Empowered by Innovation NEC

SDN and its use-cases- NV and NFV

A State-of-the-Art Survey

Sridhar K. N. Rao

NEC Technologies India Limited

sridhar.rao@nectechnologies.in

A White Paper

Contents

List of Figures

List of Tables

SDN and its Use-Cases - NV and NFV : A State-of-the-Art survey

Sridhar K. N. Rao, *Member, IEEE,*

***Abstract*—Three concepts - (a) network programmability by clear separation of data and control planes and (b) sharing of network infrastructure to provide multitenancy, including traffic and address isolation, in large data center networks and (c) replacing the functions that traditionally run on a specialized hardware, with the software-realizations that run on commodity servers - have gained lot of attention by both Industry and research-community over past few years. These three concepts are broadly referred as software defined networking (SDN), network virtualization (NV) and network functions virtualization (NFV). This paper presents a thorough study of these three concepts, including how SDN technology can complement the network virtualization and network functions virtualization. SDN, is about applying modularity to network control, which gives network designer the freedom to re-factor the control plane. This modularity has found its application in various areas including network virtualization. This work begins with the survey of software defined networking, considering various perspectives. The survey of SDN is followed by discussing how SDN plays a significant role in NV and NFV. Finally, this work also attempts to explore future directions in SDN based on current trends.**

Keywords*—Software defined networking, Network Virtualization, Network Functions Virtualization, OpenFlow, Data Center, Overlay, Underlay, Network Planes, Programmable networks.

I. Introduction

Thomas S. Kuhn, in his highly-influential work 'The Structure of Scientific Revolutions', defines the term paradigm as “universally recognized scientific achievements that, for a time, provide model problems and solutions for a community of researchers”. Going by this meaning, Software Defined Networking (SDN) is a new networking paradigm. Referring again to Kuhn's words, SDN is also “sufficiently open-ended to leave all sorts of problems for the redefined group of practitioners to resolve'.

SDN has been defined by different researchers in different terms, and the one which is more general and inclusive one is provided by Heller [64], as follows “is a refactoring of the relationship between network devices and the software that controls them”. The term network device is used to include - gateways, routers, network bridges, switches, hubs, protocol converters, proxy servers, firewalls, network address translators, multiplexers, network interface controllers, modems, terminal adapters, line drivers, wireless access points, etc. We know that every single network device typically has to perform three distinct activities, which are mapped correspondingly to three different planes of the network; Data, Control and Management.

Sridhar K. N. Rao is with the NEC Technologies India Limited as Group Technical Specialist. e-mail: sridhar.rao@nectechnologies.in

Data plane is responsible for processing the transit traffic, which decides what to do with packets arriving on an ingress interface. It is also termed as forwarding plane, as it mainly refers to a forwarding table to decide proper egress interface. From the perspective of packets the Data plane usually handle end-station/user-generated packets that are always forwarded by network devices to other end-station devices. The *control-plane* is concerned with collecting, processing and managing the network information in order to decide the forwarding behavior. Control plane typically includes various tables and suite of protocols that work on these tables. Hence, control plane handles network device generated/received packets that are used for the creation and operation of the network itself. Typical protocols that run at control plane are routing, interface state management, connectivity management, adjacent device discovery, topology or reachability information exchange and Service provisioning. The *management plane* is used to interact - or monitor - with the device, in order to manage the network. Management plane also runs its own suite of protocols (such as SNMP), apart from supporting configurations of interfaces, network (IP subnets) and control-plane protocols. In a traditional network device, the data-plane activities are carried out by dedicated hardware (or 'high-speed code'), while the control plane operations are handled by the device CPU.

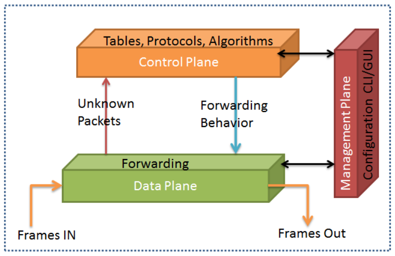

Fig. 1. Planes of A Networking Device

Considering the above description of planes, SDN eliminates the complex and static nature of legacy distributed network architectures by using the **standards-based software abstraction** between the network *control plane* and underlying *data forwarding plane*. Abstractions solve architectural problems and makes architecture move evolvable [55]. And, as mentioned by Scott Shenker, “SDN is about achieving Forwarding, State Distribution and Specification abstractions at network control planes” [81].

In a typical SDN, the network intelligence is logically centralized in controllers (software-based), which enables *the control logic* to be designed and operated on a global network view, as a centralized application, rather than a distributed system [64]. A program running on a logically-centralized controller manages the network directly by configuring the packet-handling mechanisms in the underlying network devices. The packet processing rules are installed on network devices to realize various tasks of managing a network, ranging from routing and traffic monitoring to access control and server load balancing. However, control plane state and logic must inevitably be physically distributed to achieve responsiveness, reliability, and scalability goals [64].

From the realization perspective, an SDN is any network that gives us the flexibility to choose between points on the implementation design axes: centralized to distributed, micro-flow to aggregated-flow, reactive to proactive, virtual to physical, and fully-consistent to eventually-consistent [64]. That is, SDN adds flexibility to control-plane implementation choices. As a result, the network devices become simple packet forwarding devices (the data plane) that can be programmed via an open interface (e.g., OpenFlow [66], [70]).

From the above discussion we can see that SDN includes the following (a) open/standards-based interface to hardware (b) Network operating system (c) Well-defined APIs to write various network applications. In summary, SDN is about clear separation of the data and control planes of the network devices, and about having sufficient abstraction at the control plane to support the provision of novel services in the network.

The SDN's key aspect of providing freedom of refactoring the network control plane to the network designers has found its application in various domains such as network virtualization and network functions virtualization.

Network Virtualization envisages an architecture in which the physical aspects of a network is decoupled with the virtual aspects of a network [21]. The physical network continues the work in the way as it was designed to do: forwarding packets by utilizing standard (routing) protocols. The virtual network, on the other hand, maintains various operational policies such as access control lists, configuration policies, and network services. However, network virtualization, as an use case of SDN, promises to solve various (data-center) networking challenges such as flexibility, resource utilization and on-demand deployments in enterprise data centers.

On the other hand, Network operators are increasingly burdened by complexities as they attempt to accommodate the ever-growing demand for new services and more bandwidth [185]. In addition, a growing and increasingly diverse population of proprietary appliances that make service additions and upgrades more and more difficult. Network Functions Virtualization (NFV) aims to address these problems by evolving standard IT virtualization technology to consolidate many network equipment types onto industry standard high volume servers and switches [174]. It involves implementing network functions in software that can run on a range of industry standard server hardware, and that can be moved to, or instantiated in, various locations in the network as required, without the need to install new equipment. The type of appliances addressed by NFV are typically turn-key in-line systems that maintain real-time state of subscriber mobility, voice and media calls, security, contextual content management, etc. Hence, operators have turned their focus to streamline their infrastructures through NFV, and this necessitates a consolidation of network functions onto industry-standard servers, switches, and storage hardware located in data and distribution centers. Network Functions Virtualization is highly complementary to (SDN), but they are not dependent on each other.

In this report, we provide a thorough survey of SDN, and its use-cases: NV and NFV. The Section II covers various aspects of SDN ranging from history to use-cases and applications . The Section III discusses network-virtualization and the role of SDN. Similarly, Section IV focuses on network functions virtualization and SDN. The current trends and future directions are discussed in Section V, and the Section VI concludes the report.

II. SDN - A SURVEY

In this section, we provide a brief survey of the state-of-the-art in SDN. We begin with discussing the historical perspective of programmable networks from early ideas to recent developments, covering the use-cases and drivers. Then, we present the SDN architecture models and the OpenFlow standard in particular, discussing current alternatives for implementation and testing SDN-based protocols and services. We also provide a summary on SDN tools, languages, standards, and open-source/commercial products and solutions. Further, we examine current and future SDN applications, and explore promising research directions based on the SDN paradigm.

A. *SDN Precursors: Road to SDN*

SDN was designed to simplify network hardware while improving the flexibility of network control. While SDN has received a considerable amount of industry attention, it is worth noting that the idea of programmable networks and decoupled control logic has been around for many years. In this section, we provide an overview of early programmable networking efforts, precursors to the current SDN paradigm that laid the foundation for many of the ideas we are seeing today. The detailed study can be found here [173] and here [174]. The Table I and Figure 2 provide the summary of such technologies that act as precursors to SDN.

The Open Signaling (OPENSIG) working group began in 1995 with a series of workshops dedicated to making ATM, Internet and mobile networks more open, extensible, and programmable" [31]. The core of their proposal was to provide access to the network hardware via open, programmable network interfaces; this would allow the deployment of new services through a distributed programming environment. Motivated by these ideas, an IETF working group was created, which led to the General Switch Management Protocol (GSMP) [38], a general purpose protocol to control a label switch. In the mid 1990s, the Active Networking [93], [94] initiative proposed the idea of a network infrastructure that would be programmable for customized services. However, it never achieved the significance necessary to be transferred to widespread use and industry deployment, mainly due to practical security and performance concerns [74].The 4D project [85], [47], [28] advocated a clean slate design that emphasized separation between the routing decision logic and the protocols governing

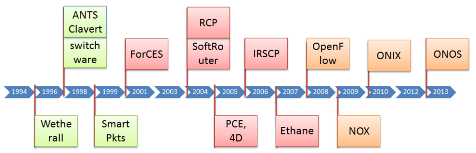

Fig. 2. SDN Precursors

Technology	Remarks
Open Signaling	Based on a clear separation between switching hardware and control software, the concept of open signaling creates an open programmable networking environment. Network entities can be realized as high level objects with well defined software interfaces, facilitating the creation of multiple mechanisms for connection management.
Active Networking	Refers to the addition of user=controllable computing capabilities to data networks. Network is no longer viewed a passive mover of bits, but as a general computing engine.
NETCONF [41]	Defines a simple mechanism through which a network device can be managed, configuration data information can be retrieved, and new configuration data can be uploaded and manipulated. Protocol allows the device to expose a full and formal APIs
ForCES [39]	Defines a framework and associated protocol(s) to standardize information exchange between the control and forwarding plane - this allows CEs and FEs to become physically separated standard components.
4D [47]	Based on three Principles: network-level objectives, network-wide views, and direct control. Re-factor functionality into four components: data, discovery, dissemination, and decision planes
Ethane [33]	network architecture for the enterprise single network-wide fine-grain policy, and then enforces it directly centralized controller that manages the admittance and routing of flows
RCP [28]	Refactoring the IP routing architecture to create a logically centralized control plane separated from forwarding elements. In RCP, the focus is on the BGP decision process, and the route control process needs large operators' perspective
SANE [24]	A protection architecture for enterprise networks. Defines a single protection layer that governs all connectivity within the enterprise. All routing and access control decisions are made by a logically-centralized server that grants access to services according to declarative access control policies
SS7 [73]	Common Channel Signaling (CCS) is a signaling method which provides control and management in the telephone network. CCS uses a separate out-of-band signaling network to carry signaling messages. SS7 is a CCS system.

TABLE I. SDN-RELATED TECHNOLOGIES

the interaction between network elements. Finally, Ethane [33] laid the foundation for what would become Software-Defined Networking. Ethane's identity based access control would likely be implemented as an application on top of a SDN controller such as NOX [48]. A parallel approach to Software-Defined Networking is under development by the IETF Forwarding and Control Element Separation (ForCES)Working Group [39].

B. SDN Architecture

Figure 3 depicts the SDN architecture. As shown in the figure, there are three different tiers:

- Application Tier: Encompasses solutions that focus on the expansion of network services. These solutions are mainly software applications that communicate with the controller.

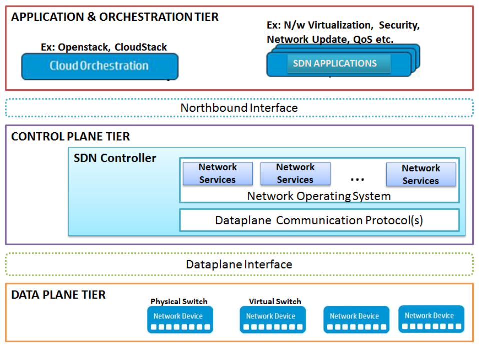

Fig. 3. SDN Architecture

- Control Plane Tier: Includes a logically-centralized SDN controller, that maintains a global view of the network that takes requests through clearly defined APIs from application layer and perform consolidated management and monitoring of network devices via standard protocols.
- Infrastructure or Data-plane Tier: Involves the physical network equipment, including Ethernet switches and routers. Provides programmable and high speed hardware and software, which is complaint with industry standards.

At the bottom layer, the physical network consists of the hardware forwarding devices which store the forwarding information base (FIB) state of the network data plane (e.g., TCAM Entries and configured port speeds), as well as associated metadata including packet, flow, and port counters. The devices of the physical network may be grouped into one or more separate controller domains, where each domain has at least one physical controller. Dataplane interface or standards-based protocols, typically termed as 'southbound protocols', define the control communications between the controller platform and data plane devices such as physical and virtual switches and routers. There are various southbound protocols such as Openflow, PCEP, SNMP, OVSDB, etc.

The control-plane tier is the core of the SDN, and is realized by the controllers of each domain, which collect the physical network state distributed across every control domain. This component is sometimes called the Network Operating System (NOS), as it enables the SDN to present an abstraction of the physical network state to an instance of the control application (running in Application Layer), in the form of a global network view.

Northbound Open APIs refer to the software interfaces between the software modules of the controller and the SDN applications. These interfaces are published and open to customers, partners, and the open source community for development. The application and orchestration tools may utilize these APIs to interact with the SDN Controller.

Application layer covers an array of applications to meet different customer demands such as network automation, flexibility and programmability, etc. Some of the domains of SDN applications include traffic engineering, network virtualization, network monitoring and analysis, network service discovery, access control, etc. The control logic for each application instance may be run as a separate process directly on the controller hardware within each domain.

From the above description of the architecture, we can see that the SDN controller acts as a key element - both in terms of south-bound and north-bound interactions. Ashton et. al., [192] prepared a checklist of what to look for in an SDN controller. In this list, they included the following: (a) OpenFlow Support (versions and extensions) (b) Network Functionality (network isolation, QoS support) (c) Programmability (APIs and policy definitions) (d) Reliability (redundancy and availability) (e) Centralized Monitoring and Visualization (f) Network virtualization support (creation and management) (g) Scalability (Number of switches, hosts, flows) (h) Performance (delays, drops and throughput) (i) Security (Authentications and filters) and (j) Vendor Characteristics. This checklist provides an understanding of challenges in developing a SDN controller.

C. SDN Control Models

Enterprises and network operators who deploy a software-defined network (SDN) typically use one of the two different models of SDN – centralized and distributed; and each model has different infrastructure elements and requirements to consider. A successful deployment will require choosing the right SDN architecture, then testing it in an organized way, based on the right infrastructure.

1) Centralized controller model - or Revolutionary model: The centralized model of SDN technologies, as discussed in the SDN precursors subsection, has evolved from researchers who aimed to replace adaptive discovery, a distributed process, with central control of forwarding. In this architecture, a central software process – or centralized SDN controller – maintains the entire topology and status of the network's devices and understands the network addressing and connectivity from the user's perspective. The central process running on the controller may support various algorithms and policies to define routes through the network for each and every flow [135], [153]. The paths are created by addressing all devices along the way to update their (forwarding) table to forward the packets along the path correctly. The OpenFlow protocol was designed to support the centralized controller to communicate forwarding table changes to network devices. This communication could happen either proactively, based on a network map, or reactively, in response to a request from a device. The challenge in this SDN approach is the lack of a proven model for centralized control, as there is no proof that central control of networking can scale and there exists no currently accepted technology to test its capabilities. The problems of scalability and central control can be addressed in a data center, a delivery network or a WAN core, rather than whole of the Internet or even an enterprise WAN. Deploying centralized SDN in these smaller domains can spread widely and quickly. However, recently various researchers and vendors are working to explore the best way of linking disparate SDN domains into a complete end-to-end network.

2) Distributed SDN – or evolutionary model: The distributed nature of networking intelligence in the Internet has been highly successful. Also, the same control protocols that are used in the Internet have also taken hold in local and private networks. Many experts and vendors, who have both witnessed and been part of this success of adaptive distributed model, believe the 'purist-model' (fully centralized SDN strategy), is a revolutionary one and the evolutionary approach is more prudent. To this group, which includes many IP experts and router equipment vendors, the software that should be defining network behavior is higher-level software, perhaps even application software [135].

The distributed model 'adds' mechanisms of software-based control to traditional IP and Ethernet networks. We should note that the distributed model, like the centralized model, accepts the need to gather information about network status and collect it at a central point where it can be acted on to manage the performance. However, in the distributed model, the goal of SDN is to offer more controllable behavior. Typically, such goals are achieved by leveraging on various existing protocols like MPLS, GRE, and policy-based protocols. For example, PCRF (Policy and Charging Rules Function) - part of the Evolved Packet Core (EPC) that supports service data flow detection, policy enforcement and flow-based charging, could be the principal means by the SDN controllers decide how to set up and manage flows [166]. The main challenge for distributed SDN is that it is yet to prove that it can offer the kind of control granularities over traffic and connectivity that centralized SDN technologies could offer. In addition, there's also the problem of what should be the accepted framework for applying distributed SDN principles - when various vendors choose different approach to integrate SDN into their existing closed systems.

D. SDN and Openflow

In the initial white-paper [70], OpenFlow was simply referred as "a way for researchers to run experimental protocols in the networks they use every day". The analogy used by the authors in describing openflow is to think of openflow as a general language or an instruction set that lets one write a control program for the network rather than having to rewrite all of code on each individual router.

In centralized SDN architecture - the model that standards group Open Networking Foundation (ONF) support - the key element is the connecting technology that communicates central control decisions to devices. OpenFlow has become the official protocol to use in a centralized SDN model to make high-level routing decisions. As a result, the creation of central-control SDN must be based on selecting devices and control software that support the OpenFlow standard. As summarized by Reitblatt et. al., [84] "Despite the conceptual appeal of centralized control, an OpenFlow network is still a distributed system, with inevitable delays between the switches and the controller".

1) Openflow roadmap: Initially the OpenFlow protocol standardized a dataplane model and a control-plane API, mainly by relying on the technologies that legacy switches already supported [174]. For example, network elements included the support of fine-grained access control and flow monitoring. On these elements, enabling OpenFlow's initial set of capabilities was as easy as performing a firmware upgrade (without any hardware upgrade) to make it openflow-capable. It is well-known that OpenFlow's initial target deployment scenario was campus networks, meeting the needs of a networking research community actively looking for ways to conduct experimental work on network architectures within a research-friendly operational setting - the 'Clean-Slate' program [83]. In 2008-09, the OpenFlow group at Stanford (in partnership with 7 other universities) led an effort to deploy OpenFlow testbeds across many campuses and demonstrate the capabilities of the protocol both on a single campus network and over a wide-area backbone network spanning multiple campuses [34]. As real SDN use cases materialized on these campuses, OpenFlow began to take hold in other realms, such as data-center networks, where there was a distinct need to manage network traffic at large scales. This led many experts to state that SDN/Openflow "was born in the campus, matured in the data center" [186]. Although OpenFlow includes many of the principles from earlier work on the separation of control and data planes, the rise of OpenFlow offered several additional intellectual contributions - such as (a) Generalizing network devices and functions, (b) the vision of a network operating system (c) distributed state management techniques [174].

2) Openflow Pros and Cons: Over the past few years, almost all of the major network-device vendors have announced OpenFlow support. As there are several versions of the OpenFlow standard, vendors may not have released software for the latest version. A number of OpenFlow-compatible controllers can provide central control for a community of devices, and many OpenFlow-based SDN trials and deployments have been conducted using these tools. This success can be attributed to either a planned or un-planned collaborations between equipment vendors, chipset designers, network operators, and networking researchers.

OpenFlow is not yet a complete standard, and is still undergoing significant changes. Tom Nolle [132], highlighted following shortfalls of openflow. Openflow doesn't have a mechanism for fully managing devices, particularly for controlling port/trunk interfaces and queuing. OpenFlow doesn't communicate network status fully either, so central control software will need some source of information about loads, traffic and operational state for nodes and trunks. OpenFlow also lacks a specific process for establishing the paths to take between controller and switch, so there's a question of how OpenFlow networks "boot from bare metal," meaning how they become operational from a newly installed state. Finally, majority of the existing controllers have exposed northbound APIs, by which any application or management software can gain access to the features supported by the controller. However, it is still not clear what kind of software is available to take advantage of these interfaces and how it would be integrated with the controller. All of this makes it hard to conceptualize how a current network could evolve to support a centralized SDN model. The capabilities of the control software that runs above the SDN controller itself, are the key to success for the centralized SDN model [132].

E. Performance Studies in SDN

SDN raises significant scalability, performance, robustness, and security challenges. In the subsequent paragraphs we review a number of research efforts focusing on addressing these issues at the switch [68] and controller [96] [105] levels.

In DIFANE [107], flow entries are proactively pushed to switches in an attempt to reduce the number of requests to the controller. Devoflow [36] proposes to handle short-lived flows in switches and long-lived flows in the controller to mitigate flow setup delay and controller overhead. The work proposed in [72] advocates replacing counters on ASIC by a stream of rule-matching records and processing them in the CPU to allow efficient access to counters. Similarly, some recent proposals like [67] have advocated adding a general-purpose CPU, either on-switch or nearby, that may be used to supplement or take over certain functions and reduce the complexity of the ASIC design. Authors of [68] apply network processor based acceleration cards to perform OpenFlow switching. They propose and describe the design options and report results that show a 20% reduction in packet delay. Also, in [92], an architectural design to improve look-up performance of OpenFlow switching in Linux is proposed.

FLARE [76] is a new network node model focusing on deeply programmable networks" that provides programmability for the data plane, the control plane, as well as the interface between them. A recent study shows that one single controller can handle up to 6.000.000 flows/s [3]. However, for increased scalability and especially for reliability and robustness, it has been recognized that the logically centralized controller must be physically distributed. Onix [62], Kando [51], and HyperFlow [95] use this approach to achieve robust and scalable control plane. In [64], trade-offs related to control distribution, such as staleness versus optimality and application logic complexity versus robustness to inconsistency are identified and quantified. In [53], the controller placement problem is discussed in terms of the number of controllers needed and where to place them in the network. In [34] a SDN variant, inspired by MPLS, is proposed along with the notions of edge controllers and fabric controllers: the edge-controller controls ingress and egress switches and handle the host-network interface, while the fabric controller handles fabric switches and the operator-network interface. The work presented in [106] proposes a software-defined traffic measurement architecture, which separates the measurement data plane from the control plane.

F. SDN Tools and Platforms

1) Controllers: The separation of control and data plane - a decoupled system - has been compared to an operating system [17], in which the controller provides a programmatic interface to the network that can be used to implement management tasks and offer new functionalities. The work presented in [25] discusses important aspects in controller design including hierarchical control, data model, scalability, and extensibility. There are various open-source controllers such as NOX [17], Maestro [29], Beacon [22], SNAC [23], Helios [5] and Trema [20]. There are controllers that are written in Java such as

Jaxon [7], Beacon [1], Maestro [29] and Floodlight [38], and some are written in Python such as POX [16] and Ryu [17] and some written in C such as ovs-controller [14] and MUL [8]. There are also some specialized controllers such as FlowVisor [97], and RouteFlow [77] [86], SNAC [19] and NodeFlow [64].

More recently, OpenDaylight's [108] release of Hydrogen [109] is one of the largest open source initiatives. OpenDaylight is an open-source platform for network programmability to enable SDN and create a solid foundation for Network Functions Virtualization (NFV) for networks. The Hydrogen release includes Controller, along with plugins, libraries and tools.

2) Switches: There are various implementations of open-source openflow switches and switching platforms. Some of the examples include Open vSwitch [14], Pantou [15], of-softswitch [11] , Indigo [6], OpenFaucet [110], OpenFlowJ [111]. Open vSwitch is an OpenFlow switch that is used both as a virtual-switch in virtualized environments and also has been ported to multiple hardware platforms. It is now part of the Linux kernel on all popular distributions such as Ubuntu and Fedora. Pica8's Xorplus is an open switch software platform for hardware switching chips that includes an L2/L3 stack and support for OpenFlow. Indigo, is from the developers of floodlight controller, and is designed as a 'for-hardware-switching' OpenFlow implementation based on the Stanford reference implementation. Similar to controllers, there are switching stacks implemented in different languages such as Java (OpenFlowJ), Python (OpenFaucet), C (Pantou), and Haskell (Nettle).

3) Metering, Measuring, Management: There has been various tools for monitoring and measuring, such as OFRewind [112], OESS [113], Quagga [114], FlowScale [115], OpenTM [116], ENVI [117] and LAVI [118]. Below, we describe few of these tools under the headings of monitoring and management tools.

- Monitoring Tools:
  ENVI: Extensible Network Visualization and Control Network visualization and control Framework. EVNI can show network topologies and custom controls as well. Topology and network information can even be queried and displayed. ENVI is extensible and available as a Python library and NOX add-on. The source code and documentation available freely, which is developed in Java
  LAVI: LAVI acts as a backend for Network visualization. It can be both enhanced and customized according to requirement, and also available as NOX module. The source code available freely, which is developed in c/c++.
  OpenTM: OpenTM is a traffic matrix estimator for openflow networks. It has built-in features to directly and accurately measure traffic matrix with low overhead at open flow switch. The routing information is learned from the OpenFlow controller to intelligently choose the switches from which to obtain flow statistics.
- Management Tools:
  OFRewind: OFRewind records traffic at both control and data layers. It has the ability to replay data at various levels and on different hardware configurations. OFrewind, which is supported by the OpenFlow group, is a good tool for debugging and troubleshooting network at various levels.
  FlowScale: FlowScale is an excellent tool for distributing traffic over multiple physical switch ports. It includes features such as policy configuration, hot swapping, traffic mirroring and failover support. It is mainly seen as a good tool for debugging traffic flow.
  OESS : An abbreviation for Open exchange software suite. It is a good tool for configuring and controlling dynamic virtual circuit networks. It includes significant amount of features with simple user interface. It also exposes APIs to use OESS data in other application. OESS, is supported by strong NDDI group, and provides variety of options to control the traffic flow of network.

4) Verification, Validation, testing and debugging: To cater the various needs such as verification of the network flow, understanding the behaviors of the network devices like switches, hosts and controllers, bug fixing of the network, finding inevitable delays affecting communication with the controller and checking the establishment of the network, there have been various tools developed under the heading of verification and validation. Some of example tools are NICE [32], CBench, Anteater [69], OFRewind [158], ndB [49], OFLOPS [119], Header Space Analysis [133]. The realm of formal verification in SDN [131] [144] [155] [156] can be categorized under the following headings. Under each heading, the existing works are enlisted. The detailed description of formal verification in networking in general and SDN in particular can be found in the recent work [172].

- **Data Plane Verification**: Anteater [69], Header Space Analysis (HSA) [133], FlowChecker [134], VeriFlow [59], NetPlumber [133], NetSigtht, ndb [49], modeling and performance evaluation of openflow-switch [57].
- **Control Plane Verification**: NICE Framework [32], isolation and localization of software faults [159], machine verification of controllers [152], per-packet and per-flow consistency of network updates [145] (also Frenetic framework [165], FlowLog [151], data state and network state abstractions for model-checking SDN Controller [150], safe update protocol [165].
- **Network Debugging**: ndb [49], Header space analysis , OFRewind framework, isolation and localization of software faults in SDN, Pip [160], troubleshooting [154] [157] [161] and Automatic Debugging [161].
- **Protocol Verification**: Proof assistant Coq tool [163] for creating a featherweight version of the OpenFlow protocol [131].
- **Property Verification**:The tools Anteater [69], Header Space Analysis (HSA) [133], FlowChecker [134], VeriFlow [59] use an automatic solver to check properties of a logical representation of switch configurations.
- **Reachability Analysis and Loop Detection**: Header Space Analysis (HSA) [133], Veriflow, NetPlumber, AP Verifier [137], Network configuration in a box [139] .
- **Isolation Verification**: AP Verifier and Splendid Isolation [138]

- **Configuration Management**: FlowChecker, AntEater [69], ConfigChecker,
- **Network Security**: FLOVER [141] Firewall Analysis [142], Protocol-manipulation attacks [143].
- **Automatic Synthesis**: Reactive Synthesis and Model Checking [146]

Below, we provide the description of the some of the important tools:

OFLOPS: Is the abbreviation of Open FLow Operations Per Second. OFLOPS is a performance testing tool for OpenFlow switches. It detects problems appearing when the device is heavily loaded, but is also capable of discovering some inconsistencies between data-plane and control-plane. OFLOPS is a C-language based open source test framework, which is focused on the performance aspect of the implementation. **OFTest** is an unified framework used to test correctness of OpenFlow switches. OFTest is a Python based open source test framework. **NICE-OF**: is a python based open-source tool for testing OpenFlow Controller application for NOX controller platform. NICE-OF Uses novel method of symbolic execution and model checking to test the controller application. NICE tool finds bugs in controller applications through systematic testing of application behaviors. **Cbench**: It is a program for testing OpenFlow controllers. Cbench emulates a bunch of switches which connect to a controller, send packet-in messages, and watch for flow-mods to get pushed down into the flow-tables of the switches. It is an Open Source tool developed using PERL and C languages. **OFTRACE**: Debugging open flow Control Traffic. Control channel traffic can be captured using tcpdump or wireshark and then parsed using oftrace to understand the type of control messages. Python based test framework **AntEater**: Anteater, a tool that analyzes the data plane state of network devices. Anteater's primary goal is to detect and diagnose a broad and general class of network problems. The system detects problems by analyzing the contents of forwarding tables contained in routers, switches, firewalls, and other networking equipment. **HSA**: Header Space Analysis statically analyze the data plane configuration to detect connectivity and isolation errors. It is a framework that can identify a range of network configuration problems.

5) Simulation and Emulation: Simulation has been used extensively to evaluate network performance. In particular, simulation provides a detailed operational view of the network protocols at play and their behaviors. There has been various efforts in developing OpenFlow network simulator and emulators such as ESTINET [120], NS3 [54], CloudSim [121] NetKit/AutoNetKit [122], [123], MiniNet [124] [63], VL2 [125], CORE [126], Air-in-a-Box [127]. Below, we provide the description of few simulator or emulators.

EstiNet 8.0 is an openflow network simulator-cum-emulator that can simulate hundreds of openflow (version 1.1.0) switches. It can also run the real-world NOX controller, without any modification, during simulation to control these switches. It is one of the scalable simulator/emulators for openflow. Being a commercial product, it is made to simulate the real world scenarios with excellent compatibility features.

The Mininet open-source network simulator is designed to support research and education in the field of Software Defined Networking systems. Mininet creates a simulated network that runs real software on the components of the network so it can be used to interactively test networking software. Mininet is designed to easily create virtual software-defined networks consisting of an OpenFlow controller, a flat Ethernet network of multiple OpenFlow-enabled Ethernet switches, and multiple hosts connected to those switches. It has built-in functions that support using different types of controllers and switches. We can also create complex custom scenarios using the Mininet Python API. Mininet uses Linux network namespaces to create virtual nodes in the simulated network.

ns-3 [54] is a open source discrete-event network simulator for Internet systems. NS3 simulations can support OpenFlow switches. However, it requires a simulation-only controller writtenwithin the ns-3 namespace, which the OpenFlow switches refer to via a static singletonclass. NS3 simulation credibility needs to be improved. In its current state, the NS-3 Click Integration is limited to use only with L3, leaving NS-3 to handle L2.

6) Languages for Programming Software Defined Networks: The SDN paradigm began with the development of the NOX controller and the OpenFlow forwarding specification. NOX's programming model was based on OpenFlow messages; each incoming OpenFlow message was a NOX event. However, there have been various high-level programming languages proposed for OpenFlow networks. Some of them are RIPL [128], Nettle [98], Pyretic [167] Frentic [44], NetCore [129], OF-Lib [130], Procera [99], FML [55] NetKat [168], declarative language for SDN [169], logic programming for SDN [170]. Below we briefly describe few languages.

RipL [128]: is a Python library to simplify the creation of data center code, such as OpenFlow network controllers, simulations, or Mininet topologies. Some of the features of RipL include, topology graph for creating custom data center simulators, a way to build a functional multipath-capable network controller, a no-code way to run a fat-tree or other data center topologies on Mininet.

Nettle [98] [100]: Is a good alternative to NOX for writing controller programs for openflow networks. Nettle supports event driven callback mechanism for writing switch response mechanisms.It is based on functional reactive programming, which makes it possible integrate components written in higher level domain specific languages (DSL).

Frenetic [44]: Frentic is a network programming language with higher level functionalities required for openflow networks. It uses various algorithms to encapsulate low level complexities and create clear functionalities for programmers.

G. SDN Commercial Products/Solutions

The Table II, provides the list of companies and their SDN products/solutions. As there are too many companies developing various SDN related products and solutions, it would be difficult to accommodate all in one single table. The below list provides the list of other vendors/solutions classified in terms of different SDN aspects.

Name of the Company	Product Description
NEC	Programmable Flow Controller(s), Programmable Flow Switching Platforms, PF1000 and PF1200 (vSwitches)
IBM	Switching Platforms, Programmable Network Controller ,SDN for Virtual Environments (vSwitch)
VMWare	Nicira Network Virtualization Platform (NVP),vSphere Distributed Switch (vSwitch)
Arista	Switching Platforms, Arista Extensible Modular Operating System (SDK),
Big Switch	Big Network Controller, Big Tap, Big Virtual Switch (vSwitch), Switch Light
Midokura	Midonet Network Virtualization solution
Cisco	Switching Platforms,Cisco XNC Controller,Cisco OnePK (SDK), Nexus 1000V Series Switches (vSwitch)
Brocade	Switching Platforms, Application Resource Broker (ARB),
Intel	Alta - FM6000
Dell	Switching Platforms
HP	Switching Platforms, HP Virtual Application Networks SDN Controller, HP FlexFabric Virtual Switch 5900v (vSwitch)
Microsoft	Hyper-V Virtual Switch (vSwitch)
Plumgrid	Virtual Network Infrastructure Platform for Cloud Data Centers
Fujitsu	ServerView Resource Orchestrator, Switching Platform, virtual appliance platform
Nuage	Virtualized Services Controller (VSC), Virtualized Services Platform (VSP), Virtual Routing and Switching (vSwitch), Virtualized Services Directory (VSD)
Extreme Networks	Switching Platforms
Huawei	Switching Platforms, Service provisioning and orchestration
Juniper	Switching Platforms, Contrail Controller,
Ericsson	Switching Platforms

TABLE II. SDN COMMERCIAL VENDORS

- Switching Platform Vendors: Pica8 , Transmode, Centec Networks, Accedian Networks, Noviflow, Netgear, Intune Networks, Cyan.
- Service provisioning and orchestration: Amartus, Cloud Scaling, NetYce, Cyan, Accedian Networks.
- L4-7 Appliances Vendors: A10 Networks, Openwave Mobility, LineRate Systems.
- Network Operating System: 6WINDGate Networking Software.
- Controllers : Netsocket vFlow Controller, Tail-f systems Network Control System (NCS), Plexxi Control, Vello CX16000, NTT Virtual Network Controller, Turk Telecom YakamOS .
- Virtualization Software: Axsh's OpenVNet.
- Network and configuration management Vendor: SDNsquare (SDN2)
- Applications : Tekelec's Diameter Signaling router and policy server, Estinet's Network Emulator, Accedian's NanoNID, Netsocket's vNetCommander,
- Performance assurance and measurement: DVQattest active test agent, SQprobe - software probe, StealthWatch System
- Software Development Platform : One Convergence's Edge Acceleration Platform
- Silicon Vendors: Axxia Communication Processors, Centec Networks, Netronome, EZChip, and Broadcom .
- Test and Measurement Solution: Radware's DefenseFlow
- Virtual Network Applications: Embrane's Heleos
- Virtual Switch: Netsocket's Virtual switch, Noviflow's NoviWare 100, Accedian Networks' V-NID Product Suite.

H. Applications of SDN

1) SDN Drivers and Use-Cases/Applications: Jim Metzler [171] argued that "It is important to realize that few, if any, IT organizations want a SDN. What IT organizations want is to solve current problems and add value. If SDN can enable them to do that better than alternative approaches to networking, then it will be widely adopted". One of the key strategic benefits that SDN provides is that it enables dynamic access to state information such as the status of queues. As a result, applications can specify the services that they want from the network. For example, an application that is delay sensitive can request the end-to-end path with the least amount of delay. Some of the additional benefits of SDN in the data center are that it enables network virtualization as well as the automation of provisioning and management and the implementation of policies such as QoS.

The combination of highly virtualised environments inside enterprise networks, along with an explosion of mobile traffic, are exposing the limitations of existing networks, and driving the need for a new era of dynamic and scalable networks of the future. SDN enables organizations to significantly increase the utilization of their WAN links and hence reduce cost. In the campus, SDN enables the implementation of policies such as QoS and it also enables the automation of key security functionality. Finally, in BYOD and 'bandwidth-intensive' applications are some of the key drivers for SDN adoption.

We can classify the Applications of SDN with respect to following domains

- Data Centers
- Service Providers
- Campus Networks
- Other Networks

Below, we describe some of the use-cases within each of these domains. A table of SDN use-cases, from which some of the below examples are taken, is provided in SDNCentral-webspace [187].

2) Data Center Networks:

- Network Virtualization: Network virtualization, which will be discussed in detail in Section III is one of the major applications of SDN in datacenter networks. Two major scenarios of network virtualization, for which SDN is used in datacenters, are to realize multi-tenant Networks and stretched/extended networks. SDN is applied to dynamically create segregated topologically-equivalent networks (multitenancy) across a datacenter, and to create location-agnostic networks, across racks or across datacenters, with VM mobility and dynamic reallocation of resources (stretched/extended networks). Some of the advantages of SDN in datacenters for network virtualization are better utilization of datacenter resources, faster turnaround times, improved recovery times in disasters, overcome various limitations such as 4K of VLAN.

- Service Insertion or Service Chaining: To create dynamic chains of L4-7 services on a per tenant basis to accommodate self-service L4-7 service selection or policy-based L4-7 (e.g. turning on DDoS protection in response to attacks, self-service firewall, IPS services in hosting environments, DPI in mobile WAN environments). The advantages of using SDN in this case, as claimed by authors were [187], provisioning times reduced from weeks to minutes, improved agility and self-service allows for new revenue and service opportunities with substantially lower costs to service.
- Tap Aggregation: Provide visibility and troubleshooting capabilities on any port in a multi-switch deployment without use of numerous expensive network packet brokers (NPB). The advantages of using SDN are: Dramatic savings and cost reduction, savings of 50-100K per 24 to 48 switches in the infrastructure. Less overhead in initial deployment, reducing need to run extra cables from NPBs to every switch.
- Energy Saving: Heller et al. [52] propose ElasticTree, a network-wide power manager that utilizes SDN to find the minimum-power network subset which satisfies current traffic conditions and turns off switches that are not needed. As a result, they show energy savings between 25-62% under varying traffic conditions. The Honeyguide [88] approach to energy optimization in which it uses virtual machine migration to increase the number of machines and switches, that can be shutdown.
- VM Migration: In [46], an algorithm for efficient migration with bandwidth guarantees using OpenFlow was proposed. LIME [58] is a SDN-based solution for live migration of Virtual Machines, which handles the network state during migration and automatically configures network devices at new locations.

3) Service Provider and Transport Networks: SDN provides a fully programmatic Operator-Network interface, which allows it to address a wide variety of operator requirements without changing any of the lower-level aspects of the network. SDN achieves this flexibility by decoupling the control plane from the topology of the data plane, so that the distribution model of the control plane need not mimic the distribution of the data plane. Below, we enlist few use-cases related to service provider networks.

- Dynamic WAN reroute: This work focus on forwarding large amounts of trusted data bypassing expensive security devices. SDN is used to provide dynamic yet authenticated programmable access to flow-level bypass using APIs to network switches and routers. The advantages of using SDN in this case, as mentioned by authors [187] are: Savings of hundreds of thousands of dollars unnecessary investment in 10Gbps or 100Gbps L4-7 firewalls, load-balancers, IPS/IDS that process unnecessary traffic.
- Dynamic WAN interconnects: To create dynamic interconnects at Internet interchanges between enterprise links or between service providers using cost-effective high-performance switches. The advantages of using SDN in this case are: (a) Ability to instantly connect (c) Reduces the operational expense in creating cross-organization interconnects providing ability to enable self-service.
- Bandwidth on Demand: Enable programmatic controls on carrier links to request extra bandwidth when needed (e.g. DR, backups). The advantages of using SDN in this case are reduced operational expense allowing self-service by customers, and increased agility saving long periods of manual provisioning.
- NFV and Virtual Edge : In combination with NFV initiatives, replace existing Customer Premises Equipment (CPE) at residences and businesses with lightweight versions, moving common functions and complex traffic handling into POP (points-of-presence) or SP datacenter. The advantages of using SDN are: increased usable lifespan of on-premises equipment, improved troubleshooting, and flexibility to sell new services to business.
- Software defined Internet Architecture [81]: Borrows from MPLS the distinction between network edge and core to split tasks between inter-domain and intra-domain components. As only the boundary routers and their associated controller in each domain are involved in inter-domain tasks, changes to inter-domain service models would be limited to software modifications at the inter-domain controllers rather than the entire infrastructure.
- Other Works: Another approach to inter-AS routing [23] uses NOX and OpenFlow to implement BGP-like functionality. Alternatively, an extensible session protocol [61] supports application-driven configuration of network resources across domains.

4) Campus/Enterprise/Home Networks: There have been various use-cases within campus/enterprise networks. The applications such as Video Streaming and Collaboration, BYOD and Seamless Mobility and Network Virtualization (Slicing/Traffic Isolation), Application Aware Routing, are explored using SDN. Below we enlist few works that have been addressed by various researchers and vendors in these types of networks.

- Security and Policy Enforcement: Anomaly detection [71] stateless/stateful firewall, dynamic network access control [78], intrusion detection (IDS), SLA policies Management, consistent network updates [84], Network Administration Control and troubleshooting [102]
- Management Simplification [60]: Planned maintenance, outsourcing [42], instrumentation [30], unified control and management [45].
- Traffic Management: Traffic engineering, IGP migration, Traffic monitoring, Flow/load balancing [50] [101], Efficient Resource Management, Bandwidth Guarantees,
- Visualization and Monitoring [75]: NetGraph [82] ,

5) Other Networks: **Infrastructure-based wireless networks**: Several efforts have focused on ubiquitous connectivity in the context of infrastructure-based wireless access networks, such as WiFi and cellular networks. Some of the related works are OpenRoads project [103], [104], Odin Project [90], OpenRadio [22], software defined cellular networks [65] and

OpenFlow in wireless mesh environments [35], [37], [79]. **Information-Centric Networking (ICN)**: is a new paradigm proposed for the future architecture of the Internet, which aims to increase the efficiency of content delivery and content availability. This new concept has been popularized recently by a number of architecture proposals, such as Content-Centric Networking (CCN) [80], also known as the Named Data Networking (NDN) project [56]. A number of projects [26], [89], [91], [97] have proposed using SDN concepts to realize ICNs. As OpenFlow expands to support customized header matchings, SDN can be employed as key enabling technology for ICNs.

6) Additional Significant Research Works: Below, we enlist some of the additional research works that highlight the different applications of SDN.

- Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks [175]. In this work authors propose a Software-Defined wireless sensor networks architecture and address key technical challenges for its core component, Sensor OpenFlow.
- Software-defined Networking based capacity sharing in hybrid networks [176]. This work proposes a novel approach to capacity sharing in hybrid networked environments, i.e., environments that consist of infrastructure-based as well as infrastructure- less networks.
- A Content Management Layer for Software-Defined Information Centric Networks [177]. Authors propose an extension of the SDN abstractions to allow content centric functionality natively.
- OpenFlow Random Host Mutation: Transparent Moving Target Defense using Software Defined Networking [178]. In this work, Authors use OpenFlow to develop a moving target defense (MTD) architecture that transparently mutates IP addresses with high unpredictability and rate, while maintaining configuration integrity and minimizing operation overhead.
- Data Centers as Software Defined Networks: Traffic Redundancy Elimination with Wireless Cards at Routers [179]. Authors propose a novel architecture of data center networks (DCN), which adds wireless network card to both servers and routers, to address the problem of traffic redundancy elimination.

III. Network Virtualization

As mentioned by Carapinha et. al., [223] virtualization, in general, provides an abstraction between user and physical resources, so that the user gets the illusion of direct interaction with those physical resources. Network virtualization technology is a key component that not only is an integral part of the overall design to support the evolution and coexistence of different network architectures but also acts as an abstraction layer between services and infrastructure to facilitate innovation [224].

Some researchers [174] caution that a precise definition of network virtualization is elusive, and some disagree as to whether some of the mechanisms such as slicing represent forms of network virtualization. Accordingly, they define the scope of network virtualization to include any technology that facilitates hosting a virtual network on an underlying physical network infrastructure. However, there have been some well agreed definitions on what a virtualized networks should be. A virtualized network may include overlays, tunnels, virtual devices, and multitenancy. But, it must provide total physical, location and stateful independence. That is, a virtualized network is a faithful and accurate reproduction of the physical network that is fully isolated and provides both location independence and physical network state independence [225].

The concept of network virtualization is not new, and the concept has been realized in the past in the form of VLANs and VPNs, which have been a highly successful approach to provide separate virtual networks over a common physical infrastructure. A complete study on network virtualization would require a separate survey [226]–[229], and below we just summarize in few sentences. The detailed survey of the evolution of Network virtualization and programmable networks can be found in [173].

VPNs fulfill the basic goal of providing different logical networks over a shared infrastructure. However, it suffers from a few limitations, such as (a) all virtual networks are based on the same technology and protocol stack (b) lack of true isolation of virtual network resources (c) lack of clean separation of the roles of infrastructure provider and VPN service provider (d) only network administrators could create these virtual networks (e) incrementally deploying new technologies is difficult. To overcome some of these challenges, the researchers and practitioners resorted to running overlay networks, where a small set of 'enhanced' nodes use tunnels to form their own topology on top of a legacy network [173]. In an overlay network, the 'enhanced' nodes run their own control-plane protocol, and direct data traffic (and control-plane messages) to each other by encapsulating packets, sending them through the legacy network, and stripping the encapsulation at the other end.

Network virtualization goes a step further (of what VPNs achieve) by enabling independent programmability of virtual networks [223]. So, a virtual network is no longer necessarily be based on IP, or any other specific technology, and any kind of network architecture can in principle be built over a virtual network. Another important strength of virtualization is the native capability to handle multi-provider scenarios and hide the specificities of the network infrastructure, including the existence of multiple administrative domains. Although some workaround solutions exist, this has traditionally been a complicated issue for VPNs. Finally, network virtualization aims to provide true isolation (instead of just an illusory isolation, as provided by VPNs) of virtual networks - sharing the same infrastructure - by employing various approaches such as using different operating system instances.

A. SDN and Network virtualization

As argued by Martin Cassado of VMWare [230], Network virtualization and SDN are two different things and somewhat incomparable. SDN is a mechanism (that is relevant to system builders) and network virtualization is a solution. He goes on to give the software-development analogy: “A programming language would be the SDN using which the program(s) can be built out of it. Whereas, network virtualization, on the other hand, is a product category or solution set that customers use to change the paradigm of their network”. That is, just as server virtualization changed the paradigm for server operations and management, network virtualization changed the paradigm for network operations and management. So, one can use SDN in NV but it is not mandatory. The researchers, apart from Martin Cassado, have also mentioned that SDN can be applied to many problems: graphic engineering, security, policy or network virtualization. However, SDN in NV provides the possibility of deep programmability of network infrastructures for quickly modifying network behavior and providing more sophisticated policy controls through rich applications.

Other researchers [174] argue that although network virtualization has gained prominence as a use case for SDN, the concept predates modern-day SDN and has in fact evolved in parallel with programmable networking. The two technologies (SDN and NV) are in fact tightly coupled: Programmable networks often presumed mechanisms for sharing the infrastructure (across multiple tenants in a data center, administrative groups in a campus, or experiments in an experimental facility) and supporting logical network topologies that differ from the physical network, both of which are central tenets of network virtualization.

Apart from SDN as an enabling technology for network virtualization in which Cloud providers need a way to allow multiple customers (or tenants) to share the same network infrastructure, the researchers have pointed additional two ways in which SDN and NV can relate to each-other. These additional two ways are (1) evaluating SDN control applications in a virtualized environments before deploying on operational network (ex: MiniNet) (2) Virtualizing an SDN network (ex: FlowVisor)

In this remaining part we will focus mostly on the network virtualization in cloud environment (Data Centers). In this context, we will broadly classify network virtualization into two broad approaches - hop-by-hop and distributed edge overlays [189].

B. Distributed Edge Overlays and Hop-by-Hop Network Virtualization in multitenant environments

Over past few years, data Centers are increasingly being consolidated and outsourced in an effort to (a)improve the deployment time of applications and (b) reduce the operational costs [191]. This aspect of consolidation also coincides with an increasing demand for compute, storage, and network resources from complex applications. In order to scale compute, storage, and network resources, physical resources are being ’abstracted’ from their logical representation - which is typically referred to as server, storage, and network virtualization. As discussed ealier, virtualization is a broad term, and can be implemented in various layers of computer systems or networks.

Though Oxford Dictionary defined tenant as a “person who occupies land or property rented from a landlord”, in the context of information-technology it refers to any client-organization. Whereas, Multitenancy, according to wikipedia [232], refers to a principle in software architecture where a single instance of the software runs on a server, serving multiple client-organizations (tenants). Multi-tenant data centers [231] are ones where individual tenants could belong to a different company (in the case of a public provider) or a different department (in the case of an internal company data center). Each tenant has the expectation of a level of security and privacy separating their resources from those of other tenants.

To a tenant, the ’virtual’ data centers are similar to their physical counterparts - consisting of end stations attached to a network, and complete with services. Such services can range from load balancers to firewalls [191]. However, unlike a physical data center, end stations connect to a virtual network - that is, to end stations, a virtual network looks like a normal network (e.g., providing an Ethernet or L3 service). The end stations connected to the virtual network are those belonging to a tenant’s specific virtual network. The tenant, customer that has defined and is using/managing a particular virtual network and its associated services, can also create multiple, and different, virtual network instances.

A key multitenancy requirement is traffic isolation, so that a tenant’s traffic is not visible to any other tenant. This isolation can be achieved by assigning one or more virtual networks to each tenant such that traffic within a virtual network is isolated from traffic in other virtual networks.

The question is how is the network sliced into multiple virtual networks? The answer would be either use ’hop-by-hop’ virtualization [189] or use an overlay [188]. These two, considering routing/forwarding, can also be seen as the two schemes/methods for using OpenFlow [194].

1) Hop-by-Hop virtualization: VLANs are an example of hop-by-hop virtualization. In this approach, there is one VLAN per tenant and each switch in the network must be aware of the VLANs and the media access control (MAC) addresses of multiple tenants. The problems with this approach are numerous, such as: (a) explosion of the forwarding state on the aggregation switches (b) the necessity to reconfigure physical switches every time a tenant or virtual machine is added (c) the complexity and slowness of VLAN configuration (d) the complexity of VLAN pruning to avoid unnecessary flooding, stability, and (e)convergence problems of Layer 2 protocols, etc. Hop-by-hop OpenFlow is also an example of hop-by-hop virtualization. This is the approach where a centralized controller creates a flow path from one edge of the network to the other edge of the network using OpenFlow to install a flow on each switch in the path, including the aggregation or core switches. Typically, this approach uses a “reactive controller” where the first packet of each new flow is sent to a centralized SDN controller that applies policy, computes the path, and uses OpenFlow to install a flow into each switch on the path. Unfortunately, the hop-by-hop OpenFlow approach suffers from most of the same problems as VLANs and introduces some new ones as well [188]. There were some companies using the reactive hop-by-hop approach but most of them have since evolved their position. *The commercial product that uses hop-by-hop approach is the NEC’s Programmable-flow [148].*

The hop-by-hop approach, like the NEC's programmable flow, has some important advantages. This approach can achieve the major goals of network virtualization, such as being end-to-end and abstracting and pooling network resources in a manner similar to how server virtualization abstracts and pools compute resources. This provides it the strength to create tenant-specific virtual networks (that enables complete isolation between each tenant) whose topology is decoupled from the topology of the underlying physical network, and dynamically create policy-based virtual networks to meet a wide range of requirements. Another advantage of hop-by-hop approach is the higher network utilization [194].

2) Distributed Edge Overlays: Distributed edge overlays (DEO) have gained a lot of importance as a mechanism for network virtualization among many vendors such as Cisco, Nicira, Microsoft, Midokura, IBM, Juniper, Nuage Networks and Plumgrid. The basic idea behind overlays is to use tunneling (generally L2 in L3) to create an overlay network from the edge that exposes a virtual view of one or more network to end hosts. The edge may be, for example, a vswitch in the hypervisor, or the first hop physical switch. DEO solutions can be roughly decomposed into three independent components - encapsulation format (tunnelling protocol), the control plane and the logical view (defines the network services available to the virtual machine) [221], [222].

As described by Narten et. al [191], and reproduced below, Overlays are based on what is commonly known as a "map-and-encap" architecture, that involves three distinct and logically separable steps:

- The first-hop overlay device implements a mapping operation that determines where the encapsulated packet should be sent to reach its intended destination VM. Specifically, the mapping function maps the destination address (either L2 or L3) of a packet received from a VM into the corresponding destination address of the egress device. The destination address will be the underlay address of the device doing the decapsulation and is an IP address.
- Once the mapping has been determined, the ingress overlay device encapsulates the received packet within an overlay header.
- The final step is to actually forward the (now encapsulated) packet to its destination. The packet is forwarded by the underlay (i.e., the IP network) based entirely on its outer address. Upon receipt at the destination, the egress overlay device decapsulates the original packet and delivers it to the intended recipient VM.

The core idea of an overlay network is that some form of encapsulation, or indirection, is used to decouple a network service from the underlying infrastructure. Per-service state is restricted at the edge of the network and the underlying physical infrastructure of the core network has no or little visibility of the actual services offered. This layering approach enables the core network to scale and evolve independently of the offered services. The overlay approach was pioneered by software vendors such as VMware (Nicira) and Microsoft. But it is now widely supported by major networking vendors as well.

The overlay approach has many advantages, including [189]:

- It reduces the size of the forwarding/flow tables in the physical (also termed underlay) switches. The reason behind this is; as only addresses of physical servers and not addresses of virtual machines are maintained.
- It reduces lot of overhead when it comes to managing multi-tenants. As, adding a new tenant, which may translate to adding a new virtual machine, or applying a new policy only involves modifications on the 'edge' switches (edge switches are typically virtual switches or virtual routers in the hypervisors), and not the physical switches in the underlay.
- It provides a seamless migration path for introducing SDN into existing networks without disrupting existing services provided on those networks.

Overlay approach, which involves encapsulation techniques, are not without drawbacks [189], [221], [222]. The drawbacks include (a) Overheads: encapsulation overhead of the frame size and processing overhead on the server from lack of ability to use NIC offload functionality (b) complications with load-balancing and end-to-end Quality of service (c) Interoperability issues with devices such as firewalls (d) Difficulty to troubleshooting the network

Juniper Networks' White-paper [189] suggests that "reactive hop-by-hop networks and proactive overlay networks are two extremes of a continuous spectrum. Between these two extremes, there are intermediate steps". The methods to make hop-by-hop reactive networks more scalable (using coarser grained flows in the aggregation switches, using a proactive model, or using tunnels) are simply a gradual transition from the reactive hop-by-hop model to the proactive overlay model [189]. However, the decision on which approach to adopt may purely depend the needs and challenges of the end-customer.

3) SDN and Network Virtualization: Some Existing Open Solutions: There are different SDN based Network Virtualization solutions such as FlowVisor [87], FlowN [40], and Autoslice [27]. Below we describe the first two solutions proposed to support NV in SDN environments.

FlowVisor: A network virtualization layer which works as a proxy between switch and a guest controller. Acts as a special purpose controller which sits between switches and and controllers. Slice-Configuration in FlowVisor involves creation of slices using any combination of switches ports, ethernet address, IP address or TCP/UDP port, where each slice is controlled by different controller. Flowvisor also enforce a strong isolation between slices in terms of bandwidth, topology, switch CPU, flowspace and flowtable entries. FlowVisor slices network on five dimensions: Bandwidth, Topology, Traffic, Device CPU and Forwarding Tables.

FlowN [40]: An efficient and scalable virtualization solution which has been built as an extension to the NOX openflow controller. FlowN, instead of slices, uses database in order to do the efficient storing and manipulation of mapping. It gives freedom to each tenant for designing the controller based on their own need and also provide full virtualization. The virtual topologies are decoupled from the physical infrastructure and mapping of physical to virtual are not exposed to tenants. It also enables virtual address space and bandwidth isolation.

FlowVisor provides better performance for small number of virtual networks while FlowN provides better performance for virtual networks more than 100. FlowVisor does not use any databses for storing mapping of virtual and physical network data, while it uses slice interception and configuration for deciding packet path over network. FlowN, on the other hand, uses database mechanism to store mapping between virtual and physical entities which leads to better performance for FlowN over large number of virtual networks.

IV. Network Functions Virtualization

Proprietary appliances that are too diverse, and ever growing in numbers, make the operation of service additions and upgrades increasingly difficult. Typically, these appliances are turn-key in-line systems, that maintain real-time state of subscriber mobility, voice and media calls, security and contextual content management [185]. Network Functions Virtualization (NFV) is an initiative of the ETSI Industry Specification Group to virtualize network functions previously performed by these proprietary dedicated hardware [181], [238].

The white paper on NFV [180] defines NFV as a consolidation of Network functions onto industry-standard servers, switches and storage hardware located in Data/Distribution centers - an optimized data plane under virtualization. Network functions virtualization (NFV) allows administrators to replace physical network devices (traditional) with software that is running on commodity servers. This software realizes the 'network functions' that were previously provided by the dedicated hardware (network devices).

Network Functions Virtualization is about implementing network functions in software - that run today on proprietary hardware - leveraging (high volume) standard servers and IT virtualization. To understand the significance of NFV, we should consider the recent trends. The trends include increased user mobility, explosion of devices and traffic, emergence and growth of cloud services, convergence of computing, storage and networks and finally new virtualization technologies that abstract underlying hardware yielding elasticity, scalability and automation. Accordingly, challenges and issues with respect to these trends include - (a) huge capital investment (b) operators facing increasing disparity between costs and revenues, (c) increasing complexity -large and increasing variety of proprietary hardware appliances in operator's network, (d) reduced hardware lifecycle (e) lack of flexibility and agility: cannot move network resources where and when needed (f) launching new services is difficult and takes too long, and often requires yet another proprietary box which needs to be integrated into existing infrastructure. *The major advantage of using NFV is to address the above challenges and issues.* As highlighted by Yamazaki et. al [197], NFV helps to reduce network operator CAPEX and OPEX through reduced equipment costs and reduced power consumption, and also helps to reduce complexity and make managing a network and deploying new capabilities easier and faster.

As mentioned by Manzalini [196], there are quite a lot of middle-boxes deployed in current networks: not only these nodes are contributing to the so-called "network ossification", but also they represent a significant part of the network capital and operational expenses (e.g., due to the management effort that they require). Basically, a middle-box is a stateful system supporting a narrow specialized network functions (e.g., layer 4 to 7) and it is based on purpose-built hardware (typically closed and expensive). *The next significant advantage of using NFV is in removing, or even reducing, the number of middle-boxes deployed in current networks, which would realize several advantages such as cost savings and increased flexibility.*

NFV also supports multi-versioning and multi-tenancy of network functions, and allows use of a single physical platform for different applications, users and tenants [193]. As described by Yun Chao [195] NFV enables new ways to implement resilience, service assurance, test and diagnostics and security surveillance. It facilitates innovation towards new network functions and services that are only practical in a pure software network environment. We should note that NFV is applicable to any data plane and control plane functions -fixed or mobile networks, and also the automation of management and configuration of functions is very important for NFV to achieve scalability. Ultimately, NFV aims to transform the way network operators architect and operate their networks [195].

A. Functions under NFV and the Use-Cases

The network devices that the commodity server and the software aim to replace can range from firewalls and VPN gateways to switches and routers. Researchers [198] argue that almost any network function can be virtualized. The NFV focus in the market today includes switching elements, network appliances, network services and applications. Considering the description in the white-paper [180], the below list summarizes various network functions that are considered for NFV [199].

- Switching elements such as Broadband remote access server (BRAS) or Broadband Network Gateway (BNG), carrier grade NAT, and routers.
- Mobile network nodes such as Home Location Register/Home Subscriber Server (HLR/HSS), Serving GPRS Support NodeMobility Management Entity (SGSN-MME), Gateway GPRS support node/Packet Data Network Gateway (GGSN/PDN-GW), RNC, NodeB and Evolved Node B (eNodeB).
- Functions in home routers and set top boxes
- Virtualized home environments.
- Tunneling gateway elements such as IPSec/SSL virtual private network gateways.
- Traffic analysis elements such as Deep Packet Inspection (DPI), Quality of Experience (QoE) measurement.
- Service Assurance, Service Level Agreement (SLA) monitoring, Test and Diagnostics.
- Next-Generation Networks (NGN) signaling such as Session Border Controller (SBCs), IP Multimedia Subsystem (IMS).
- Converged and network-wide functions such as AAA servers, policy control and charging platforms.
- Application-level optimization including Content delivery network (CDNs), Cache Servers, Load Balancers, Application Accelerators.

- Security functions such as Firewalls, virus scanners, intrusion detection systems, spam protection

Considering the above listing, the use-cases of NFV can cover virtualization of - mobile core/edge network nodes, home environment, content delivery networks, mobile base Station, wireless LAN controllers, customer premise equipments (CPE) and operation services systems (OSS).

1) Example Deployments: Below, we enlist three practical deployments of NFV by major Telecom vendors.

- British telecom, in partnership with HP, Intel, wind-river and tail-f, developed as proof of concept, a combined BRAS and CDN functions on Intel Xeon Processor 5600 Series HP c7000 BladeSystem using Intel 82599 10 Gigabit Ethernet Controller sidecars [200]. BRAS is chosen as an 'acid test', whereas CDN is chosen as it architecturally complements BRAS.
- Move access network functions to the Data Center: Deutche Telecom recently launched the 'Terastream' project, which simplify IP access networks with NFV and SDN [201], [202].
- Reduce OPEX by simplifying home networks: France Telecom's proposal for virtual home gateway to Broadband Forum. This proposal aims to increase flexibility in service deployment at the edge. The project is supported by Telefonica, Portugal Telecom and China Telecom [203]

B. NFV: Realization requirements

Efficient realization of virtualized network functions should consider various aspects such as flexible processing, good performance with respect to handling millions of packets, efficient isolation mechanisms, flow migration and per-flow state to support mobility and finally support for multiple domains. Hence, when planning for the network functions virtualization, the administrator has to address the following issues.

- Network functions to virtualize - considering the return of investment.
- Approach to integrate with the existing network management infrastructure - interoperability.
- Scalability and elasticity issues.
- Resource management approach, and APIs to expose for ease of management.
- Availability and service failover/recovery mechanisms.

With the success of NFV, next generation network functions will be implemented as pure software instances running on standard servers - unbundled virtualized components of capacity and functionality. One of the existing approaches, as explained by Barkai et. al. [185] to realize such a network functions is to divide the whole process into two-steps - as described below.

- Significant component-based unbundling: Unbundling refers to breaking up the packages that once offered as a single unit, providing particular parts of them at a scale and cost unmatchable by the original package provider [204]. In this case, it is the unbundling of both capacity and functionality locked in monolithic systems. In this first step, the monolithic systems are broken to components that will be running on virtual machines.
- Dynamically assemble discrete functional components to elastic end-to-end services. In this second step, the broken down components are assembled using various techniques such as flow-mapping [185].

One should note that such "scatter-gather" rearrangement of carrier functionality needs to work on commodity hardware. In addition, realization of NFV may also pose the challenge of assembling components developed by third-parties, into whole solutions.

1) Requirements from the VM perspective: Researchers [205] argue that the traditional virtual machines VMs may not be able to address the challenges of NFV. Traditional VMs are Fat - with huge overheads, poor performance, and limited to only few VMs per server. Whereas, the VMs for NFV needs to be "Small" - with minimal overhead, excellent performance, many number of VMs per server, and natively supports network functions. One of the most popular VMs designed for NFV is ClickOS [205]. ClickOS is both light-weight (just around 1.4MB) and scalable. One can run many instances (in multiple thousands) on a single commodity PC. ClickOS also boots and migrates quickly, which can be used to realize the following network functions: Broadband Remote Access Server (BRAS), residential gateway, deep packet inspection, caching solutions, and enterprise firewalls.

C. NFV as a Use-Case of SDN

NFV provides immense possibilities for the providers to take advantage of the full potential of their network infrastructures, and to offer novel commercial services to their customers. Software Defined networking (SDN) is not a requirement for NFV, but the two technologies are complementary - Intel terms NFV as a complementary initiative to SDN. Administrators/Engineers can implement NFV, choosing to rely on traditional networking algorithms such as spanning tree or IGRP instead of moving to an SDN architecture. Yet, SDN can improve performance and simplify operations in a network functions virtualization environment.

Jim Machi, in his report [206], discusses elaborately this relation between SDN and NFV. He begins with mentioning that both technologies are designed to increase flexibility, decrease costs, support scalability, and speed the introduction of new services. In addition, Both SDN and NFV are likely drivers for innovation in telecommunications, networking and enterprise data centers. Finally, both owe their existence to similar market forces, including: (a)Better processor capability - significant improvement in the processor technology (b) Simplification in connectivity - scope for separation of planes. (c) Virtualization maturity.

While SDN was originally conceived to control the operation of network hardware devices, researchers have shown that it can just as easily integrate into an NFV environment, communicating with software-based components on commodity servers. The first white paper on NFV suggests that NFV and SDN have some overlap, but neither SDN is a subset of

NFV, nor the other way around. So, the important questions to address are, Where do SDN and NFV intersect ? and, How will the interaction between SDN and NFV impact the evolution of both ideas? These are the interesting and challenging questions that are, and will be, addressed by various researchers and vendors, in different ways in future.

In general, as long as NFV addresses the general case of 'policy-managed' forwarding, and need dynamic service orchestration. SDN can play a role in realization of the same - i.e., NFV can increase network efficiency by using SDN concepts. NFV may demand virtual network overlays - the model of tunnels and vSwitches, would segregate virtual functions to prevent accidental or malicious interaction. This use of network overlays, for virtual function segregation, in NFV will drive the need for applying SDN based solutions. The multi-tenancy requirement would also influence NFV to adopt a software-overlay network model. In addition, if NFV allows services to be composed of virtual functions hosted in different data centers, that would require virtual networks to stretch across data centers and become end-to-end. An end-to-end virtual network would be far more interesting to enterprises than one limited to the data center. SDN will play a crucial role in such an extended NFV realization.

Hence, SDN when applied to NFV can help in addressing the challenges of dynamic resource management and intelligent service orchestration. When using SDN-based approach for NFV, there are lots of aspects that need to be looked such as:

- To seamlessly integrate the virtualized network functions (VNF) into the existing SDN and cloud architectures - importantly, from the perspective of multi-tenancy.
- To ensure that the considered SDN controller that has all the necessary features the architecture needs. For example, flow-mapping service may or may-not exist as an in-built feature of the controller.
- To ensure if the scalability aspect of the existing SDN-architecture is sufficient to support potentially hundreds of millions of subscriber flows and their 'affinity-associations' [185] to virtualization function instances.
- To extend the existing SDN control model - decoupling of not just control and forwarding, but also control and forwarding topologies [185]

Different SDN solution vendors address some or all of these challenges in different ways. A typical approach would be include necessary network services at the control-tier and provide robust, efficient north-bound abstractions (APIs), apart from having an eco-system including application developers. OpenDaylight [109] is one of an excellent examples to follow.

V. Current Trends and Future Directions

SDN is moving towards next level of standardization and various companies are bringing out their products in the market. The vendors who have their own SDN controllers are moving towards setting up an '*ecosystem*'. The SDN ecosystem is where all the stake-holders - Asic Vendors, Switch vendors, Controller vendors and Network application services vendors - join hands to either achieve interoperability or provide complete suite of solution. The success of the open-source controller platforms like Opendaylight, with well-defined North-bound APIs, has fueled the trend of controlling SDN networks smoothly through applications. These applications are developed targeting various domains and network services. In this section, we describe few domains that are expected to dominate the SDN scenario over next few years. The authors of [173] also provide a few domains to look out for in future, that are different from the ones mentioned below. We also mention some interesting problems to focus in future, following the current trends.

A. Data Centers of Network Service Providers: Telecom Cloud and NFV

UNIFY FP7 project [233] co-ordinator Andrs Csszr, states that "Telecommunication providers struggle with low service flexibility, increasing complexity and related costs". He continues to argues that, although "cloud" has been an active field of research over the past few years, currently there has been little integration between the vast networking assets and data centres of telecom providers. Hence, it has turned out to be a crucial aspect for network service providers to be able to offer advanced services (along with connectivity), and to be able to optimize network economics by improving operations. In addition, it is also important to take advantage of the full flexibility and capabilities offered by networking equipment in order to introduce new services, along with improving the existing ones, towards better customer experience and reduced operational expenses (OpEx).

Network service providers (NSP), either already own or are building data centers at some of their sites [234]–[237]. Data centers, that are the enablers for cloud services, are currently the only well programmable part of the infrastructure. NSPs' DCs not only allow offering cloud services to customers (enterprise as well as residential) but also provide an opportunity to concentrate on existing and new telco functions, further increasing their utilization, and decreasing costs compared to monolithic, vendor dependent and/or purpose oriented gear [235]. So, "Telco Cloud" is also about virtualising networking and telecom functions and moving these to the service provider's cloud infrastructure [236]. In addition, the Telco Cloud will be offered to other service providers allowing the deployment and advertisement of own services, leading to new revenues for NSPs [233]. Volker Held, head of Nokia Solutions and Networks (NSN), highlights Five key 'ingredients' for "telco-cloud" [239] - meeting signaling and latency demands, avoiding vendor-locking, maintaining scalability and flexibility via orchestration and application management, integration into the FCAPS (Fault, Configuration, Accounting, Performance, Security)system, business process overhaul.

We argue that SDN will have a major role to play in such telecommunication cloud, especially in addressing different key 'ingredients' mentioned above. Some of the key strengths of SDN that may play a significant role in such scenarios are [233]: (a) SDN enables a new control architecture and operation practices with fine granular control over services (b) SDN gets rid of monolithic nodes and enables the introduction of new features and services via a (logically) centralized controller. Some of the specific advantages of SDN with respect to telco-cloud, according to the various researchers [233], [240], [241], can be summarized as follows. The fine granular control of traffic flows, enabled by SDN, allows NSPs to offer several value added functions (such as firewalls, parental control, etc.) in addition to connectivity services. These value added functions may be offered in flexible bundles and may be personalised to subscribers. Technically, the traffic flows are passing a service chain, i.e. a series of advanced service functions (ASFs). SDN allows fine granular control of traffic steering between ASFs, and, indeed, service chaining became one of the most promising applications of SDN. While these existing technologies enable flexible placement of service components and fine granular traffic steering between these components, NSPs still do not possess means to optimally orchestrate the placement of service functions in their networks, which opens up immense scope for interesting future works.

B. Radio Access Networks

An important part of the cellular network infrastructure is the radio access network (RAN) that provides wide-area wireless connectivity to mobile devices. The fundamental problem that RAN addresses is how best to use and manage limited spectrum to achieve the connectivity with mobile devices. In a dense wireless deployments, becomes a difficult task to allocate radio resources, implement handovers, manage interference, balance load between cells [213]. Applying SDN to RANs, has been an interesting and a growing trend over past few years. Aditya et. al, [213] propose SoftRAN - software defined centralized control plane for radio access networks that abstracts all base stations in a local geographical area as a virtual big-base station comprised of a central controller and radio elements. The SoftRAN proposal is based on the argument that LTEs current distributed control plane is sub-optimal in achieving the necessary objectives. Following this trend, it would be very interesting to focus on SDN in LTE environments. For example, the ability to automate the management processes has emerged as a key technology requirement. Mainly the requirement to collect and analyze the quality measurement data from the base station and mobile terminals, and correspondingly change the settings of the base station would be an interesting problem to address. Such works may act as a classic use-case of using concepts of self-configuring and self-optimizing wireless networks in LTE deployments. The concept of Self-organizing (SON) in wireless networks is not new, and this work [214] summarizes the benefits of SON in wireless backhaul networks. An interesting problem that could be addressed based on SDN concept would be an approach realizing various scheduling mechanisms - managing uplink and downlink flows - on enodeB.

C. Optical Transport Networks - Interconnecting Data Centers of Cloud Providers and Enterprises

ITU-T defines an Optical Transport Network (OTN) as a set of Optical Network Elements (ONE) connected by optical fiber links, able to provide functionality of transport, multiplexing, switching, management, supervision and survivability of optical channels carrying client signals [212]. Figure 4 depicts a possible scenario of data-center interconnect via WANs, in which both the data-center network and the WAN are based on SDN technology. This figure also highlights that individual SDN-controllers may in-turn be managed by the a higher-order controller or cloud-orchestration tools.

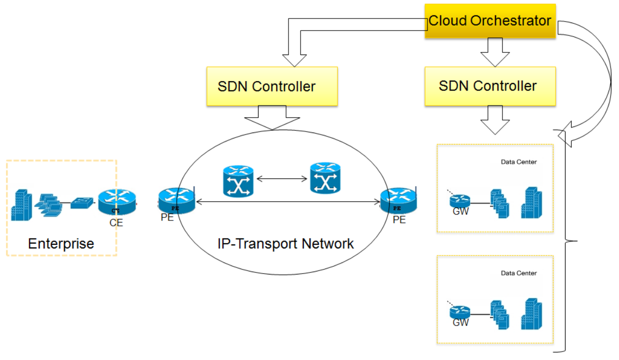

Fig. 4. SDN in Transport Network Interconnecting Data-Centers

Cloud providers may not own network infrastructure and count on network providers to interconnect distributed DCs - by Wide Area Network (WAN). Examples include the alliance of IBM SmartCloud and Microsoft Azure with AT&T virtual private networking for global cloud services and for providing a more secure and reliable connectivity for enterprise customers, respectively.

Data center WAN interconnects today are pre-allocated, static optical trunks of high capacity. In other words, current inter-data-center connections are configured as static big fat pipes, which entails large bit rate over-provisioning and thus high operational costs for DC operators. These optical pipes carry aggregated packet traffic originating from within the data centers while routing decisions are made by devices at the data center edges. There has been a lot of work on SDN enabled transport architecture (Transport SDN) that meshes seamlessly with the deployment of SDN within the Data Centers [182]. Such programmable architecture abstracts a core transport node into a programmable virtual switch that leverages the OpenFlow protocol for control [183] [184]. Hence, by including the wide area network (WAN) as an intrinsic component of cloud-based services, network operators may have the necessary tools to realize the complete potential of the WAN infrastructure decoupling todays clouds from their physical data center "anchors" and opening them up for innovation [211].

The transport network, typically consists of Wavelength-Switched Optical Network (WSON) network. OTN, includes set of standards, which allow interoperability and the generic transport of any protocol across an optical network. The implementation is based on the "wrapper" approach - for example, IP packets are wrapped with optical-communication specific headers. An IP/MPLS network design on top of such networks, may create label switch paths (LSC) based on optical transmission aspects too. In such designs, there exists various interesting problems to address. For example, a joint **(multi-domain) path computation process**, based on existing PCE (path computation element) framework, may be essential [210]. In addition, lot of challenges exist in managing and in-turn **translating the failures at the optical network level to the particular tenant's network level**. *So, future works focusing on supporting a Packet and Optical transport network that evolves from existing integrated control and data plane routers towards SDN where SDN-controller(s) provide the forwarding rules, would be extremely useful.*

D. Optical Networks and Multi-domain Management within data-centers

Recently Mayur et. al [209], highlighted that Extending SDN to support interconnectivity of IT resources, such as virtual computing [virtual machines (VMs)] and storage using emerging optical transport [4] and switching technologies (e.g., elastic optical networks), as well as existing packet networks, will enable application-aware/ service-aware traffic flow handling and routing within data-centers. An SDN-based solution, with the concept of separation of a data plane and control plane, to address the challenges in optical-network connectivity, can lead to lower operating expenditures and more efficient networks. SDN-based unified control and management of an optical network poses various challenges such as [209]:

- Design and implementation of an abstraction mechanism that can hide the heterogeneous optical transport layer technology details and realize the aforementioned generalized switching entity definition.
- Cross technology constraints for bandwidth allocation and traffic mapping in networks comprising of heterogeneous technological domains, e.g., packet over single or hybrid optical transport technologies.
- Definition of a unified optical transport and switching granularity (i.e., optical flow) that can be generalized for different optical transport technologies (fixed DWDM, flexi DWDM, etc.) and be compatible with electronic packet switching technology
- Taking into account physical layer specific features of different optical transport technologies, such as power, impairments, and switching constraints.

Hence, it may prove to be a challenging future work that focuses on addressing the aforementioned challenges.

E. Integrating cloud orchestration tools, SDN/Openflow Controllers and Switches for NV and NFV

Over the past few years, the trend of integrating SDN/Openflow Controllers and switches with Cloud orchestration tools to achieve various goals has seen an exponential growth. Below, we enlist some important related works.

- Openstack Neutron Plugins: Neutron is an OpenStack project to provide "networking as a service" between interface devices (e.g., vNICs) managed by other Openstack services (e.g., nova) [217], [218]. Neutron lets one to use a set of different backends called "plugins" that work with a growing variety of networking technologies. These plugins may be distributed as part of the main Neutron release, or separately. Various vendors have developed plugins to integrate their products with Openstack [218]. This trend of integrating with openstack either via drivers or plugins is ever increasing, and some researchers are even focusing on enhancing the Neutron APIs.
- CloudNFV [215]: CloudNFV is a multi-vendor consortium, which mainly aims to build an unified data model that incorporates data and policies of services and resources, in addition to orchestration process. It aims to decouple the creation of management information from the way it is presented. For example, the architecture can present information in a simple network management protocol, SNMP form, regardless of whether it originated in SNMP form.
- FARO (Future Architecture for Resources Orchestration) [216]. Authors argue that orchestrating involve 2 classes of challenges; Technical aspects - embed services and network functions into physical/virtual infrastructures and Economical aspects - trade-off between automating SLA and price negotiation. FARO focuses on orchestrating various services such as - NFV, device-to-device communication, collaborative telecommunication and content services, and federation of networks.
- Ericsson Real-time Cloud [211]: This work relies on combing cloud, NFV and service provider SDN. Together, these technologies transform the network and the cloud into a Network-enabled-Cloud one that is more fluid, more dynamic and more responsive to emerging service needs. Operators need to ensure that their networks remain a relevant and vital part of users everyday experience, and deliver added value in new and unique ways. Emerging network-enabled Cloud, Network Functions Virtualization and software-defined networking technologies will help operators do just this, by enabling common management and orchestration across network resources and cloud applications.
- Alcatel cloud-band [219]: CloudBand is Alcatel's end-to-end NFV platform. Being open and multivendor, it has been built to support the stringent needs of carriers and speed the move to NFV. By introducing the CloudBand ecosystem program, Alcatel is making CloudBand available to the entire industry for free. The goal, as claimed by Alcatel, is to foster collaboration and experimentation that will accelerate adoption of NFV and create new business opportunities across the industry.

From the above works, it is evident that there is a rise of strong culture of developing an ecosystem of system integrators, application developers, original equipment manufactures and service providers. In such a scenario, it would be very interesting to focus on the 'interfacing-components', for example northbound or southbound interfaces of SDN controllers, in developing novel solutions. There will be a strong need for architects to visualize end-to-end solutions in addressing various challenges of the end-customers or in developing new network services.

VI. Conclusions

SDN, is about applying modularity to network control, which gives the network designer the freedom to re-factor the control plane. This modularity has found its application in various areas including network virtualization. In this work, we presented a thorough study of SDN and how SDN technology can complement the network virtualization and network functions virtualization. We began the survey with a history of programmable networks in general and SDN in particular. We described the SDN architecture in detail as well as the role of OpenFlow standard. We presented current SDN tools, commercial products and vendors, open-source solutions, applications, use-cases, frameworks and research-works. We also provided detailed survey of two important use-cases of SDN - NV and NFV. We concluded with a discussion of future directions enabled by SDN focusing mainly on cloud and data-centers.

References

[1] Beacon. https://openflow.stanford.edu/display/Beacon/Home.

[2] Connected cloud control: Open flow in mirage. http://www.openmirage.org/blog/announcingmirage openflow.

[3] Controller performance comparisons. http://www.openflow.org/wk/index.php/ Controller Performance Comparisons.

[4] Floodlight, an open sdn controller. http://floodlight.openflowhub.org/.

[5] Helios by nec. http://www.nec.com/.

[6] Indigo: Open source openflow switches. http://www.openflowhub.org/display/Indigo/.

[7] Jaxon:java based openflow controller. http://jaxon.onuos.org/.

[8] Mul. http://sourceforge.net/p/mul/wiki/Home/.

[9] The node flow openflow controller. http://garyberger.net/?p=537.

[10] Node.js. http://nodejs.org/.

[11] ofsoftswitch13 cpqd. https://github.com/CPqD/ofsoftswitch13.

[12] Open networking foundation. https://www.opennetworking.org/about.

[13] Open Networking Research Center (ONRC). http://onrc.net.

[14] Open vswitch and ovs controller. http://openvswitch.org/.

[15] Pantou: Openflow 1.0 for openwrt. http://www.openflow.org/wk/index.php/ OpenFlow 1.0 for OpenWRT.

[16] Pox. http://www.noxrepo.org/pox/about pox/.

[17] Ryu. http://osrg.github.com/ryu/.

[18] Sdn troubleshooting simulator. http://ucb sts.github.com/sts/.

[19] Simple Network Access Control (SNAC). http://www.openflow.org/wp/snac/.

[20] Trema openflow controller framework. https://github.com/trema/trema.

[21] Nicira Networks: Disruptive Network Virtualization Stanford CasePublisher 204-2012-1. 21 May 2012

[22] M. Bansal, J. Mehlman, S. Katti, and P. Levis. Openradio: a programmable wireless dataplane. In Proceedings of the first workshop on Hot topics in software defined networks, pages 109 114. ACM, 2012.

[23] R. Bennesby, P. Fonseca, E. Mota, and A. Passito. An inter as routing component for software defined networks. In Network Operations and Management Symposium (NOMS), 2012 IEEE, pages 138 145, 2012.

[24] D. Alexander, W. Arbaugh, A. Keromytis, and J. Smith. Secure active network environment archtiecture: Realization in SwitchWare. IEEE Network Magazine , pages 3745, May 1998 International Conference on, pages 1-5, May.

[25] R. Bifulco, R. Canonico, M. Brunner, P. Hasselmeyer, and F. Mir. A practical experience in designing an openflow controller. In Software Defined Networking (EWSDN), 2012 European Workshop on, pages 61 66, Oct.

[26] N. Blefari Melazzi, A. Detti, G. Mazza, G. Morabito, S. Salsano, and L. Veltri. An openflow based testbed for information centric networking. Future Network and Mobile Summit, pages 4 6, 2012.

[27] Z. Bozakov and P. Papadimitriou. Autoslice: automated and scalable slicing for software defined networks. In Proceedings of the 2012 ACM conference on CoNEXT student workshop, CoNEXT Student '12, pages 3 4, New York, NY, USA, 2012. ACM.

[28] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a routing control platform. In Proceedings of the 2nd conference on Symposium on Networked Systems Design and Implementation Volume 2, pages 15 28. USENIX Association, 2005.

[29] Z. Cai, A. Cox, and T. Ng. Maestro: A system for scalable openflow control. Technical Report TR10 08, Rice University, December 2010.

[30] K. Calvert, W. Edwards, N. Feamster, R. Grinter, Y. Deng, and X. Zhou. Instrumenting home networks. ACM SIGCOMM Computer Communication Review, 41(1):84-89, 2011.

[31] A. Campbell, I. Katzela, K. Miki, and J. Vicente. Open signaling for atm, internet and mobile networks (opensig'98). ACM SIGCOMM Computer Communication Review, 29(1):97-108, 1999.

[32] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford. A nice way to test openflow applications. NSDI, Apr, 2012.

[33] M. Casado, M. Freedman, J. Pettit, J. Luo, N. Keown, and S. Shenker. Ethane: Taking control of the enterprise. ACM SIGCOMM Computer Communication Review, 37(4):1-12,2007.

[34] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian. Fabric: a retrospective on evolving sdn. In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 85-90, New York, NY, USA, 2012. ACM.

[35] A. Coyle and H. Nguyen. A frequency control algorithm for a mobile adhoc network. In Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, November 2010.

[36] A. Curtis, J. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. Devoflow: Scaling flow management for high performance networks. In ACM SIGCOMM, 2011.

[37] P. Dely, A. Kassler, and N. Bayer. Openflow for wireless mesh networks. In Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN), pages 1 6. IEEE, 2011.

[38] A. Doria, F. Hellstrand, K. Sundell, and T. Worster. General Switch Management Protocol (GSMP) V3. RFC 3292 (Proposed Standard), June 2002.

[39] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern. Forwarding and Control Element Separation (ForCES) Protocol Specification. RFC 5810 (Proposed Standard), Mar. 2010.

[40] D. Drutskoy, E. Keller, and J. Rexford. Scalable network virtualization in software defined networks. Internet Computing, IEEE, PP(99):1 1.

[41] R. Enns. NETCONF Configuration Protocol. RFC 4741 (Proposed Standard), Dec. 2006. Obsoleted by RFC 6241.

[42] N. Feamster. Outsourcing home network security. In Proceedings of the 2010 ACM SIGCOMM workshop on Home networks, pages 37 42. ACM, 2010.

[43] A. Feldmann. Internet clean slate design: what and why? SIGCOMM Comput. Commun. Rev., 37(3):59 64, July 2007.

[44] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. Frenetic: a network programming language. In Proceedings of the 16th ACM SIGPLAN international conference on Functional programming, ICFP '11, pages 279 291, New York, NY, USA, 2011. ACM.

[45] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella. Toward software defined middlebox networking. 2012.

[46] S. Ghorbani and 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, pages 67-72, New York, NY, USA 2012. ACM.

[47] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4d approach to network control and management. ACM SIGCOMM Computer communication Review, 35(5):41-54, 2005.

[48] N. Gude, T. Koponen, J. Pettit, B. Pfa, M. Casado, N. McKeown, and S. Shenker. Nox: towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 38(3):105-110, 2008.

[49] N. Handigol, B. Heller, V. Jeyakumar, D. Mazi eres, and N. McKeown. Where is the debugger for my software defined network? In Proceedings of the first workshop on Hot topics in software de ned networks, HotSDN '12, pages 55-60, New York, NY, USA, 2012. ACM.

[50] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari. Plug n serve: Load balancing web traffic using openflow. ACM SIGCOMM Demo, 2009.

[51] S. Hassas Yeganeh and Y. Ganjali. Kandoo: a framework for ecient and scalable ooading of control applications. In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 19-24, New York, NY, USA, 2012. ACM.

[52] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. Elastictree: Saving energy in data center networks. In Proceedings of the 7th USENIX conference on Networked systems design and implementation, pages 17-17. USENIX Association, 2010.

[53] B. Heller, R. Sherwood, and N. McKeown. The controller placement problem. In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 7-12, New York, NY, USA, 2012. ACM.

[54] T. Henderson, M. Lacage, G. Riley, C. Dowell, and J. Kopena. Network simulations with the ns 3 simulator. SIGCOMM demonstration, 2008.

[55] T. L. Hinrichs, N. S. Gude, M. Casado, J. C. Mitchell, and S. Shenker. Practical declarative network management. In Proceedings of the 1st ACM workshop on Research on enterprise networking, WREN '09, pages 1-10, New York, NY, USA, 2009. ACM.

[56] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking named content. In Proceedings of the 5th international conference on Emerging networking experiments and technologies, pages 1-12. ACM, 2009.

[57] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran Gia. Modeling and performance evaluation of an openflow architecture. In Teletra c Congress (ITC), 2011 23rd International, pages 1-7, Sept.

[58] E. Keller, S. Ghorbani, M. Caesar, and J. Rexford. Live migration of an entire network (and its hosts). In Proceedings of the 11th ACM Workshop on Hot Topics in Networks, HotNets XI, pages 109-114, New York, NY, USA, 2012. ACM.

[59] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey. Veriflow: verifying network wide invariants in real time. In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 49-54, New York, NY, USA, 2012. ACM.

[60] H. Kim and N. Feamster. Improving network management with software defined networking. Communications Magazine, IEEE, 51(2):114-119, February.

[61] E. Kissel, G. Fernandes, M. Jaee, M. Swany, and M. Zhang. Driving software defined networks with xsp. In SDN S12: Workshop on Software Defined Networks, 2012.

[62] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, et al. Onix: A distributed control platform for large scale production networks. OSDI, Oct, 2010.

[63] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: rapid prototyping for software defined networks. In Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks, 2010.

[64] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann. Logically centralized?: state distribution trade os in software defined networks. In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 1-6, New York, NY, USA, 2012. ACM.

[65] L. Li, Z. Mao, and J. Rexford. Toward software defined cellular networks. 2012.

[66] T. A. Limoncelli. Openflow: a radical new idea in networking. Commun. ACM, 55(8):42-47, Aug. 2012.

[67] G. Lu, R. Miao, Y. Xiong, and C. Guo. Using cpu as a traffic co processing unit in commodity switches. In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 31-36, New York, NY, USA, 2012. ACM.

[68] Y. Luo, P. Cascon, E. Murray, and J. Ortega. Accelerating openflow switching with network processors. In Proceedings of the 5th ACM/IEEESymposium on Architectures for Networking and Communications Systems, ANCS '09, pages 70-71, New York, NY, USA, 2009. ACM.

[69] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. B. Godfrey, and S. T. King. Debugging the data plane with anteater. In Proceedings of the ACM SIGCOMM 2011 conference, SIGCOMM '11, pages 290-301, New York, NY, USA, 2011. ACM.

[70] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2):69-74, 2008.

[71] S. Mehdi, J. Khalid, and S. Khayam. Revisiting traffic anomaly detection using software defined networking. In Recent Advances in Intrusion Detection, pages 161-180. Springer, 2011.

[72] J. C. Mogul and P. Congdon. Hey, you darned counters!: get on my asic! In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 25-30, New York, NY, USA, 2012. ACM.

[73] John G. Van Bosse and Fabrizio U. Devetak (2007). Signaling in telecommunication networks (2nd ed.). John Wiley and Sons. p. 111. ISBN 978-0-471-66288-4.

[74] J. Moore and S. Nettles. Towards practical programmable packets. In Proceedings of the 20th Conference on Computer Communications (INFOCOM). Citeseer, 2001.

[75] R. Mortier, T. Rodden, T. Lodge, D. McAuley, C. Rotsos, A. Moore, A. Koliousis, and J. Sventek. Control and understanding: Owning your home network. In Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on, pages 1-10. IEEE, 2012.

[76] A. Nakao. Flare : Open deeply programmable network node architecture. http://netseminar.stanford.edu/10 18 12.html.

[77] M. R. Nascimento, C. E. Rothenberg, M. R. Salvador, C. N. A. Correa, S. C. de Lucena, and M. F. Magalh aes. Virtual routers as a service: the routeflow approach leveraging software defined networks. In Proceedings of the 6th International Conference on Future Internet Technologies, CFI '11, pages 34-37, New York, NY, USA, 2011. ACM.

[78] A. Nayak, A. Reimers, N. Feamster, and R. Clark. Resonance: Dynamic access control for enterprise networks. In Proceedings of the 1st ACM workshop on Research on enterprise networking, pages 11-18. ACM, 2009.

[79] X. Nguyen. Software defined networking in wireless mesh network. Msc. thesis, INRIA, UNSA, August 2012.

[80] A. Passarella. Review: A survey on content centric technologies for the current internet: Cdn and p2p solutions. Comput. Commun., 35(1):1-32, Jan. 2012.

[81] B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker. Software defined Internet architecture: decoupling architecture from infrastructure. In Proceedings of the 11th ACM Workshop on Hot Topics in Networks, HotNets XI, pages 43-48, New York, NY, USA, 2012. ACM.

[82] R. Raghavendra, J. Lobo, and K. W. Lee. Dynamic graph query primitives for sdn based cloudnetwork management. In Proceedings of

the first workshop on Hot topics in software defined networks, HotSDN '12, pages 97-102, New York, NY, USA, 2012. ACM.

[83] The Clean-Slate: An Interdisciplinary Program at Stanford. http://cleanslate.stanford.edu/

[84] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker. Abstraffictions for network update. In Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication, SIGCOMM '12, pages 323-334, New York, NY, USA, 2012. ACM.

[85] J. Rexford, A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, G. Xie, J. Zhan, and H. Zhang. Network wide decision making: Toward a wafer thin control plane. In Proc. HotNets, pages 59-64. Citeseer, 2004.

[86] C. E. Rothenberg, M. R. Nascimento, M. R. Salvador, C. N. A. Correa, S. Cunha de Lucena, and R. Raszuk. Revisiting routing control platforms with the eyes and muscles of software defined networking. In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 13-18, New York, NY, USA, 2012. ACM.

[87] R. Sherwood, M. Chan, A. Covington, G. Gibb, M. Flajslik, N. Handigol, T. Huang, P. Kazemian, M. Kobayashi, J. Naous, et al. Carving research slices out of your production networks with openflow. ACM SIGCOMM Computer Communication Review, 40(1):129-130, 2010.

[88] H. Shirayanagi, H. Yamada, and K. Kono. Honeyguide: A vm migration aware network topology for saving energy consumption in data center networks. In Computers and Communications (ISCC), 2012 IEEE Symposium on, pages 000460-000467. IEEE, 2012.

[89] J. Suh, H. Jung, T. Kwon, and Y. Choi. Chow: Content oriented networking over openflow. In Open Networking Summit, April 2012.

[90] L. Suresh, J. Schulz Zander, R. Merz, A. Feldmann, and T. Vazao. Towards programmable enterprise wlans with odin. In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 115-120, New York, NY, USA, 2012. ACM.

[91] D. Syrivelis, G. Parisis, D. Trossen, P. Flegkas, V. Sourlas, T. Korakis, and L. Tassiulas. Pursuing a software defined information centric network. In Software Defined Networking (EWSDN), 2012 European Workshop on, pages 103-108, Oct 2012.

[92] V. Tanyingyong, M. Hidell, and P. Sjodin. Improving pc based openflow switching performance. In Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS '10, pages 13:1-13:2, New York, NY, USA, 2010. ACM.

[93] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden. A survey of active network research. Communications Magazine, IEEE, 35(1):80-86, 1997.

[94] D. Tennenhouse and D. Wetherall. Towards an active network architecture. In DARPA Active NEtworks Conference and Exposition, 2002. Proceedings, pages 2-15. IEEE, 2002.

[95] A. Tootoonchian and Y. Ganjali. Hyperow: A distributed control plane for openflow. In Proceedings of the 2010 internet network management conference on Research on enterprise networking, pages 3-3. USENIX Association, 2010.

[96] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood. On controller performance in software defined networks. In USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot ICE), 2012.

[97] L. Veltri, G. Morabito, S. Salsano, N. Blefari Melazzi, and A. Detti. Supporting information centric functionality in software defined networks. IEEE ICC Workshop on Software Defined Networks, June 2012.

[98] A. Voellmy and P. Hudak. Nettle: taking the sting out of programming network routers. In Proceedings of the 13th international conference on Practical aspects of declarative languages, PADL'11, pages 235-249, Berlin, Heidelberg, 2011. Springer Verlag.

[99] A. Voellmy, H. Kim, and N. Feamster. Procera: a language for high level reactive network control. In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 43-48, New York, NY, USA, 2012. ACM.

[100] A. Voellmy and J. Wang. Scalable software defined network controllers. In Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication, SIGCOMM '12, pages 289-290, New York, NY, USA, 2012. ACM.

[101] R. Wang, D. Butnariu, and J. Rexford. Openflow based server load balancing gone wild. In Workshop of HotICE, volume 11, 2011.

[102] A. Wundsam, D. Levin, S. Seetharaman, and A. Feldmann. Ofrewind: enabling record and replay troubleshooting for networks. In Proceedings of the 2011 USENIX conference on USENIX annual technical conference, USENIXATC'11, pages 29-29, Berkeley, CA, USA, 2011. USENIX Association.

[103] K. Yap, M. Kobayashi, R. Sherwood, T. Huang, M. Chan, N. Handigol, and N. McKeown. Openroads: Empowering research in mobile networks. ACM SIGCOMM Computer Communication Review, 40(1):125-126, 2010.

[104] K. Yap, R. Sherwood, M. Kobayashi, T. Huang, M. Chan, N. Handigol, N. McKeown, and G. Parulkar. Blueprint for introducing innovation into wireless mobile networks. In Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures, pages 25-32. ACM, 2010.

[105] S. Yeganeh, A. Tootoonchian, and Y. Ganjali. On scalability of software defined networking. Communications Magazine, IEEE, 51(2):136-141, February.

[106] M. Yu, L. Jose, and R. Miao. Software defined traffic measurement with opensketch. In Proceedings 10th USENIX Symposium on Networked Systems Design and Implementation, NSDI'13, 2013.

[107] M. Yu, J. Rexford, M. Freedman, and J. Wang. Scalable flow based networking with difane. In Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM, pages 351-362. ACM, 2010.

[108] Opendaylight : "Open Daylight Website". http://www.opendaylight.org/ Retrieved 2014-04-14

[109] Opendaylight :"Open Daylight Wiki" https://wiki.opendaylight.org/view/Main_Page Retrieved 2014-04-14

[110] OpenFaucet: https://github.com/rlenglet/openfaucet Retrieved 2014-04-14

[111] OpenFlowJ: https://bitbucket.org/openflowj/openflowj Retrieved 2014-04-14

[112] Wundsam, Andreas and Levin, Dan and Seetharaman, Srini and Feldmann, Anja (2011). OFRewind: Enabling Record and Replay Troubleshooting for Networks. Proceedings of Usenix Annual Technical Conference (Usenix ATC '11). Usenix, 327340.

[113] OESS: Open Exchange Software Suite, http://globalnoc.iu.edu/sdn/oess.html

[114] Nascimento, M. R., C. Esteve Rothenberg, Salvador, M. R., and Magalhaes, M. F. (2010). QuagFlow: partnering Quagga with OpenFlow. SIGCOMM CCR, 40:441442.

[115] FlowScale - http://www.openflowhub.org/display/FlowScale/FlowScale+Home

[116] Amin Tootoonchian, Monia Ghobadi, Yashar Ganjali, OpenTM: Traffic Matrix Estimator for OpenFlow Networks Passive and Active Network Measurement Conference (PAM), Zurich, Switzerland, April 2010.

[117] ENVI git://github.com/dound/envi.git

[118] LAVI http://www.openflow.org/wk/index.php/LAVI"

[119] Charalampos Rotsos, Nadi Sarrar, Steve Uhlig, Rob Sherwood, and Andrew W. Moore OFLOPS: An Open Framework for OpenFlow Switch Evaluation. Proceedings of Passive and Active Measurements Conference (PAM '12)

[120] Shie-Yuan Wang, Chih-Liang Chou, and Chun-Ming Yang. EstiNet openflow network simulator and emulator. IEEE Communications Magazine (2013)

[121] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya, CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience

(SPE), Volume 41, Number 1, Pages: 23-50, ISSN: 0038-0644, Wiley Press, New York, USA, January, 2011.

[122] autonetkit.org/

[123] http://ftp.uk.linux.org/pub/linux/Networking/netkit

[124] Mininet: An Instant Virtual Network on your Laptop: mininet.org

[125] . A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta VL2: A Scalable and Flexible Data Center Network. In SIGCOMM , Aug 2009

[126] CORE: http://cs.itd.nrl.navy.mil/work/core/index.php

[127] Air-in-a-Box: air-in-a-box.sourceforge.net/

[128] RIPL: https://github.com/brandonheller/ripl

[129] Christopher Monsanto, Nate Foster, Rob Harrison, and David Walker. 2012. A compiler and run-time system for network programming languages. In Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '12). ACM, New York, NY, USA, 217-230

[130] OF-Lib: https://github.com/TrafficLab/oflib-node

[131] "A. Guha, M. Reitblatt, and N. Foster, Formal foundations for software defined networks, in Open Net Summit, 2013."

[132] Tom Nolle, Centralized vs. decentralized SDN architecture: Which works for you? http://searchsdn.techtarget.com/tip/Centralized-vs-decentralized-SDN-architecture-Which-works-for-you

[133] "P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte, Real time network policy checking using header space analysis, in USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2013."

[134] E. Al-Shaer and S. Al-Haj, Flowchecker: Configuration analysis and verification of federated openflow infrastructures, in Proceedings of the 3rd ACM workshop on Assurable and usable security configuration, pp. 3744, ACM, 2010.

[135] Tom Nolle: SDN technologies primer: Revolution or evolution in architecture? http://searchsdn.techtarget.com/tip/SDN-technologies-primer-Revolution-or-evolution-in-architecture

[136] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte, Real time network policy checking using header space analysis, in USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2013.

[137] H. Yang and S. S. Lam, Real-time verification of network properties using atomic predicates, ICNP, the IEEE International Conference on Network Protocols, 2013.

[138] S. Gutz, A. Story, C. Schlesinger, and N. Foster, Splendid isolation: A slice abstraction for software-defined networks, in Proceedings of the first workshop on Hot topics in software defined networks, pp. 7984, ACM, 2012.

[139] E. Al-Shaer, W. Marrero, A. El-Atawy, and K. ElBadawi, Network configuration in a box: Towards end-to-end verification of network reachability and security, in Network Protocols, 2009. ICNP 2009. 17th IEEE International Conference on, pp. 123132, IEEE, 2009.

[140] S. Zhang, A. Mahmoud, S. Malik, and S. Narain, Verification and synthesis of firewalls using SAT and QBF, in Network Protocols (ICNP), 2012 20th IEEE International Conference on, pp. 16, IEEE,2012.

[141] S. Son, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, Model checking invariant security properties in openflow,

[142] T. Nelson, C. Barratt, D. J. Dougherty, K. Fisler, and S. Krishnamurthi, The margrave tool for firewall analysis, in USENIX Large Installation System Administration Conference, 2010.

[143] N. Kothari, R. Mahajan, T. Millstein, R. Govindan, and M. Musuvathi, Finding protocol manipulation attacks, SIGCOMM-Computer Communication Review, vol. 41, no. 4, p. 26, 2011.

[144] A.Wang, L. Jia, C. Liu, B. T. Loo, O. Sokolsky, and P. Basu, Formally verifiable networking, in HotNets, ACM Sigcomm, 2009.

[145] A. Noyes, T. Warszawski, and N. Foster, Toward synthesis of network updates, in Workshop on Synthesis (SYNT), 2013.

[146] "U. T. B. T. L. Anduo Wang, Salar Moarref and A. Scedrov., Automated synthesis of reactive controllers for software-defined networks, The 3rd International Workshop on Rigorous Protocol Engineering, WRIPE 2013, 2013."

[147] H. Zeng, P. Kazemian, G. Varghese, and N. McKeown, Automatic test packet generation, in Proceedings of the 8th international conference on Emerging networking experiments and technologies, pp. 241252, ACM, 2012

[148] NEC ProgrammableFlow : http://www.nec.com/en/global/prod/pflow/.

[149] C. Scott, A. Wundsam, S. Whitlock, A. Or, E. Huang, K. Zarifis, and S. Shenker, How did we get into this mess? isolating faultinducing inputs to sdn control software, tech. rep., Technical Report UCB/EECS-2013-8, EECS Department, University of California, Berkeley, 2013

[150] D. Sethi, S. Narayana, and S. Malik, Abstractions for Model Checking SDN Controllers, in Formal Methods in Computer Aided Design, 2013.

[151] T. Nelson, A. Guha, D. J. Dougherty, K. Fisler, and S. Krishnamurthi, A balance of power: Expressive, analyzable controller programming, 2013.

[152] A. Guha, M. Reitblatt, and N. Foster, Machine-verified network controllers., in PLDI, pp. 483494, 2013.

[153] Allied Telesis, Demystifying Software-Defined Networking, http://www.alliedtelesis.com/userfiles/file/WP_Demystifying_SDN_RevA.pdf

[154] "B. Heller, C. Scott, N. McKeown, S. Shenker, A. Wundsam, H. Zeng, S. Whitlock, V. Jeyakumar, N. Handigol, J. McCauley, et al., Leveraging SDN layering to systematically troubleshoot networks, in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, pp. 3742, ACM, 2013."

[155] M.-K. Shin, K.-H. N. M. Kang, and J.-Y. Choi, Formal specification and programming for sdn, IETF 84 Proceedings, 2012.

[156] R. W. Skowyra, A. Lapets, A. Bestavros, and A. Kfoury, Verifiablysafe software-defined networks for CPS, in Proceedings of the 2nd ACM international conference on High confidence networked systems, pp. 101110, ACM, 2013.

[157] N. A. Handigol, Using packet histories to troubleshoot networks. PhD thesis, Stanford University, 2013.

[158] A. Wundsam, D. Levin, S. Seetharaman, and A. Feldmann, Ofrewind: enabling record and replay troubleshooting for networks, in USENIX ATC, 2011.

[159] R. C. Scott, A. Wundsam, K. Zarifis, and S. Shenker, What, Where, and When: Software Fault Localization for SDN, tech. rep., Technical Report UCB/EECS-2012-178, EECS Department, University of California, Berkeley, 2012.

[160] P. Reynolds, C. E. Killian, J. L. Wiener, J. C. Mogul, M. A. Shah, and A. Vahdat, Pip: Detecting the unexpected in distributed systems., in NSDI, vol. 6, pp. 115128, 2006.

[161] C. Scott, A. Wundsam, S. Whitlock, A. Or, E. Huang, K. Zarifis, and S. Shenker, Automatic troubleshooting for sdn control software,2013

[162] "B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M. Hellerstein, P. Maniatis, R. Ramakrishnan, T. Roscoe, and I. Stoica, Declarative networking, Communications of the ACM, vol. 52, no. 11, pp. 8795, 2009."

[163] G. Stewart, Computational verification of network programs in coq, in Certified Programs and Proofs, 2013

[164] The Frenetic Research Project [Online]. http://www.frenetic-lang.org. Accessed: 2013-09-12.

[165] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, Abstractions for network update, in Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication, pp. 323334, ACM, 2012

[166] Graham Finnie, Policy Control and SDN: A Perfect Match ,https://www.sandvine.com/downloads/general/analyst-reports/analystreport-heavy-reading-policy-control-and-sdn-a-perfect-match.pdf

[167] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, Composing software defined networks, NSDI, Apr, 2013.

[168] C. J. Anderson, N. Foster, A. Guha, J.-B. Jeannin, D. Kozen, C. Schlesinger, and D. Walker, NetKAT: Semantic Foundations for Networks, 2013.

[169] T. L. Hinrichs, N. S. Gude, M. Casado, J. C. Mitchell, and S. Shenker, Practical declarative network management, in Proceedings of the 1st ACM workshop on Research on enterprise networking, pp. 110, ACM, 2009.

[170] N. P. Katta, J. Rexford, and D. Walker, Logic programming for software-defined networks, in Workshop on Cross-Model Design and Validation (XLDI), 2012.

[171] Jim Metzler, SDN ControllersThe SDN Journey, http://www.sdncentral.com/education/sdnjourney-sdn-controllers-metzler/.

[172] Qadir, J. and Hasan, O. (2013). Applying Formal Methods to Networking: Theory, Techniques and Applications.. CoRR, abs/1311.4303.

[173] B. Nunes, Marc Mendonca, Xuan-Nam Nguyen, K. Obraczka, Thierry Turletti,"A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks", IEEE Communications Surveys and Tutorials.

[174] Nick Feamster, Jennifer Rexford, and Ellen Zegura. 2013. The Road to SDN. Queue 11, 12, pages 20 (December 2013), 21 pages

[175] Tie Luo; Hwee-Pink Tan; Quek, T.Q.S., "Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks," Communications Letters, IEEE , vol.16, no.11, pp.1896,1899, November 2012

[176] Santos, M.A.S.; de Oliveira, B.T.; Margi, C.B.; Nunes, B.A.A.; Turletti, T.; Obraczka, K., "Software-defined networking based capacity sharing in hybrid networks," Network Protocols (ICNP), 2013 21st IEEE International Conference on , vol., no., pp.1,6, 7-10 Oct. 2013

[177] Abhishek Chanda and Cedric Westphal. 2013. A content management layer for software-defined information centric networks. In Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking (ICN '13). ACM, New York, NY, USA, 47-48.

[178] H Jafarian, E Al-Shaer, Q Duan "Openflow Random Host Mutation: Transparent Moving Target Defense using Software Defined Networking, HotSDN 2012.

[179] Yong Cui; Shihan Xiao; Chunpeng Liao; Stojmenovic, I.; Minming Li, "Data Centers as Software Defined Networks: Traffic Redundancy Elimination with Wireless Cards at Routers," Selected Areas in Communications, IEEE Journal on , vol.31, no.12, pp.2658,2672, December 2013

[180] "Network Functions Virtualization Introductory White Paper". ETSI. 22 October 2012, http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[181] SDN/NFV Primer : http://www.6wind.com/software-defined-networking/sdn-nfv-primer/

[182] Sadasivarao, A.; Syed, S.; Ping Pan; Liou, C.; Monga, I.; Chin Guok; Lake, A., "Bursting Data between Data Centers: Case for Transport SDN," High-Performance Interconnects (HOTI), 2013 IEEE 21st Annual Symposium on , vol., no., pp.87,90, 21-23 Aug. 2013

[183] D. Mcdysan, Software defined networking opportunities for transport, IEEE Communications Magazine , vol. 51, no. 3, pp. 2831, 2013.

[184] A. Sadasivarao, S. Syed, P. Pan, C. Liou, A. Lake, C. Guok, , and I. Monga, Open Transport Switch - A Software Defined Networking Architecture for Transport Networks, in Proceedings of the ACM SIGCOMM , 2013

[185] Sharon Barkai, Randy Katz, Dino Farinacci, and David Meyer, Software defined flow-mapping for scaling virtualized network functions. HotSDN, page 149-150. ACM, (2013)

[186] Prayson Pate,NFV and SDN: Whats the Difference?. http://www.sdncentral.com/technology/nfv-and-sdn-whats-the-difference/2013/03/

[187] NFV, Network Virtualization, OpenFlow and SDN Use Cases http://www.sdncentral.com/sdn-use-cases/

[188] Network Overlays: An Introduction http://www.networkcomputing.com/networking/network-overlays-an-introduction/

[189] Proactive Overlay versus Reactive Hop-by-Hop, Juniper Networks, http://www.juniper.net/us/en/local/pdf/whitepapers/2000515-en.pdf

[190] Tatsuhiro Ando, Osamu Shimokuni, Katsuhito Asano Network Virtualization for Large-Scale Data Centers, http://www.fujitsu.com/downloads/MAG/vol49-3/paper14.pdf

[191] Narten, T., Black, D., Dutt, D., Fang, L., Gray, E., Kreeger, L., Napierala, M., and Sridhavan, M., "Problem Statement: Overlays for Network Virtualization," draft-narten-nvo3-overlay-problem-statement-04 (work in progress), August 2012

[192] Ashton, Metzler, et. al. Ten Things to Look for in an SDN Controller. https://www.necam.com/Docs/?id=23865bd4-f10a-49f7-b6be-a17c61ad6fff

[193] Telefonica: NFV Move in the network http://www.ietf.org/proceedings/87/slides/slides-87-nsc-0.pdf

[194] Hideo Kitazume, Takaaki Koyama, Toshiharu Kishi, and Tomoko Inoue Network Virtualization Technology for Cloud Services. NTT Information Sharing Platform Laboratories. https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201112fa4.html

[195] Yun Chao Hu, Defining NFV, ITU Workshop on Software Defined Networking (SDN) http://www.itu.int/en/ITU-T/Workshops/S1P2_Yun_Chao_Hu_V2.pptx

[196] Antonio Manzalini, Future Edge ICT Networks, IEEE COMSOC MMTC E-Letter, Vol.7, No.7, September 2012.

[197] Koji Yamazaki,Accelerating SDN/NFV with Transparent Offloading Architecture, https://www.usenix.org/conference/ons2014/technical-sessions/presentation/yamazaki.

[198] 6Wind, SDN/NFV Primer: http://www.6wind.com/software-defined-networking/sdn-nfv-primer/

[199] Andy Reid, Network Functions Virtualization and ETSI NFV ISG http://www.commnet.ac.uk/documents/commnet_workshop_networks/CommNets_EPSRC_workshop_Reid.pdf

[200] Bob Briscoe, Don Clarke, Pete Willis, Andy Reid, Paul Veitch, Network Functions Virtualization http://www.ietf.org/proceedings/86/slides/slides-86-sdnrg-1.pdf

[201] Terast3eam: A simplified service delivery model. https://ripe67.ripe.net/presentations/131-ripe2-2.pdf

[202] SDN in Terastream: http://www.opennetsummit.org/pdf/2013/presentations/axel_clauberg_hakan_millroth.pdf

[203] Daniel Abgrall, Virtual Home Gateway. http://archive.eurescom.eu/~pub/deliverables/documents/P2000-series/P2055/D1/P2055-D1.pdf

[204] Unbudling- Wiki: http://en.wikipedia.org/wiki/Unbundling

[205] Joao Martins, Mohamed Ahmed, Costin Raiciu, and Felipe Huici. 2013. Enabling fast, dynamic network processing with clickOS. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (HotSDN '13). ACM, New York, NY, USA, 67-72.

[206] Jim Machi, NFV Said to SDN: Ill Be There for You http://www.sdncentral.com/market/nfv-said-sdn-ill/2013/12/

[207] OpenFlow-enabled SDN and Network Functions Virtualization https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-sdn-nvf-solution.pdf

[208] NEC's SDN for the Service Driven Network http://www.nec.com/en/global/solutions/nsp/sdn/doc/sdn-nfv_wp.pdf

[209] Mayur Channegowda, Reza Nejabati, and Dimitra Simeonidou, Software-Defined Optical Networks Technology and Infrastructure: Enabling Software-Defined Optical Network Operations, Journal of Optical Communications and Networking, Vol. 5, Issue 10, pp. A274-A282 , 2013.

[210] S. Martnez, V. Lpez, M. Chamania, O. Gonzlez, A. Jukan, J.P. Fernndez-Palacios, Assessing the Performance of Multi-Layer Path Com-

putation Algorithms for different PCE Architectures, (OFC/NFOEC), 17-21 March 2013.

[211] Ericsson White paper, The real-time cloud, February 2014.

[212] ITU-T OTN Definition: http://www.itu.int/ITU-T/2001-2004/com15/otn/definitions.html

[213] Aditya Gudipati, Daniel Perry, Li Erran Li, Sachin Katti, SoftRAN: Software Defined Radio Access Network, HotSDN 2013.

[214] S. Khan, J. Edstam, B. Varga, J. Rosenberg, J. Volkering, M. Stumpe, The benefits of self-organizing backhaul networks, Ericsson Review, September 2013.

[215] CloudNFV. http://www.cloudnfv.com/

[216] R. Guerzoni, Z. Despotovic, R. Trivisonno, I. Vaishnavi, A. Hecker, S. Beker Future Architectures for Resource Orchestration, ETSI workshop on Future Networks, Sophia Antipolis, 9-11 April 2013.

[217] Openstack: https://www.openstack.org/

[218] Openstack Neutron: https://wiki.openstack.org/Neutron

[219] Alcatel Cloudband: http://www.alcatel-lucent.com/solutions/cloudband

[220] Pica8's xorplus http://sourceforge.net/p/xorplus/home/Pica8\%20Xorplus/

[221] L. Lewin-Eytan, K. Barabash, R. Cohen, V. Jain, and A. Levin. Designing modular overlay solutions for network virtualization. In IBM Technical Paper, 2012.

[222] Network Heresy, Network Virtualization, Encapsulation, and Stateless Transport Tunneling. March 2012. http://networkheresy.com/2012/03/04/network-virtualization-encapsulation-and-stateless-tcp-transport-stt/

[223] Jorge Carapinha and Javier Jimnez, Network Virtualization a View from the Bottom, VISA09, August 17, 2009, Barcelona, Spain.

[224] D. Schlosser, M. Jarschel, M. Duelli, T. Hofeld, K. Hoffmann, M. Hoffmann, H.-J. Morper, D. Jurca, A. Khan, A Use Case Driven Approach To Network Virtualization, IEEE Kaleidoscope 2010.

[225] Scott Lowe, Network Overlays vs. Network Virtualization, http://blog.scottlowe.org/2013/04/16/network-overlays-vs-network-virtualization/

[226] Raj Jain and Subharthi Paul, "Network Virtualization and Software Defined Networking for Cloud Computing - A Survey," IEEE Communications Managzine, Nov 2013, pp. 24-31.

[227] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba, A Survey of Network Virtualization, Technical Report: CS-2008-25, October 15, 2008.

[228] Md. Faizul Bari, Raouf Boutaba, Rafael Esteves, Lisandro Zambenedetti Granville, Maxim Podlesny, Md Golam Rabbani, Qi Zhang, and Mohamed Faten Zhani Data Center Network Virtualization: A Survey, IEEE Communications Surveys and Tutorials, VOL. 15, NO. 2, 2013.

[229] Piotr Rygielski and Samuel Kounev. Network Virtualization for QoS-Aware Resource Management in Cloud Data Centers: A Survey. PIK - Praxis der Informationsverarbeitung und Kommunikation, 36(1), 2013.

[230] Michelle McNickle, SDN vs. network virtualization: Q&A with VMware's Martin Casado http://searchsdn.techtarget.com/news/2240183487/SDN-vs-network-virtualization-QA-with-VMwares-Martin-Casado

[231] Teemu Koponen et. al. Network Virtualization in Multi-tenant Datacenters, Technical Report - TR-2013-001E, VMWare, 2013.

[232] Multitenancy, Wikipedia, http://en.wikipedia.org/wiki/Multitenancy

[233] Andrs Csszr et. al. Unifying Cloud and Carrier Networks (UNIFY), www.fp7-unify.eu

[234] Ericsson: The Telecom Cloud Opportunity, White Paper, 2012. http://www.ericsson.com/res/site_AU/docs/2012/ericsson_telecom_cloud_discussion_paper.pdf

[235] TeliaSonera: The Telco and the Cloud. Whitepaper http://www.teliasonera.com/Documents/Public%20policy%20documents/WhitePaperOnCloudServices.pdf

[236] Huawei: Telco Cloud- The Time is Now. http://www.huawei.com/en/about-huawei/publications/communicate/hw-193377.htm

[237] Alcatel-Lucent: Open Cloud Architecture http://www.alcatel-lucent.com/solutions/cloud.

[238] ETSI NFV: www.etsi.org/technologies-clusters/technologies/nfv

[239] Volker Held, 5 key ingredients to build your telco cloud, http://blogs.nsn.com/mobile-networks/2013/11/14/5-key-ingredients-for-building-the-telco-cloud/

[240] Eugen Borcoci, Software Defined Networking and Architectures, Netware 2013 Conference August 25, 2013 Barcelona

[241] Li Erran Li Z. Morley Mao Jennifer Rexford, Toward Software-Dened Cellular Networks, European Workshop on Software Defined Networking (EWSDN), 2012

Sridhar K. N. Rao Sridhar received his Ph.D degree in computer science from National University of Singapore, in 2007, his M.Tech. degree in computer science from KREC, Suratkal, India, in 2000, and his B.E. degree in instrumentation and electronics from SIT, Tumkur, Bangalore University, India, in August 1997. Prior to NEC technologies, Sridhar has worked as post-doctoral fellow at Microsoft Innovation Center, Politecnico Di Torino, Turin, Italy, and as a research fellow at Institute for Infocomm Research (I2R) Singapore, and as Research Lead at SRM Research Institute, Bangalore. His research interests are mainly in the domain of next-generation wired and wireless networking, such as openflow, software defined networking, software-defined radio based systems for congnitive networks, Hotspot 2.0 and Internet of Things. He has also worked on various development and deployment projects involving both commercial and open-source open-flow based controllers and switches, ZigBee, WiFi and WiMax.