives and maintain a global vision of the entire network. However, it is hard to tightly coordinate the control actions and keep the consistency of network states among distributed functions. For example, inconsistent routing decisions may be generated from inconsistent topology information collected from routing protocols, which could create forwarding loops and broadcast storms in the network, and involve serious performance and correctness problems.

3.2. Pros and Cons of the Distributed SDN Model

The distributed SDN model aims to eliminate the single point of failure and enables scale up by sharing the load among several controllers. Distributed SDN control planes have been designed to be more responsive to handle local network events in data centers, where the controller instances share a huge amount of information to ensure fine-grained, network-wide consistency. In particular, for multi-domain SDNs with a large variety of network technologies ranging from high-capacity optical fiber to bandwidth-limited wireless links, the distributed SDN architecture is easily able to adapt to the users' and applications' requirements. Moreover, a distributed controller is more responsive, robust and can react faster and efficiently to global event handling.

The research efforts closely related to the distributed SDN model can be classified into three classes. The first class focuses on improving the performance of specific controllers like Maestro [40] and McNettle [41]. These controllers exploit switch-level parallelism to process flows from different switches concurrently. A second class of solutions proposes to distribute controllers. HyperFlow [42], Onix [43], and Devolved [44] controllers try to distribute the control plane between different network partitions while maintaining a logically centralized control using rendezvous synchronization points between locally selected events, distributed hash table to support controller clustering, or even distributed file system.

A third class of solutions proposes multi layered distributed controllers. Kandoo [45] distinguishes two-layers of hierarchical distributed controllers: (i) bottom-layer, a group of locally non-connected distributed controllers, i.e., each managing one or more switches without any knowledge of the network-wide state, and (ii) the top-layer, a logically centralized root controller that maintains the network-wide state. In addition, authors in [46] propose a cluster-based distributed model where a master controller is selected based on the load in the network so that if the load increases, the master node can be switched to a less loaded one. Likewise, authors in [47] introduce a SDN Controller Cluster (SCC) composed of multiple controller instances interconnected over East-West interfaces. Similarly, [48] describes a controller placement problem to decide the optimal number of controllers needed and their placement in a SDN network.

Despite the ability of those solutions to provide a distributed SDN model, several key challenges must be addressed in the future SDN to improve scalability and robustness of networks. First, the above approaches require a consistent network-wide view in all controllers. Also, the mapping between control planes and forwarding planes must be automated instead of the current static configuration which can result in uneven load distribution among the controllers. Second, these approaches cannot obtain a global optimal view of the entire network. Further, finding an optimal number of distributed controllers that ensure linear scale up of the SDN network when their number increases is hard. Finally, most of these approaches use local algorithms to develop coordination protocols in which each controller needs to respond only to events that take place in its local neighborhood. Thus, there remains the need to synchronize the overall local and distributed events to provide a global picture of the network.

3.3. Opportunities in Evolving towards a Hybrid SDN Control architecture

To address the limitations in each of the approaches described above, hybrid SDN architectures are being considered. However, a critical challenge stems from determining how much of network abstraction modules can be centralized and efficiently designed to support logically centralized control tasks, and at the same time provide physically distributed protocols. Consequently, to derive the advantages of both the centralized and the distributed architectures, a hybrid control plane is required to achieve such coordination. The hybrid SDN model leverages the benefits of the simple control of managing specific data flows as in the centralized model with the scalability and resilience of the distributed model.

It requires several key components to orchestrate the communication between SDN controllers. These orchestrators will require standard interfaces, mechanisms, and policies to manipulate and interact with the control planes in distributed environments, and support high-availability and fault-tolerance capabilities [49].

The hybrid SDN model may be useful in providing answers as to what state belongs in distributed protocols, what state must stay local in switches, and what state should be centralized. It could enhance network performance by enabling efficient resource usage because it will be possible to adjust more finely and automate each aspect of the network at the application level. Furthermore, the hybrid SDN model could provide management policies to solve state synchronization, security issues, and enable network optimization in cases of control plane overload. Also, hybrid SDN deployment model could allow truly non-disruptive migration. It allows upgrading existing infrastructure without the need to change the overall system.

4. Enhanced Programmability Needs for SDN-based Network Visibility and Management

With the increasing adoption of SDN, monitoring the activity between the controller and the switch to determine potential future impact, e.g., security vulnerabilities, is becoming more complex and error prone. This section describes key issues in SDN-based network visibility and management highlighting the challenges in enabling consistent SDN models, and provides promising directions to handle these challenges.

4.1. Network Visibility and Management Challenges with SDN

SDN introduces new management and monitoring capabilities that can improve performance and reduce the bottlenecks in the network. Although SDN monitoring tools can be powerful in small and medium sized network topologies, yet debugging, troubleshooting, monitoring and enforcing security compliance are very difficult tasks in distributed SDN. OpenFlow makes this even harder due to the poor choice of the monitoring location and the statistics that should be output for the OpenFlow SDN session. Diagnosing the network performance and bottlenecks without visibility into the traffic characteristics introduces new complexity with regard to the consistency of the SDN network.

SDN monitoring tools should be able to capture the network's information and behavior to help in carrying out management decisions. In particular, network configuration remains a difficult task in carrier-grade networks because administrators are required to grapple with low-level, vendor-specific interfaces to implement high-level functions. Ideally, network operators should be able to enforce various high-level policies, respond to a wide range of network events (e.g., intrusions). However, specifying high-level policies in terms of distributed, low-level

configurations remains difficult because SDN provides little to no mechanism for automatically responding to events that may occur. To enable these possibilities, service providers need programming interfaces to support an event-driven model to coordinate different asynchronous events associated with their networks.

Several tools for testing OpenFlow networks had been developed to provide monitoring functionality. For example, ENVI [50] and SAGE [51] can retrieve switch configurations, query link utilizations, or even modify a switch's flow table to alter routing. Moreover, they provide user interfaces to supervise the network bottlenecks in case of congestion, monitor flow status (e.g., bandwidth, delay, and loss) and computing/networking resources (e.g., network I/O, CPU, memory). Despite these capabilities, both efforts incur critical limitations in terms of their real-time performance.

4.2. Promising Directions in Network Visibility and Management for SDN

The SDN monitoring tools should provide a proactive capability that leverages the flexibility and programmability of SDN to create elastic network monitoring. They should provide high-level, declarative management languages to ensure the consistency of the network states and detect failures in real-time. These languages should implement a wide range of network policies for different management tasks (i.e., QoS, VLAN, network isolation, slicing, etc.) to prevent errors as they arise, independent of any specific controller's implementation. Thus, existing tools need to be extended to support service-awareness [52] to interactively map the flows onto services, identify selected flows from all lists of flows (i.e., flowspace), and monitor the status of the selected flows [53].

Moreover, the proactive capabilities of the monitoring tools should be able to classify the flows into different categories based on their behaviors and allocate different ports to each class based on multiple packet matching (e.g., IP, TCP port, VLAN, etc.) [54]. Additionally, they should provide individual functional modules to supervise the topology discovery either of the entire network or specific slices. Furthermore, a proactive SDN monitoring tool should be able to scale to wide area networks to deal with multiple controllers, alternate flow paths during troubleshooting, and monitor the flow's status such as bandwidth statistics [55].

The expected impact of SDN monitoring tools is to be able to scale to large-scale networks to deal with multiple controllers (i.e., in a distributed SDN model). They have to provide closed control loop functions dedicated to self-configuration, self-optimization, and self-healing. The control loop diagnostics and decision making processes need to be adapted automatically, e.g., by predicting the future actions based on the results of the previous ones. This proactive capability should leverage the flexibility and programmability of SDN, and improve its effectiveness and efficiency, owing to the cognitive processes that will enable creating more elastic network management either for the entire network or specific slices [56].

To further enhance the monitoring efficiency, Dynamic Measurement-Aware (DMR) Routing was introduced in [57]. The DMR selects flows based on their importance, i.e., big flows are split into several small sub-flows, and the most important flows are routed using a separate flow table entry. The sub-flows belonging to the same category can be inspected using Deep Packet Inspection (DPI) box [58]. The DPI can be used to learn and update flows dynamically and send them back to the monitoring tools.

In summary, implementing SDN traffic monitoring tools pose a series of challenges. First, there is substantial difficulty in creating network baselines with different taxonomies. For example, simplistic approaches try to describe traffic in terms of protocols and port numbers. Other approaches concentrate on bandwidth utilization and network topology to provide a global view of flows on the network. More advanced monitoring approaches try to classify traffic according to the flow's importance. Additional difficulties include finding the right tools which provide visualization at scale. Unfortunately, existing tools that are able to depict dozens or hundreds of nodes, face severe limitations when working with thousands or millions of nodes. They also face problems supporting multiple strategies, i.e., they suffer from topology location problem, cannot place instruments in enough locations, and are unable to visualize the network based on observed traffic patterns. Doing this in an automated way would prove extremely useful to network administrators and defenders. Finally, these tools must operate in an open SDN and vendor-independent environment, i.e., net-

working teams should be willing to consider deploying their tools and techniques on open platforms so they can devise and deploy their own network appliances.

5. Routing and Service Convergence using SDN

The value of SDN for service providers with dense and highly distributed networks lies specifically in the ability to provide them more options on locating resources thereby conferring a competitive advantage when delivering certain kinds of services. SDN may reduce the policy complexity by enabling scalability in inter-domain and providing validation for the claims of seamless evolution.

The major challenges for operators in the near future pertain to providing convergent, dynamic and adaptive networks in the context of a multi-services, multi-protocols, and multi-technology environment. Also, they are likely to face other key issues concerning the necessary network resources required to deliver differentiated and measurable quality of service (QoS). In this section we describe the key challenges and open issues in OpenFlow-based SDN for network operators.

5.1. SDN-based Routing Algorithms

The design of routing algorithms with OpenFlow can be performed either using native forwarding or tunneling mechanisms. With the native OpenFlow forwarding, the controller installs flow entries for specific header fields of the traffic. The selected fields should be unique within the entire network to allow the controller to perform conflict resolution and ensure the isolation of the traffic. For the tunneling mechanism, the controller should allow routing strategies by establishing a tunnel between the ingress and the egress nodes to perform packet encapsulation and decapsulation. The latter solution eliminates the need for conflict resolution and reduces the size of flow table entries in the switch. Such a functionality can be implemented either with MPLS-TE (i.e., MPLS Traffic Engineering) or Q-in-Q VLAN (i.e., 802.1Q). It also avoids the complex and error-prone process of per-packet verification in favor of creating a globally-consistent policy. Both mechanisms should provide the ability to add and remove an end-to-end path easily.

Using the approach outlined above, SDN should be able to deploy permanent and transient paths based on the instantiation of the OpenFlow rules. For the permanent path deployment, OpenFlow rules are injected as permanent entries in the flow table. However, this approach limits the scalability of the network because the rules will never be removed, which restricts the deployed flows to the number of available flow entries.

The transient path deployment allows the injection of the flow rules on demand as temporary entries. That is, when a switch cannot find a flow table entry that matches an incoming packet, the switch forwards the packet to the controller to dynamically compute the path "online" and determine the appropriate policy to be applied and injected. These operations increase the delay for flow setup but could improve scalability when combined with a suitable replacement policy for flow table entries.

5.2. SDN for Coordination between Routing protocols

The coordination between Interior Gateway Protocols (IGP), such as RIP2 [59], and Exterior Gateway Protocols (EGP), such as BGP [60], is one of the most important challenges in SDN networks. The limitation of the existing routing systems arises from their inability to coordinate their activities in an autonomous system (AS). For example, the violation of the SLA (Service Level Agreement) in the network may limit the automatic reaction of IGP protocols even if the link characteristics (e.g., link weight) allow the delivery of the packets because the scope of IGP protocols is limited to ingress routers inside a given AS. The violation of the SLA outside its AS is hidden and cannot be detected. Thus, the outgoing traffic may increase drastically in the network in the absence of load balancing and traffic management policies in the network.

Traditional approaches, such as the Routing Control Platform (RCP) [61], coordinate the inter-domain communication and route all the traffic on behalf of the BGP routers. Nevertheless, RCP incurs a couple of issues when choosing the best forwarding path: (i) it does not provide a clear separation between control and data planes; (ii) the enhancement of RCP based on the externalization of routing tables inside the hash tables compliant with OpenFlow routing tables [62] helped to improve the convergence time of finding the best path selection [63], but it does not provide a way to split traffic over multiple paths.

SDN can provide full coordination within intra-AS and inter-AS by simplifying the forwarding plane and let-

ting the SDN controller deal with the routing algorithms. Pushing routing strategies into separate controllers may enhance traffic management and load balancing, and absorb temporary picks in the network. Authors in [64] introduced the CONTRACT framework to dynamically adapt routing policies and improve SLA requirements. CONTRACT provides a set of algorithms for SDN routers to coordinate their routing actions. It evaluates routing decisions against the SLA and performs load balancing, traffic policing and filtering as necessary. However, in large-scale networks the interconnection of BGP routers should cope with forwarding loop management, signaling, and routing decisions. These tasks may increase the traffic congestion when packets traverse inter-domain multi-paths.

To cope with SDN congestion issues, we believe that routing with SDN should provide the controller with simpler and less error-prone policies that allow the best path selection to deliver SDN packets. For example, in this context the OpenFlow SDN wild-card rules must be revisited. These rules match on certain packet-header fields (e.g., MAC addresses, IP addresses, and TCP/UDP ports) with a timeout that triggers the switch to delete the rule after a fixed time interval (i.e., a hard timeout) or a specified period of inactivity (i.e., a soft timeout) to limit the convergence time of the routing algorithm.

5.3. Multicast Communication

Multicast communication is another important criterion for scalable communications. To demonstrate the feasibility of implementing multicast in OpenFlow switches, the authors in [65] implemented a prototype to demonstrate the feasibility of stateless multicast with Bloom filters. Even if the Bloom controller implements flexible multicast policies on the flow table entries, it does not provide any approach to manage multicast groups. Likewise, the authors in [66] extended a SDN controller with a Domain Title Service Agents (DTSA) to act as a logical bus that spans the applications and the switches. The DTSA establishes and maintains OpenFlow sessions between end applications by intercepting incoming messages from the controller to modify the flow tables accordingly. Similarly, authors in [67] extended OpenFlow to support multicast communication in a data center. They implemented a Layer 2 (MAC address) SDN controller that encapsulates MAC addresses into IP headers and creates virtual tunnels over overlay network between end points. However, tunneling mechanisms require that end-to-end path must be fixed a priori between source and destinations.

Future SDN networks should address this issue by encouraging further investigation in facilitating IP multicast. We believe that OpenFlow could provide a promising and flexible solution to solve the dynamic group member joining/leaving problem in IP multicast supported in overlay networks.

5.4. Network Convergence and QoS

With the need for flexibility and the efficient coexistence of multiple services (i.e., telephone, video and data) within a single switch, enterprises have to cope with a large demand for Quality of Service (QoS), Quality of Experience (QoE), robustness, ease of modification/upgrading, security and privacy [68, 69]. Providers must be able to create virtual network slices in the same network equipment while simultaneously assuring performance and isolation required across different services to enable application-driven QoS. However, SDN and its OpenFlow specification do not provide any support for differentiated QoS.

Recently, some researchers focused on providing QoS support for SDN-enabled applications without involving OpenFlow. For example, [70, 71, 72] introduced the OpenQoS framework to enable route optimization and per-flow granularity management. OpenQoS helps network administrators specify the QoS strategy that flows should follow, and how the controller can allocate the network resources and perform service differentiation. Likewise, authors in [39] proposed enhancing SDN controllers with QoS API to provide service differentiation and fine-grained automated QoS control in networks having multiple slices. Additionally, authors in [73] introduced the Port Control Protocol (PCP) to provide application-driven QoS. Applications explicitly signal their QoS requirements to the PCP, which uses the application's meta-data to implement the required QoS into the network equipment.

Although these different approaches aim to provide QoS mechanisms to support flexible and differentiated service management across different applications, none of them can automate QoS provisioning in real-time. Instead, a human in the loop, e.g., a network administrator, is required to specify the configuration of each service

before the communication begins. This strategy unfortunately loses the flexibility of network. We believe that the integration of session control protocols as a northbound interface is a promising approach to enable end-to-end QoS signaling among distributed SDN domains. This layer may provide proactive application-driven configuration of dynamic resource allocation with support for authentication and authorization [74]. From this perspective, the ability to predict the network behavior as a function of the network services to be delivered is of paramount importance for service providers. This way they can assess the impact of introducing new services, activating additional network features or enforcing a given set of (new) policies from both a financial and technical standpoints.

5.5. Shortest Path Forwarding using OpenFlow

Spanning Tree Protocols (STPs) were proposed for different controller platforms to ensure loop-free topologies and to prevent broadcast storms for Ethernet-based networks, such as data centers and cloud computing. The NOX Basic Spanning Tree module is an example of a controller with built-in STP. In contrast to their ability to prevent broadcast radiation, the existence of different STPs together in the same network may introduce several interoperability problems as they are not able to communicate with existing L2 forwarding protocols. In particular, the controllers lack control of the active topology and topology recovery after a link failure. Topology discovery is needed by the STP module to allow removing flows for subsequent frames after failure. Also, SDN controllers will suffer from suboptimal path calculation, interruption and long convergence every time topologies change [75]. Path re-calculation should be triggered thereby yielding a new active path.

5.6. Inter SDN Domain Communication

SDN is gradually being adopted by carrier-grade providers over heterogeneous, multi-technologies (e.g., WiFi, WiMax, UMTS/3G/4G, satellite, IP/Ethernet/ATM, etc.), and large-scale networks [76]. Each portion of these networks can be seen as independent isolated domains, typically controlled by a set of SDN controllers across multiple SDN domains, perhaps under the authority of a single operator or collaborating operators. Interconnecting isolated SDN domains involves some coordination to define who will be allowed to manage the entire network, install/update new program on the core infrastructure, what actions can each program execute, and how they can be performed within each tier of the network.

One way to match isolated SDN domains can be performed through novel SDN-specific protocols. For example, Inter-SDN domain protocol (SDNi) [77] can be seen as an interface between isolated SDN domains to coordinate the controller's behaviors. It should enable them to exchange control information related to topologies, events, energy consumption, and QoS requirements on each domain. Additionally, the SDNi protocol should be able to exchange reachability information and propagate the Service Level Agreement (SLA) end-to-end over heterogeneous networks.

However, SDNi still lacks a semantic network model to ensure the extensibility of its transport mechanisms and syntax. We suggest defining new types of message formats that may be used inside SDNi to ensure interoperability across multi-technologies, multi-domain SDNi networks. We believe SDNi can be implemented as an extension to BGP and SIP signaling to provide policy-based, vertical integration between an overlay-virtual-network technology (including SIP/SBC) and the data plane. SBCs (Session Border Controllers) may be implemented as northbound interface to pool the intelligence from the application/session layers within the network layer. The session management may be integrated with application-enabled SDN (A-SDN) provisioning mechanisms [73]. The A-SDN enables dynamic and automatic resource provisioning on network switches thereby allowing service providers to effectively deal with the surge in traffic without resorting to over provisioning of networks typically required in the past. Ideally, a fully automated service is required [78], from negotiation to ordering [79], through delivery assurance and charging should be supported.

6. SDN for Cloud-based Networks

The introduction and deployment of cloud-based services have emerged as an important solution that offers enterprises a cost-effective business model. However, many network functions have extreme characteristics and

performance requirements, which have created new challenges such as servers and network virtualization, mobile clouds, and security. These need to be addressed with intelligent network virtualization, high-speed packet processing, and load-balancing. SDN can be seen as a new and complementary technology to virtualization, which is poised to tackle the challenges of network-enabled cloud and web-scale deployments. In this section we describe the different challenges and opportunities in this space.

6.1. SDN Model for Information-Centric Networking

Information-Centric Networking (ICN) is receiving substantial attention in cloud networks because it provides users with data content that is based on naming the information instead of the communication channels between hosts. It also introduces new features such as in-network caching or content-based service differentiation. Yet, ICN still faces a number of challenges in realizing its full potential. In particular, a typical deployment of ICN schemes employs different transmission techniques and packet formats that are not interoperable. Also, deploying ICN-capable equipment in existing networks is hard, and requires replacing and/or updating existing operational network equipment with ICN-aware ones. Moreover, ICN schemes need to ensure the uniqueness of names in the network to handle lookups from large name spaces which can be greater than IP address spaces.

SDN offers a promising solution to facilitate the deployment of different ICN schemes without requiring redeployment of new ICN capable hardware. The authors in [80] propose a unified framework over virtualized networks to enable interoperability between different ICN architectures. Similarly, the CONET (COntent NETwork) framework [81] provides content-centric users access to remote named resources rather than to remote hosts. CONET extensions provide northbound interfaces to interconnect ICN nodes to the OpenFlow controller [82]. A unified packet format achieves end-to-end connectivity among different ICN schemes. Likewise, the SAIL (Scalable and Adaptive Internet Solutions) project [83] supports the notion of a flash network to enable dynamic resource provisioning on a time-scale comparable to existing compute and storage resources. Flash network slices are used to construct and connect scalable and distributed virtual services across data centers to end users across the globe.

Despite these solutions not requiring the replacement of existing network nodes, they incur several key limitations related to packet fragmentation introduced by their packet format. In particular, since supporting ICN over SDN requires ICN information in each packet, packet fragmentation remains a bottleneck that decreases the performance of the network. An approach that may resolve this issue is to let SDN controllers assign fixed-length labels to uniquely identify the different network paths and allow them to forward packets based on these path labels [84].

6.2. SDN for Data Centers and Cloud

The value of SDN in clouds and data centers lies specifically in its ability to provide new capabilities like network virtualization, automating resource provisioning, and creating new services on top of the provisioned network resources [85]. SDN promises an interactive solution to implement new capabilities, e.g., to enable cloud applications and services to retrieve network topology, monitor the underlying network conditions such as failures, and initiate and adjust network connectivity/tunneling. For instance, FlowN [86] virtualizes the address space of each application based on the fields in the packet headers and maps these virtual address spaces to physical addresses. The FlowN virtualized layer allows resource isolation and customized control logic for each tenant running its own controller.

Although data center interconnects allow bypassing the existing control plane protocols such as STP, yet interconnecting heterogeneous data centers to form federated clouds raises a challenging issue for SDN. Some researchers have extended the controller's northbound interfaces to develop customized real-time forwarding processes as cloud plug-ins, such as Neutron OpenStack, that enable virtualized address space and bandwidth isolation for each virtual link [87]. The northbound interfaces may be used by third party applications to monitor the SLA violation and adjust the network resources, if needed.

However, SDN-based virtualization incurs several key challenges, including the data center performance (e.g., coping with the increasing memory and processing overhead), slicing (e.g., network partitioning), resource provisioning (e.g., operators may over provision network links to overcome potential network congestion and packet drop within data centers, which makes it unpredictable

and costly in many networking scenarios), and QoS management (e.g., many SDN applications require northbound interfaces to communicate with third party applications, which requires monitoring the SLA violation to adjust the network resource if necessary). Since SDN is part of the network service evolution, we believe that a global solution should be provided to cope with these issues. Such an architecture may introduce a modulator SDN layer to orchestrate the communication between the applications, services and the data center infrastructure as shown by [85].

6.3. Network Function Virtualization

Network function virtualization provides a powerful way to run multiple concurrent virtual networks over a shared substrate. Cloud providers can offer virtual networks with a topology and a configuration customized to the users' needs. With a growing dependence on the network configuration, we argue that the ability to migrate the entire network would enable important new capabilities such as simplifying network resource allocation. Such an approach has motivated a novel business model called Networking-as-a-Service (NaaS) to enable customers to access virtualized network functions. For example, virtualized home gateways can be implemented as virtualized functions inside virtual machines in the cloud. Users can access the most recent versions without the need to upgrade the content by themselves.

As network functions are deployed in virtual machines, cloud operators may upgrade their infrastructure by adding new racks, installing new virtual machines or even deleting and migrating existing ones [88]. Network migration can also be applied to create green networks because virtual networks can be placed on different physical routers according to the traffic demand to reduce energy consumption. Moreover, factors such as server consolidation, load balancing, and security enforcement are driving most experiments with virtual network migration. For example, a link shared by different virtual networks could be jammed because of a DDoS attack to one of the virtual networks. In such a case, the non compromised networks could be migrated to different physical routers until the attack is resolved.

Often there are unintended consequences from virtual network migration that directly affect the physical underlying network. SDN controllers may stop an arbitrary number of applications during the migration operations, which could have significant performance degradation and high bandwidth costs to redirect traffic to other virtual network slices. Furthermore, the network may have a large amount of state collected from a set of distributed SDN switches that should be kept consistent to avoid outages, transient loops, and violation of SLAs during the migration process. Meanwhile, controllers should maintain connectivity and forward path re-routing without interruption. As such, the centralized SDN model is a single point of failure and not suitable to guarantee the consistency of the network states. It does not reduce the downtime and packet loss during the migration of network functions between virtual machines [89].

Migrating network functions requires a real-time scheduling model that can prevent failures caused by congestion and bandwidth violation. Such a model is perceived to be NP-hard and not guaranteed all the time [90]. In resolving these issues, there is a need to rethink the mapping of virtual slice requirements throughout a communication matrix onto the physical infrastructure as introduced by CloudNaaS controller framework [91]. The controller uses a placement optimizer to choose the best virtual network placement and dynamic provisioning model to reallocate resources based on their availability. We believe that a self-service provisioning model provided by the cloud can provide a rich set of network services such as seamless network isolation, customized network addressing as well as service differentiation.

Another possible direction to enhance virtual network migration can be achieved by synchronizing network states among distributed switches and create overlay tunnels to relay traffic between network slices [92]. However, this approach requires an additional proxy layer to create synchronization points for redundancy and state duplication. We believe that more advanced algorithms for scheduling virtual network migration need to be explored to leverage technologies like redundancy elimination, and reduce the overhead of copying the state for multiple slices. The NetGraph [93] framework is an example that supports incremental algorithms with practical computation time and memory requirements.

7. SDN in Wireless and Mobility Settings

Software Defined Wireless Networking (SDWN) is a SDN technology for wireless that provides radio resource and mobility management, routing, and multi homing. SDWN can provide a programmable wireless data plane to allow modular and declarative programming interfaces across the wireless stack. It also enables refactoring wireless protocols into processing and decision planes [9] as introduced by the OpenWRT [94] and Indigo [95] firmware. This decoupling allows programming the enterprise-specific requirements (e.g., an airport, a restaurant, public library) in a wireless access point (AP). This section describes the key challenges in SDWN.

7.1. Mobility and Multi homing Management with SDN

By 2020 it is predicted that there will be thousand times more connected mobile devices than today with different QoS requirements, which will interconnect to all kinds of heterogeneous and customized Internet-based services and applications. Accordingly, these developments demand a rethinking of the network design, which in turn requires us to understand the implications of using SDN in the most common wireless networking scenarios. It is also important to understand the key challenges that exist in this realm and how they can be addressed.

IP mobility support has been specified in traditional IPv4 and IPv6 networks, but presently there is not much discussion regarding mobility support in SDN. Thus, SDWN should provide radio resource management, mobility management and routing [96]. It should include novel mobility management mechanisms to maintain session continuity from the application's perspective. In particular, handoff management is a challenging issue in supporting SDN mobility because mobile terminals move rapidly between virtualized APs. Thus, SDWN should provide network connectivity through dynamic channel configuration.

The authors in [10] introduced the concept of slicing the wireless infrastructure into logical virtualized partitions, each managed by virtual APs. Similarly, the effort described in [8] introduced the Odin framework as a mobile agent that communicates with SDN controllers to keep a global view of flows in the network. It delegates the handoff management to a virtual AP that invokes the physical AP to manage the mobility and maintain host reachability with respect to the receiver signal. Although these approaches help to simplify the mobility management, SDN mobility functions must be enhanced to provide rapid client re-association, load balancing, and policy management such as charging, QoS, authentication, and authorization.

Another key challenge in wireless/broadband networks concerns multi homing[97]. Multi homing implies the attachment of an end-host to multiple networks at the same time so users can freely move between wireless infrastructures. This approach can be realized by applying SDN capabilities to the relay between the home network and edge networks. For example, within home networks, a virtualized residential gateway can improve service delivery between the core home network and the network-enabled devices [98]. The target architecture (i.e., virtualized residential gateway) is realized by applying SDN and NFV between the home gateway and the access network, and moving most of the gateway functionality to a virtualized execution environment. Since the virtualized residential gateway is cloud-based, it forms the core of a user's home network by connecting their network-enabled devices while benefiting from broadband Internet services such as connection sharing, Firewall security, VPN connectivity, IP telephony, audio/video streaming, Wireless LAN connectivity, etc.

Another important key challenge in wireless SDN is related to routing packets in Wireless Mesh Networks (WMNs). A WMN is a multihop wireless ad-hoc network in which stationary wireless mesh routers relay traffic on behalf of other mesh routers or client stations to form a wireless backbone [99]. The primary advantages of wireless ad-hoc networks lie in their ability to provide high fault tolerant communication even when a large number of nodes fail, simplicity of configuring wireless nodes, and their capacity to provide broadband connectivity. These networks require dynamic routing algorithms to cope with the limitations of wireless channels such as fading, interference, and broadcast [100].

SDN can partially solve these challenges by enabling programmability of the network [101]. For example, the authors in [102] introduced SDN-based CloudMAC architecture to enhance the reconfigurability and the flexibility of wireless ad-hoc networks. CloudMAC helps to improve the wireless connectivity through seamless AP switching, which provides fast packet processing and re-

duced packet loss. However, several key issues of WMNs remain unresolved [103]. In particular, the topology of ad-hoc wireless networks changes at a much higher pace than in wired networks due the variation in link quality and node movement. Additionally, wireless nodes may join and leave the network at any time so any SDN-based routing protocol must provide autonomous topology discovery as well as adaptive neighbor discovery to react swiftly to changes in the network [104].

7.2. SDN for Mobile Cloud

Mobile cloud computing is one of the technologies that are converging into a rapidly growing field of mobile and wireless network. It provides an excellent backend for applications on mobile devices giving access to resources such as storage and computing power, which are limited in the mobile device itself. The close interaction with the cloud may create an environment in which mobile devices appear attached locally to the cloud with low latency [105]. Both SDN and NFV can extend wireless access infrastructure to cloud computing and enable logical slicing of resources to multiple devices. SDN promises an attractive solution to implement new capabilities to cloud applications and services. It can help to retrieve network topology, monitor the underlying network conditions (e.g., failures), initiate and adjust network connectivity and support tunneling.

Additionally, given the dynamic needs and supply of the network resources due to the rich set of resources available in the cloud, mobile users can benefit from resource virtualization to accommodate different requirements of their mobile terminals moving within the mobile cloud. Weaving different wireless access technologies together in a fluid fashion and creating smart gateways in the cloud in a transparent manner is important to realize the full potential of mobile clouds. To this end, virtualized access networks should support fine-grained isolation per person or per application for each device in the cloud.

Despite SDN having some advantages, such as resource sharing and session management, it incurs several limitations in the context of mobile clouds. In particular, since mobile users can repeatedly trigger the embedded controller for marshalling and unmarshalling of flow rules (e.g., in OpenFlow messages), the overhead increases more significantly because of the limited computing capabilities and resources of mobile devices (i.e., extra memory consumption and extra latency). For example, mobile interactive applications (e.g., mobile gaming, virtual visits) require reliable connectivity to the cloud as well as low latency and impose higher bandwidth requirements from wireless access networks to cloud service.

Furthermore, mobile users move frequently between multiple access networks, using either the same or different mobile terminals, but often with a unique persona. In general, it is difficult to realize a Mobile Personal Grid (MPG), which requires maintaining seamless connectivity of a set of devices that belong to a particular individual to a remote public and private cloud. Maintaining seamless wireless connectivity for the MPG introduces multidimensional challenges including the need to deal with dynamic mobility management across heterogenous networks, power saving, resource availability, fluctuating operating conditions, and limitations on the movement of content across multiple devices and the cloud [105].

Addressing these limitations simultaneously may increase device complexity, degrade the network performance, and cause connectivity problems. The key challenge for mobile clouds lies in transforming physical access networks to multiple virtual and isolated networks while maintaining and managing seamless connectivity. Virtualized switch functions such as those provided by OpenVSwitch may enable controllers to connect to a specific virtual partition with autonomic reconfiguration and lossless connectivity. The virtualization of mobile protocol stacks could abstract mobile resources to accommodate their different requirements. We argue that a close interaction with the cloud may help achieve the multidimensional mobility requirements [105] and relieve the network infrastructure from complex network tasks (e.g., configuration, traffic analysis) to the cloud [106].

7.3. SDN for WPAN

Since SDN promises to reduce the complexity of the configuration and management of networks, its functionality can also be applied to many wireless infrastructure networks such as the Low-Rate Wireless Personal Area Networks (LR-WPANs). Extending SDN to LR-WPAN was considered impractical because these networks are highly constrained. LR-WPAN requires numerous low-cost nodes communicating over multiple hops to cover a large geographical area. These nodes require duty cycles to provide low energy consumption to operate for multi

year lifetimes on batteries with modest lifetimes. They also require small software footprint due their limited amount of memory storage and CPU processing speeds.

Additionally, to enable SDN for LR-WPANs, several challenges must be resolved to provide cross-layer optimization [107] as well as data aggregation. We argue that future SDN controllers should provide an appropriate module to define the rules that consider specific matching fields for LR-WPAN environment [108]. These rules may route packets based on their specific values included within the payload they carry. For example, packets may include a specific value of temperature sensors that exceed a given threshold. SDN controllers will need to configure multipath routing algorithms to route LR-WPAN packets based on multilevel thresholds. Moreover, controllers must address several key problems related to node mobility, topology discovery, self-configuration as well as self-organization [109]. They should also deal with link unreliability, and robustness to the failure of generic nodes and the control node [110].

7.4. SDN for Cellular networks

The growing demand and the diverse patterns of mobile traffic place an increasing strain on cellular networks. The best way to increase per-user capacity is to make cells small and bring the base station closer to the mobile client. SDN may provide rapid response to mobile terminals and avoid disruptions in the service across different technologies. However, supporting an increasing number of subscribers with frequent changes in user location incurs serious issues [111]. Cellular network infrastructures must redirect user's data to load balancing servers to accommodate changing network conditions and handle traffic in real-time with specific QoS priority [8]. Presently, cellular systems have a single direct link between the base station and the terminal. However, multihop networks require maintaining multilink between multiple transmitter and receiver to form multipath communication – the so called multihop cooperative network. Compared to existing technology, which include mechanisms for retransmission and multiple acknowledgments, multihop cooperative networks can overcome these limitations by providing high density access networks. However, they often suffer throughput penalties since they operate in a half duplex mode and therefore introduce insufficiency of spectrum usage.

To increase the capacity of cellular systems, SDN can provide solutions to overcome the limitations of multihop wireless networks. Cellular SDN networks (or briefly, CellSDN) [112] could maintain a Subscriber Information Base (SIB) to translate a subscriber's attributes into the switch rules to set up and reconfigure services flexibly. To this end, finer grained control of traffic in CellSDN must be adapted to cellular infrastructures to handle notifications from the users' terminals. The authors in [113] introduce a Deep Packet Inspection (DPI) engine to enable finer grained classification at the application layer. Section 4 described the use of DPI in monitoring tools, whereas here it is used for routing flows across multichannel cellular networks. Application level control could then provide flexible and fine-grained control of the users data and analyze packet contents to identify malicious traffic. Despite these possibilities, CellSDN requires specific SDN protocols to orchestrate the remote virtualized resources [114]. Moreover, these protocols must enable network slicing, traffic engineering, minimizing delays and reduction in packet loss [115].

In cellular communications, an architecture based on SDN techniques may give operators greater freedom to balance operational parameters, such as network resilience, service performance and QoE. CellSDN controllers may implement Radio Resource Management (RRM) protocol [116] as northbound interfaces to simplify the QoS provisioning. Similarly, CellSDN routers could support techniques like header compression and decompression to reduce the overhead with small packet payloads on low bandwidth links. Another important challenge in CellSDN is designing the architecture of the network. Traditionally, CellSDN was designed with centralized control and data structures in mind to improve its performance. Centralized SDN model in CellSDN results in a single point of failure. Thus, when the cellular infrastructure fails, the entire network will be unavailable. Hence, a rethinking of architecting CellSDN using a distributed SDN model is needed.

Distributed CellSDN could provide high-performance, cost-effective and distributed mobility management [117] in CellSDN. We believe that a distributed SDN model could also provide multiple parallel virtualized transmission channels to support the increasing number of mobile terminals requesting the network resources [114]. As for fixed network virtualization, there is a need to reconsider

the issue of isolating traffic into multiple wireless slices, where each slice could support a specific traffic pattern. Such an approach may help to create virtual base stations and orchestrate the available resources among different mobile devices, thereby saving power and memory usage.

8. Challenges and Opportunities for Security in SDN

Security in SDN networks poses significant challenges because its programmable aspect presents a complex set of problems to cope with [118]. It is expected that the increasing number of DDoS and malware attacks, spam, and phishing activities will change the dynamics around securing SDN infrastructures. In this context, mobile wireless networks are more vulnerable than fixed wired networks since broadcast wireless channels easily allow message eavesdropping and injection (i.e., vulnerability of channels). Moreover, mobile ad-hoc networks incur more complex security challenges due to their lack of infrastructure (e.g., security servers), which renders classical security solutions infeasible. Traditional security approaches require downtime to orchestrate topology changes while the network is reconfigured, new security configuration is inserted, and multiple security services are turned on and debugged.

8.1. Security Challenges in SDN

Security is minimally specified in SDN; thus the OpenFlow specification does not describe the certificate format to ensure data integrity. SDN security will require more sophisticated encryption and authentication mechanisms to prevent hackers and to recover packets from failure. For example, multiple controllers may mutually authenticate each other by exchanging certificates signed by third party private keys. The difference between these keys, however, raises unexpected security vulnerabilities in the entire network. Thus, the controllers may be unable to interoperate with each other since they have different keys. The OpenFlow specification recommends using a plain TCP connection for the secure channel but gives no indication about what sort of alternatives can be used to that end. Optionally, TLS sessions can be used to provide encrypted and secure channels in both directions to prevent eavesdropping but without details about the interoperable version that could be used. The specification makes no mention about how a secure connection can be established nor how the certificate can be checked between the controllers and the switches. Security issues also stem from the diversity in network configurations. The security mechanisms defined by the specification are ill-suited to protect the network against eavesdropping and controller impersonation.

In developing solutions to address the myriad of security challenges in SDNs, we need to cope with one primary challenging issue: managing the trade-offs between the network security and performance.

8.2. Opportunities for Addressing Security Concerns in SDNs

A promising approach to address the different security problems in SDN involves developing security policies with a unified syntax for the OpenFlow protocol. These policies should enable authentication, authorization, access control, and secure transport between applications and SDN controllers, or even between multiple controllers and a set of switches. These policies can be defined via the *Security Assertion Markup Language* (SAML) to exchange public/private keys as part of the OpenFlow policies, *e.g.*, within confidential OpenFlow messages that may be transported using the *Datagram Transport Layer Security (DTLS)* protocol.

Supporting intrusion detection is a crucial part of a secure SDN framework. A promising approach is to provide an Intrusion Prevention System (IPS) based on the open source SNORT project [119]. In particular, OpenFlow security algorithms should provide role-based authentication to check for contradictions in flow rules when applications attempt to insert disallowed flow rules. It may be possible to add customized security services into the network, either on a per-flow basis or per-group of flow [120]. Another approach to resolving the mobile SDN security challenges could consider separating these functions or outsource them to a third party server, as described in the OpenWiFi project [121].

Yet another promising approach involves inserting different security policies in the OpenFlow protocol using L4 to L7 services, such as URL filtering between an endpoint, rejecting flows toward a specific destination, redirecting HTTP and HTTPS traffic using a proxy, firewall, etc. Extending the OpenFlow matching rules to incorporate new security rules using different fields, such as

wild-cards, physical/virtual ports and IP addresses is another possibility. However, exposing IP addresses to network attacks may hand the adversaries significant advantage to remotely scan networks and identify their targets accurately and quickly. The authors in [122] introduced a technique called OpenFlow Random Host Mutation (OF-RHM) to hide real IP addresses from external attacks. They assign virtual IP addresses to hosts so that they can be reached only by authorized entities.

9. Conclusions

Most researchers argue that the current Internet architecture is inadequate and has reached a tipping point where most of the time and effort is spent addressing existing flaws rather than developing new ideas. To address the challenge of ossification of the Internet, researchers have started to focus on redesigning the overall architecture by breaking the tight integration of the forwarding and control plane implementations. As a result, major research projects focus on the SDN architecture to construct and present a logically centralized map of the network. The next-generation of SDN networks will benefit not only from the simplicity of the implementation, but also from the fact that maintaining and updating applications will be easier as well.

In this paper, we have surveyed a wide range of recent and state-of-the-art projects in SDN. Based on the survey, we classified key challenges and opportunities along a number of different areas including architectural models, programmability, convergence, wireless and mobility, cloud platforms, and security. We showed that the networking community is heavily involved in SDN research, however, most of the investigations continue to focus on topics such as control plane/data plane, distributed vs centralized control plane, scalability of solutions, Hybrid solutions, call graph, networking models etc.

We argue that while these research efforts are important, they need to occur in the context of overall network programmability and scalability goals. Although OpenFlow, which is a prominent SDN implementation, promises a flexible, open, and dynamic flow transmission mechanism, it also poses a set of challenges in terms of network virtualization, mobility management, operation, which will require coordinated attention from the research community for its success and wide acceptance. Our goal was to present these challenges and present current standardization efforts.

By no means have we presented an exhaustive list of opportunities. We believe additional challenges and opportunities for research exists along a broad spectrum ranging from SDN solutions used in converged packet and circuit switched networks to formal modeling and model checking to improve the trustworthiness of the SDN-based solutions.

References

[1] A. Metzger, C. C. Marquezan, Future internet apps: The next wave of adaptive service-oriented systems, in: ServiceWave, 2011, pp. 230–241.

[2] C. C. Marquezan, A. Metzger, K. Pohl, V. Engen, M. Boniface, S. Phillips, Z. Zlatev, Adaptive Web Services for Modular and Reusable Software Development: Tactics and Solution, IGI, 2012, Ch. Adaptive Future Internet Applications: Opportunities and Challenges for Adaptive Web Services Technology.

[3] N. McKeown, Software-defined networking, INFOCOM keynote talk, Apr.

[4] H. Kim, N. Feamster, Improving network management with software defined networking, Communications Magazine, IEEE 51 (2) (2013) 114–119.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, Openflow: enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review (2008) 69–74.

[6] ONF, The openflow 1.3.1 specification, Tech. rep.

[7] C. Lam, H. Liu, B. Koley, X. Zhao, V. Kamalov, V. Gill, Fiber optic communication technologies: What's needed for datacenter network operations, IEEE Communications Magazine.

[8] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, T. Vazao, Towards programmable enterprise wlans with odin, HotSDN '12, 2012, pp. 115–120.

[9] M. Bansal, J. Mehlman, S. Katti, P. Levis, Openradio: a programmable wireless dataplane, HotSDN '12, 2012, pp. 109–114.

[10] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, N. McKeown, Openroads: empowering research in mobile networks, SIGCOMM Comput. Commun. Rev. 40 (1) (2010) 125–126.

[11] J. Vasseur, J. Leroux, S. Yasukawa, S. Previdi, P. Psenak, P. Mabbey, Routing Extensions for Discovery of Multiprotocol (MPLS) Label Switch Router (LSR) Traffic Engineering (TE) Mesh Membership (2007).

[12] J. Turner, D. Taylor, Diversifying the internet, in: Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE, Vol. 2, 2005.

[13] A. Feldmann, Internet clean-slate design: What and why?, SIGCOMM Comput. Commun. Rev. 37 (3) (2007) 59–64.

[14] S. Paul, J. Pan, R. Jain, Architectures for the future networks and the next generation internet: A survey, Comput. Commun. 34 (1) (2011) 2–42.

[15] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, I. Seskar, Geni: A federated testbed for innovative network experiments, Computer Networks 61 (0) (2014) 5 – 23.
URL http://www.geni.net/

[16] [link].
URL http://www.ict-fire.eu/home.html

[17] P. Frosi, L. J. Camargos, R. Pasquini, M. S. Soares, L. F. Faina, D. M. F. O. Silva, P. R. S. L. Coelho, V. B. Bernal, S. T. Kofuji., Experimenting domain title service to meet mobility and multicast aggregation by using openflow (Sep. 2011).

[18] M. Sun, 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, S. Sharma, Design and implementation of the OFELIA FP7 facility: The european openflow testbed, Computer Networks 61 (0) (2014) 132 – 150.
URL http://www.fp7-ofelia.eu/

[19] M. Kind, F. Westphal, A. Gladisch, S. Topp, Splitarchitecture: Applying the software defined networking concept to carrier networks, in: World Telecommunications Congress (WTC), 2012, 2012, pp. 1–6.
URL http://www.fp7-sparc.eu/

[20] H. Harai, Akari architecture design for new generation network, in: Summer Topical Meeting, 2009. LEOSST '09. IEEE/LEOS, 2009, pp. 155–156.

[21] Y. Kanaumi, S. Saito, E. Kawai, S. Ishii, K. Kobayashi, S. Shimojo, Rise: A wide-area hybrid openflow network testbed., IEICE Transactions 96-B (1) (2013) 108–118.

[22] O. N. Foundation, Onf overview (2013).
URL https://www.opennetworking.org/about/onf-overview

[23] L. Foundation, Opendaylight: An open source community and meritocracy for software-defined networking, A Linux Foundation Collaborative Project (April 2013).
URL http://www.opendaylight.org/resources/publications

[24] E. Haleplidis, S. Denazis, K. Pentikousis, J. H. Salim, O. Koufopavlou, Sdn layers and architecture terminology, Tech. Rep. 01 (2013).

[25] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, J. Halpern, Forwarding and control element separation (forces) protocol specification (2010).

[26] L. Yang, R. Dantu, T. Anderson, R. Gopal, Forwarding and control element separation (forces) framework (2004).

[27] L. Dong, A. Doria, R. Gopal, R. HAAS, J. H. Salim, H. Khosravi, W. Wang, Forces protocol specification, Tech. Rep. draft-ietf-forces-protocol-17 (2008).

[28] I. Kovacevic, Forces protocol as a solution for interaction of control and forwarding planes in distributed routers, 17th Telecommunications forum TELFOR (2009) 529–532.

[29] R. Haas, Forwarding and control element separation (forces) mib (2010).

[30] A. Crouch, H. Khosravi, A. Doria, X. Wang, K. Ogawa, Forwarding and control element separation (forces) applicability statement (2010).

[31] E. Haleplidis, O. Koufopavlou, S. Denazis, Forwarding and control element separation (forces) implementation experience (2011).

[32] J. Halpern, J. H. Salim, Forces forwarding element model, Tech. Rep. draft-ietf-forces-model-15 (2008).

[33] R. HAAS, Forces mib, Tech. Rep. draft-ietf-forces-mib-10 (2008).

[34] E. Haleplidis, O. Cherkaoui, S. Hares, W. Wang, Forwarding and control element separation (forces) openflow model library, Tech. Rep. draft-haleplidis-forces-openflow-lib-01 (2012).

[35] D. Liu, B. Khasnabish, H. Deng, Architecture discussion of i2rs, Tech. Rep. 02 (2013).

[36] IUT FG NGN, The softrouter concept and the proposal for the study of separation of forwarding and routing in the next generation network (2004).

[37] Object Management Group, SDN Application Ecosystem RFI, Software Defined Networking (SDN) Working Group (Mars 2013).

[38] Object Management Group, Data-Distribution Service for Real-Time Systems - Version 1.2, `http://portals.omg.org/dds/` (January 2007).

[39] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.-J. Lee, P. Yalagandula, Automated and scalable qos control for network convergence, in: Proceedings of the 2010 internet network management conference on Research on enterprise networking, INM/WREN'10, 2010.

[40] Z. Cai, A. L. Cox, T. S. E. Ng, Maestro: A system for scalable openflow control, Tech. rep. (2010).

[41] A. Voellmy, H. Kim, N. Feamster, Procera: a language for high-level reactive network control, HotSDN '12, 2012, pp. 43–48.

[42] T. Amin, G. Yashar, Hyperflow: a distributed control plane for openflow, 2010.

[43] K. Teemu et al , Onix: a distributed control platform for large-scale production networks, OSDI'10, 2010, pp. 1–6.

[44] A. S.-W. Tam, K. Xi, H. J. Chao, Use of devolved controllers in data center networks, CoRR.

[45] H. Y. Soheil, G. Yashar, Kandoo: a framework for efficient and scalable offloading of control applications, 2012, pp. 19–24.

[46] M. O. S. Volkan Yazici, A. O. Ercan, Controlling a software-defined network via distributed controllers, 2012 NEM Summit Proceedings.

[47] Z. Cao, Z. Li, Analysis of sdn controller cluster in large-scale production networks, Tech. Rep. 00 (2013).

[48] B. Heller, R. Sherwood, N. McKeown, The controller placement problem, in: Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, 2012.

[49] T. Nadeau, P. Pan, Framework for software defined networks, Internet-Draft draft-nadeau-sdn-framework-01, internet-Draft (Oct. 2011).

[50] D. G. Underhill, An extensible network visualization and control framework, Thesis, Stanford University.

[51] Scalable adaptive graphics environment.
URL `http://www.openflow.org/wp/gui/`

[52] M. C. Andrea Simeoni, Extension of the OpenFlow framework to foster the Software Defined Network paradigm in the Future Internet (2011).

[53] S. Shin, J. Kim, Toward service-aware flow visualization over openflow-based programmable networks (8–13 2011).

[54] S. Natarajan, X. Huang, An interactive visualization framework for next generation networks, in: Proceedings of the ACM CoNEXT Student Workshop, CoNEXT '10 Student Workshop, 2010, pp. 1–14.

[55] D. Mattos, N. Fernandes, V. da Costa, L. Cardoso, M. Campista, L. H. M. K. Costa, O. Duarte, Omni: Openflow management infrastructure, in: Network of the Future (NOF), 2011 International Conference on the, 2011, pp. 52–56.
URL `http://www.gta.ufrj.br/omni/`

[56] FI-WARE Consortium, Future internet core platform, Tech. Rep. D2.2, Europeen 7th FP (November 2011).
URL `http://www.fi-ware.eu/`

[57] G. Huang, C.-N. Chuah, S. Raza, S. Seetharaman, Dynamic measurement-aware routing in practice, IEEE Network 25 (3) (2011) 29–34.

[58] R. Antonello, S. Fernandes, C. Kamienski, D. Sadok, J. Kelner, I. G´dor, G. Szabó, T. Westholm, Deep packet inspection tools and techniques in commodity platforms: Challenges and trends, Journal of Network and Computer Applications 35 (6) (2012) 1863–1878.

[59] G. Malkin, RIP Version 2 - Carrying Additional Information, RFC 1723 (Standard) (1994).

[60] Y. Rekhter, T. Li, S. Hares, A Border Gateway Protocol 4 (BGP-4), RFC 4271 (Draft Standard) (2006).

[61] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, J. van der Merwe, The case for separating routing from routers, in: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture, FDNA '04, 2004, pp. 5–12.

[62] M. Schlansker, Y. Turner, J. Tourrilhes, A. Karp, Ensemble routing for datacenter networks, in: Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS '10, 2010.

[63] A. Bianco, R. Birke, L. Giraudo, M. Palacin, Openflow switching: Data plane performance, Communications (ICC), 2010 IEEE International Conference on (2010).

[64] Z. Cai, F. Dinu, J. Zheng, A. L. Cox, T. S. E. Ng, Contract: Incorporating coordination into the ip network control plane, in: Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems, ICDCS '10, 2010, pp. 189–198.

[65] F. Nmeth, A. Stipkovits, B. Sonkoly, A. Gulyás, Towards smartflow: case studies on enhanced programmable forwarding in openflow switches, SIGCOMM Comput. Commun. Rev. 42 (4) (2012) 85–86.

[66] F. O. Silva, M. A. Gonalves, J. H. de S. Pereira, L. J. Camargos, R. Pasquini, P. F. Rosa, S. T. Kofuji, Implementing the domain title service atop openflow (May 2012).

[67] Y. Nakagawa, K. Hyoudou, T. Shimizu, A management method of ip multicast in overlay networks using openflow, in: Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, 2012, pp. 91–96.

[68] S. Civanlar, M. Parlakisik, A. M. Tekalp, B. Gorkemli, B. Kaytaz, E. Onem, A qos-enabled openflow environment for scalable video streaming, IEEE Globecom Workshops.

[69] H. E. Egilmez, B. Gorkemli, A. M. Tekalp, S. Civanlar, Scalable video streaming over openflow networks: An optimization framework for qos routing, in: ICIP, 2011, pp. 2241–2244.

[70] H. E. Egilmez, S. T. Dane, B. Gorkemli, A. M. Tekalp, Openqos: Openflow controller design and test network for multimedia delivery with quality of service, NEM Summit.

[71] A. T. H.E. Egilmez, S. Civanlar, A distributed qos routing architecture for scalable video streaming over multi-domain openflow networks, Proc. IEEE International Conference on Image Processing (ICIP 2012).

[72] H. Egilmez, Adaptive video streaming over openflow networks with quality of service, Ph.D. thesis, Koç University (2012).

[73] R. Penno, T. Reddy, M. Boucadair, D. Wing, S. Vinapamula, Application enabled sdn (a-sdn), Tech. Rep. 01 (2013).

[74] E. Kissel, G. Fernandes, M. Jaffee, M. Swany, M. Zhang, Driving software defined networks with xsp, SDN'12 (2012).

[75] J. Soeurt, I. Hoogendoorn, Shortest path forwarding using openflow, Tech. rep. (February 2012).

[76] C. Kolias, Openflow &sdnfornetworkproviders: Smart devices need smartnetworks (November 2011).

[77] H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, R. Sidi, Sdni: A message exchange protocol for software defined networks (sdns) across multiple domains, Tech. Rep. draft-yin-sdn-sdni-00.txt (2012).

[78] M. Boucadair, C. Jacquenet, Requirements for Automated (Configuration) Management, Internet-Draft draft-boucadairnetwork-automation-requirements-01, IETF Secretariat (Jun. 2013).

[79] S. Shah, K. Patel, S. Bajaj, L. Tomotaki, M.Boucadair, Inter-domain SLA Exchange, Tech. Rep. draft-ietf-idrsla-exchange-01 (June 2013).

[80] J. R. W. Liu, J. Wang, A unified framework for software-defined information-centric network, Tech. Rep. 00 (2013).

[81] A. Detti, N. Blefari Melazzi, S. Salsano, M. Pomposini, Conet: a content centric inter-networking architecture, in: Proceedings of the ACM SIGCOMM workshop on Information-centric networking, ICN '11, 2011, pp. 50–55.

[82] N. Blefari-Melazzi, A. Detti, G. Morabito, S. Salsano, L. Veltri, Information centric networking over {SDN} and openflow: Architectural aspects and experiments on the {OFELIA} testbed, Computer Networks.

[83] FP7-ICT-SAIL, SAILScalable and Adaptable Internet Solutions, Techreport, d-5.2 (D-D.1) Cloud Network Architecture Description (Jul. 2011).

[84] B. J. Ko, V. Pappas, R. Raghavendra, Y. Song, R. B. Dilmaghani, K.-w. Lee, D. Verma, An information-centric architecture for data center networks, 2012, pp. 79–84.

[85] P. Pan, T. Nadeau, Software-defined network (sdn) problem statement and use cases for data center applications, Tech. Rep. 00 (2013).

[86] D. Drutskoy, E. Keller, J. Rexford, Scalable network virtualization in software-defined networks (2012).

[87] C. Rotsos, R. Mortier, A. Madhavapeddy, B. Singh, A. W. Moore, Cost, performance & flexibility in openflow: Pick three., in: ICC, IEEE, 2012, pp. 6601–6605.

[88] R. Raghavendra, J. Lobo, K.-W. Lee, Dynamic graph query primitives for sdn-based cloudnetwork management, 2012, pp. 97–102.

[89] P. Pisa, N. Fernandes, H. Carvalho, M. Moreira, M. Campista, L. Costa, O. Duarte, Openflow and xen-based virtual network migration, in: Communications: Wireless in Developing Countries and Networks of the Future, Vol. 327, 2010, pp. 170–181.

[90] S. Ghorbani, M. Caesar, Walk the line: consistent network updates with bandwidth guarantees, in: Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, 2012.

[91] T. Benson, A. Akella, A. Shaikh, S. Sahu, Cloudnaas: a cloud networking platform for enterprise applications, 2011.

[92] E. Keller, S. Ghorbani, M. Caesar, J. Rexford, Live migration of an entire network (and its hosts) (Oct. 2012).

[93] V. S. Zaborovsky, A. Lukashin, S. Kupreenko, V. Mulukha, Dynamic access control in cloud services, in: SMC, 2011, pp. 1400–1404.

[94] Pantou, Open wireless freedom.
URL `https://openwrt.org/`

[95] OpenFlowHub, Indigo - open source openflow switches.
URL `http://www.openflowhub.org/display/Indigo`

[96] D. Liu, H. Deng, Mobility support in software defined networking, Tech. Rep. 00 (2013).

[97] T. Hau, D. Burghardt, W. Brenner, Multihoming, content delivery networks, and the market for internet connectivity, Telecommunications Policy 35 (2011) 532 – 542.

[98] F. den Hartog, P. Nooren, A. Delphinanto, E. Fledderus, On managed services lanes and their use in home networks, in: Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication, UbiComp '13 Adjunct, 2013, pp. 769–776.

[99] Y. Zhang, J. Luo, H. Hu, Wireless Mesh Networking: Architectures, Protocols and Standards, Wireless Networks and Mobile Communications, Taylor & Francis, 2006.
URL `http://books.google.fr/books?id=7t4szWRwnFkC`

[100] Y. Peng, Y. Yu, L. Guo, D. Jiang, Q. Gai, An efficient joint channel assignment and qos routing protocol for {IEEE} 802.11 multi-radio multi-channel wireless mesh networks, Journal of Network and Computer Applications 36 (2) (2013) 843 – 857.

[101] I. Ahmed, A. Mohammed, H. Alnuweiri, On the fairness of resource allocation in wireless mesh networks: a survey, Wirel. Netw. 19 (6) (2013) 1451–1468.

[102] P. Dely, Towards an architecture for openflow and wireless mesh networks, Ofelia Summer School.

[103] I. COMCAS) (Ed.), Wireless Software Defined Networks: Challenges and Oppportunities, 2013.

[104] N. Bayer, P. Dely, A. Kassler, Openflow for wireless mesh networks, IEEE International Workshop on Wireless Mesh and Ad Hoc Networks (WiMAN 2011).

[105] K.-H. Kim, S.-J. Lee, P. Congdon, On cloud-centric network architecture for multi-dimensional mobility, 2012.

[106] R. Bifulco, M. Brunner, R. Canonico, P. Hasselmeyer, F. Mir, Scalability of a mobile cloud management system, 2012.

[107] L. D. Mendes, J. J. Rodrigues, A survey on cross-layer solutions for wireless sensor networks, Journal of Network and Computer Applications 34 (2) (2011) 523 – 534.

[108] T. Luo, H.-P. Tan, T. Q. S. Quek, Sensor openflow: Enabling software-defined wireless sensor networks, IEEE Communications Letters 16 (11) (2012) 1896–1899.

[109] M. Peng, D. Liang, Y. Wei, J. Li, H.-H. Chen, Self-configuration and self-optimization in lte-advanced heterogeneous networks, Communications Magazine, IEEE 51 (5).

[110] S. Costanzo, L. Galluccio, G. Morabito, S. Palazzo, Software defined wireless networks: Unbridling sdns, Proceedings of European Workshop on Software Defined Networking.

[111] M. Mendonca, K. Obraczka, T. Turletti, The case for software-defined networking in heterogeneous networked environments, in: ACM-CoNEXT 2012, 2012, pp. 59–60.

[112] L. L. Erran, M. Z. Morley, J. Rexford, Cellsdn: Software-defined cellular networks, Tech. rep. (2012).

[113] S. Kumar, J. Turner, J. Williams, Advanced algorithms for fast and scalable deep packet inspection, in: Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems, ANCS '06, 2006, pp. 81–92.

[114] L. E. Li, Z. M. Mao, J. Rexford, Toward software-defined cellular network, Proceedings of European Workshop on Software Defined Networking.

[115] K.-K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, N. McKeown, The stanford openroads deployment, WINTECH '09, 2009, pp. 59–66.

[116] J. Gozalvez, M. C. Lucas-Estañ, J. Sanchez-Soriano, Joint radio resource management for heterogeneous wireless systems, Wirel. Netw. 18 (4) (2012) 443–455.

[117] H. Chan, D. Liu, P. Seite, H. Yokota, J. Korhonen, Requirements for distributed mobility management, Tech. Rep. draft-ietf-dmm-requirements-09.txt (2013).

[118] M. Wasserman, S. Hartman, D. Zhang, Security analysis of the open networking foundation (onf) openflow switch specification, Internet-Draft draft-mrw-sdnsec-openflow-analysis-00, informational (Oct. 2012).

[119] S. project. [link].
URL `http://www.snort.org/`

[120] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, G. Gu, A Framework for Enabling Security Controls in OpenFlow Networks, ACM (Aug. 2012).

[121] K.-K. Yap, Y. Yiakoumis, M. Kobayashi, S. Katti, G. Parulkar, N. McKeown, Separating authentication, access and accounting: A case study with openwifi, Tech. rep. (2011).

[122] J. H. Jafarian, E. Al-Shaer, Q. Duan, Openflow random host mutation: transparent moving target defense using software defined networking, HotSDN '12, 2012, pp. 127–132.