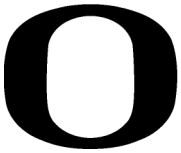
Presented to the Interdisciplinary Studies Program:



UNIVERSITY OF OREGON
APPLIED INFORMATION MANAGEMENT

Applied Information Management
and the Graduate School of the
University of Oregon
in partial fulfillment of the
requirement for the degree of
Master of Science

Software-Defined Networks (SDN): What Systems Integrators Need to Know

CAPSTONE 1 Bibliography

Joshua D. Burman
Senior Systems Engineer
Right! Systems, Inc.

University of Oregon
Applied Information
Management
Program

December 2013

Academic Extension
1277 University of Oregon
Eugene, OR 97403-1277
(800) 824-2714

Approved by

_____

Dr. Linda F. Ettinger, Capstone Instructor

_____

Dr. Kara McFall, Capstone Instructor

Software-Defined Networks (SDN): What Systems Integrators Need to Know

Joshua D. Burman

Right! Systems

Table of Contents

List of Tables and Figures

Introduction

Problem

Software–Defined Networking (SDN) is a new networking paradigm proposed as a way to programmatically control networks, with the goal of making it easier to deploy new applications and services, as well as adjust network policy and performance (Mendonca, Nunes, Nguyen, Obraczka, & Turletti, 2013). Many network switch and router manufacturers design closed systems so people cannot implement their own protocols onto their own devices (Lim & Obraczka, n.d.).

Li and Liao (2013) describe the necessity of creating more open network systems:

> A traditional switch or router consists of a high-speed data plane where the packets are routed, and a control plane that includes functions for setting up routing and forwarding. Separating the control plane from the data plane make it feasible to run the control plane in software on standard servers, and thus enables the creation of new virtualized controllers and custom-made services. (p. 1)

The lack of separation between control and data planes creates the need for independently managed, complex, and purpose-built individual devices that are both expensive and time-consuming to implement (Hui, & Koponen, 2012). To overcome this network design problem, devices can be managed using *topology abstraction* (Monsanto, Reich, Foster, Rexford, & Walker, 2013) in which the forwarding hardware is decoupled from control decisions (Mendonca, Astuto, Nguyen, Obraczka, & Turletti, 2013). Shenker (2011) notes that abstractions divide problems into tractable pieces. Software-defined networking (SDN) is designed to accomplish this abstraction.

Software-defined networking (SDN) is defined by Shinde and Tamhankar (2013) as the separation of the control plane from the data plane on network devices such as routers and switches, in order to make them more easily programmable from a centralized location. As noted by Feamster, Rexter, and Zegura (2013), over the past few years, SDN has gained significant traction in industry, as evidenced by the many commercial switches that support the application programming interface known as OpenFlow API, which is the industry's first and most prevalent implementation of SDN. Bethany Mayer, the senior vice president and general manager of Hewlett Packard's networking business unit, predicts that a significant uptake in SDN adoption will occur by 2015 (Murray, 2013).

However, according to Hui and Koponen (2012), despite the growing body of research and industrial initiatives based on software-defined networks over the past few years, knowledge of exactly how SDN works still remains unclear to many IT professionals. And according to John Dix (2013), editor in chief of Network World Magazine, only 10% of 450 IT practitioners at a recent Network World event raised their hands when asked if they understand SDN. Lenrow (2012) suggests that because of this limited amount of industry knowledge, software-defined networking is often mischaracterized as a solution looking for a problem.

Purpose

Network configuration and installation requires highly skilled personnel adept at establishing and maintaining many data network elements (Sezer et al., 2013). SDN is meant to reduce the dependency on these personnel, but many pre-sales consultants working for systems integrators have limited ability to explain the benefits to customers (Westling, personal communication, 2013). The purpose of this annotated bibliography is to define the SDN approach in a way that can be understood by networking professionals, including pre-sales

consultants, so that they are better able to explain this approach to potential SDN customers. According to Dosanjh (2012), director of worldwide sales at Aldara Networks, in order to really understand SDN, it's important to have an understanding of the environment that has created a need for this technology. Some systems integrators, and their customers, are confusing the marketplace by categorizing products and capabilities—such as network management software, and router or switch device programmability as SDN-related (Ortiz, 2013).

Research Questions

Main question. What do pre-sales network engineering consultants working for system integrators need to know about the software-defined networking (SDN) approach in order to convey the benefits and use cases for the technology?

Sub-question. Hui and Koponen (2012), note that industry and academia, are pursuing discussion about the essence of SDN with different mind-sets, different solutions and different implications in mind. According to Meyer (2013), software-defined networks are widely seen as a promising solution for resolving challenges of network abstraction. A sub-question for consideration in this study is, what specific challenges should the SDN approach enable organizations to resolve?

Audience. Customers are hearing the term 'SDN' a lot, and they want to find out what it means (Florintine, 2013). The intended audience for this paper is networking professionals in the system integrator community. According to Moore, in information technology, a systems integrator is an organization with expertise in linking together different computing systems and software applications physically or functionally (as cited in Ren, 2011, p. 25). This paper specifically focuses on network engineering pre-sales consultants working for systems integrator

organizations, who must be able to talk about the SDN approach within the larger context of information technology. Since systems integrators both sell network equipment and provide professional services to customers, networking professionals within these organizations also require a working knowledge of the technology is in order to provide useful and reliable advice. In addition, anyone seeking a greater understanding of SDN, and who would also like to join the greater discussion surrounding it, may benefit.

Search Report

This annotated bibliography is designed to provide an expanded definition of software-defined networking (SDN), focusing on its relation to information requirements of network engineering consultants, who must speak to customers about the benefits and use cases for the technology. The goal is to search for and identify published literature that addresses aspects of this expanded definition.

Search strategy. The search for relevant and credible references is conducted online using the following databases:

- Google Scholar
- ArXiv.org
- Web of Knowledge
- Computer Source
- IEEE Xplore
- CiteSeer
- ACM Digital Library

The selected databases cover a wide range of topics or are specialized in the areas of computers and high technology.

Key words include:

- Software defined network
- SDN
- OpenFlow
- Network Abstraction

The key word *software defined network* is provided by Mark Westling, Chief Technology Officer (CTO) of Right! Systems Incorporated, as a recommended focus area (personal communication, 2013). The remaining key words are derived through preliminary reading of research articles, and identification of terms that are pertinent to the stated problem.

Documentation approach. Zotero increases the efficiency of the documentation, by providing the ability to note collected articles, quickly determine duplicates, and arrange them using a taxonomy that was easy to follow. The taxonomy included arranging articles according to their relevancy to the stated problem, and also separating the less relevant and unusable articles. Articles cited, articles read, and unread articles are also added to separate collections for tracking purposes.

Reference evaluation criteria. At the onset of the search process, articles included provide a basic understanding of the concept of software-defined networks. As the search progresses, additional references containing more detailed information potentially valuable to a network engineer in a pre-sales role are utilized. The quality of references is evaluated using the following criteria, provided on the University of Oregon Libraries website, and titled Critical Evaluation of Information Sources (Bell & Frantz, 2013):

- Authority
- Objectivity
- Quality
- Currency
- Relevancy

Authority. The reference must be written by a recognized expert on the subject of SDN, computer networking, or system integrator operations. Author's credentials and/or background are reviewed prior to inclusion of their works.

Objectivity. References representing any special interests, such as corporations or industry groups, are vetted for objectivity. Any content exhibiting bias is excluded.

Quality. The reference must be clearly written, and logically arranged, so that if any reader decides to perform additional reading, they will understand the content.

Currency. The reference must be written within the past five years (2008-2013) if it pertains to SDN, otherwise ten years for subjects relating to general systems integration. The data parameters for literature pertaining to SDN are established in order to ensure the most up to date information is presented, as the environment for SDN is constantly changing.

Relevancy. The reference must be appropriate for the stated problem; focus is on software-defined networking, so that readers seeking additional information will not be directed to irrelevant content.

Annotated Bibliography

The following annotated bibliography consists of 15 selected references, organized into three categories. The categories are designed to address the main research question posed in this study, which asks: What do pre-sales network engineering consultants working for system integrators need to know about the software-defined networking (SDN) approach in order to convey the benefits and use cases for the technology? The categories (a) introduce software-defined networking to the reader through presentation of a history and explanation of SDN, including the benefits, (b) discuss example SDN use cases for implementation, and (c) explain the challenges associated with adoption of SDN technology. According to Cockburn (1995), a *use case* is a collection of possible sequences of interactions between the system under discussion and its external actors, related to a particular goal. The system under discussion for the purpose of this paper is software-defined networking. A benefit is defined as a way to reduce dependency on personnel to establish and maintain network configuration and installation (Sezer et al., 2013). According to Hui and Koponen (2012), a challenge is defined as a hard problem in software defined networks. They describe such problems as the challenges to SDN adoption, which fall into three categories: (a) technical, (b) social, and (c) economic.

Each reference is described in an annotation containing three elements: (a) a bibliographic citation, (b) the abstract, and (c) a summary. The bibliographic citation is presented in APA format. The abstract is provided as published in the reference, and in some cases is edited for length or relevancy. The summary is written in a way to present information relevant to the reader about SDN, within the framework provided by the research questions.

Category 1: History and Explanation of SDN, Including Benefits

Casado, M., Koponen, T., Shenker, S., & Tootoonchian, A. (2012). Fabric: A retrospective on evolving SDN. In *Proceedings of the first workshop on hot topics in software-defined networks* (pp. 85–90). New York, NY, USA: ACM. doi:10.1145/2342441.2342459

Abstract. MPLS was an attempt to simplify network hardware while improving the flexibility of network control. Software-defined networking (SDN) was designed to make further progress along both of these dimensions. While a significant step forward in some respects, it was a step backwards in others. In this paper we discuss SDN's shortcomings and propose how they can be overcome by adopting the insight underlying MPLS.

Summary. This article begins by describing traditional network designs, and the need for a new paradigm, which is: (a) simple, (b) vendor-neutral, (c) future-proof, and (d) flexible. According to the authors, these elements are not satisfied in today's network infrastructure. It goes on to address the emergence of multi-protocol label switching (MPLS), as a major step in the right direction to following this new paradigm, but is lacking in certain key areas. In the area of network design, three relevant interfaces are identified: (a) host—network, (b) operator—network, and (c) packet—switch. The host-network interface is how a host, such as a server, informs the network of its requirements. The operator—network interface is how operators, or network managers, inform the network of their requirements, through manual configuration or SDN. The packet—switch interface is how a packet, the actual network traffic, identifies itself to the network switch. From here the techniques by which the original Internet, MPLS and software-defined networking implement these interfaces are compared, with emphasis on how the later overcomes the deficiencies of the first two.

Das, T., Caria, M., Jukan, A., & Hoffmann, M. (2013). A Techno-economic Analysis of Network Migration to Software-Defined Networking (arXiv e-print No. 1310.0216). Retrieved from http://arxiv.org/abs/1310.0216

Abstract. As the Software Defined Networking (SDN) paradigm gains momentum, every network operator faces the obvious dilemma: when and how to migrate from existing IP routers to SDN compliant equipment. A single step complete overhaul of a fully functional network is impractical, while at the same time, the immediate benefits of SDN are obvious. A viable solution is thus a gradual migration over time, where questions of which routers should migrate first, and whether the order of migration makes a difference, can be analyzed from techno economic and traffic engineering perspective. In this paper, we address these questions from the techno economic perspective, and establish the importance of migration scheduling. We propose optimization techniques and greedy algorithms to plan an effective migration schedule, based on various techno economic aspects, such as technological gains in combinations with CapEx limitations. We demonstrate the importance of an effective migration sequence through two relevant network management metrics, namely, number of alternative paths availed by a node on migration, and network capacity savings. Our results suggest that the sequence of migration plays a vital role, especially in the early stages of network migration to SDN.

Summary. This paper begins in part one by describing software-defined networking as a potential way to create the benefit of a reduction in operational expenditures (OpEx) by (a) simplifying operations, (b) optimizing resource usage, and (c) simplifying network software upgrades. It also can reduce capital expenditures (CapEx), by utilizing

cheaper equipment to perform the control plane functions. Lastly, the paper describes a Java-based simulation that was performed to test the benefits of SDN migration, and the subsequent results, which prove a reduced CapEx investment, lower time complexity, optimal alternate path selection, and network capacity savings associate with SDN migration.

Feamster, N., Rexford, J., & Zegura, E. (2013). The road to SDN: An intellectual history of programmable networks. Retrieved from https://www.cs.princeton.edu/courses/archive/fall13/cos597E/papers/sdnhistory.pdf

Abstract. Software-defined networking (SDN) is an exciting technology that enables innovation in how we design and manage networks. Although this technology seems to have appeared suddenly, SDN is part of a long history of efforts to make computer networks more programmable. In this paper, we trace the intellectual history of programmable networks, including active networks, early efforts to separate the control and data plane, and more recent work on Open-Flow and network operating systems. We highlight key concepts, as well as the technology pushes and application pulls that spurred each innovation. Along the way, we debunk common myths and misconceptions about the technologies and clarify the relationship between SDN and related technologies such as network virtualization.

Summary. This paper begins by describing the complexity and difficulty of managing modern data networks. These problems are caused in part by routers and switches running complex, distributed control software that can be closed or proprietary. It describes how SDN consolidates the distributed control software and centralizes it on a unified management interface. The traction SDN has gained over the past few years is

then described, including the Openflow API, and the early commercial successes of SDN to include Google's wide-area traffic management system, and Nicira's network virtualization platform. For additional background, the paper chronicles the precursors to software-defined networking, and includes a graphical historic and predictive timeline spanning from 1995 to 2015. It describes the methodology for separation of the control and data planes of network devices, and why it is needed. It recounts the emergence of the OpenFlow protocol as a non-proprietary interface into the control plane of network devices. The traditional paradigms for SDN are listed, including logically centralized control using an open interface to the data plane, and distributed state management. The intellectual contributions of OpenFlow that build on existing paradigms are listed, including: (a) generalizing network devices and functions, (b) the vision of a network operating system, and (c) distributed state management techniques. Lastly, it details the concept of network virtualization in terms of abstraction. This includes a discussion of network virtualization prior to SDN, and the relationship of network virtualization to SDN.

Hui, P., & Koponen, T. (2012). Software-defined networking.

Retrieved from http://vesta.informatik.rwth-aachen.de/opus/volltexte/2013/3789/pdf/dagrep_v002_i009_p095_s12363.pdf

Abstract. This report documents the talks and discussions of Dagstuhl Seminar 12363 "Software-Defined Networking". The presented talks represent a spectrum of industrial and academic work as well as both technical and organizational developments surrounding software-defined networking (SDN). The topic of SDN has garnered significant attention over the past few years in the networking community and beyond,

and indeed the term “Software Defined Networking” itself carries different meaning among different circles. A key focus of the talks and discussions presented here is to capture the essence of SDN through concrete network applications, operational experience reports, and open research problems.

Summary. The paper begins by describing software-defined networking as moving the control plane out from the network elements into stand-alone servers, while the switching elements can remain simple, general-purpose, and cost-effective. It describes the purpose of the Dagstuhl Seminar, the organization of the seminar, followed by a brief synopsis of each talk. The purpose of the seminar is to look at current developments in software-defined networking and identify future research challenges. The relevant talks include:

1. Software defined networking: overview and use cases

   This talk states there have historically been many examples of split-architecture networks (prior to SDN), but they did not introduce sufficient abstractions to realize full convergence.

2. Evolving SDN: My view on "what the heck is it?"

   This talk states that SDN goes beyond simply decoupling the control plane from the data plane, into using modern distributed systems approaches with modular software design principles and tools.

3. Logically centralized? State distribution trade-offs in software defined networks

   This talk states that a core benefit of SDN is to enable the network control logic to be designed and operated using a global network view, as if it were a centralized application, rather than a distributed system. Hence the term, *logically centralized*.

4. Hard problems in software defined networks

   This talk describes the challenges to SDN adoption as falling into three categories: (a) technical, (b) social, and (c) economic. The technical challenges involve questions about scalability and resilience. Social challenges involve overcoming the existing dogma in network design and operation. Finally, the economic challenges derive from disrupting the existing vendor and operator ecosystem.

Lastly, the paper contains a summary of panel discussions on each day of the seminar. These include answers to some basic questions and concerns about SDN, which provide a good starting point for additional research.

Li, C.-S., & Liao, W. (2013). Software-defined networks [Guest Editorial]. *IEEE Communications Magazine*, *51*(2), 113–113. doi:10.1109/MCOM.2013.6461194

Abstract. This one-page editorial describes the need for software-defined networking in the current information technology ecosystem, along with its value proposition. It also describes other featured articles available from IEEE.

Summary. This article states the Internet is at a critical juncture and has serious shortcomings. It describes David Clark's clean-slate initiative, which focuses on evolving the current vertically integrated network models into a more horizontal and open model though the use of SDN. It introduces the value proposition of SDN, including the rapid introduction of new network functions at software speed, and more seamless integration of the network with IT processes. It also introduces four related articles exploring the current and future directions in SDN networks, for readers seeking additional information.

Lim, J., & Obraczka, K. (n.d.). Software-defined networking and services. Retrieved from http://surf-it.soe.ucsc.edu/sites/default/files/Lim_surfit12_report.pdf

Abstract. Software-defined networking (SDN) is a new paradigm that aim at making networks programmable by decoupling network control from the data forwarding hardware. OpenFlow, a notable SDN protocol, provides an API that lets a logically centralized control element, called a network operating system or "controller", to organize OpenFlow-enabled switches. Such abstractions promote network innovation by permitting finer-grained control and simplified service deployment over the underlying data plane. In our work, we examine how SDN can be used to provide customized services to users. In order to test and evaluate the performance of such services, we create an OpenFlow testbed consisting of OpenFlow-enabled network elements.

Summary. This paper begins with a brief description of the background behind software-defined networking. It explains how traditional switch and router manufacturers design closed systems, preventing in-house development of protocols, and how OpenFlow is developed to counter this problem. It describes views from both an SDN and non-SDN network. In a non-SDN network, the data plane and control plane exist together in each network device. In an SDN network, the control plane is offloaded to a central server via an API such as OpenFlow. It then describes a test bed using gigabit routers supporting OpenFlow, along with two computers and a laptop running Floodlight, an OpenFlow software controller, to illustrate SDN in action.

Mendonca, M., Astuto, B. N., Nguyen, X. N., Obraczka, K., & Turletti, T. (2013). A survey of software-defined networking: Past, present, and future of programmable networks. Retrieved from http://hal.inria.fr/hal-00825087/

Abstract. The idea of programmable networks has recently re-gained considerable momentum due to the emergence of the software-defined networking (SDN) paradigm. SDN, often referred to as a "radical new idea in networking", promises to dramatically simplify network management and enable innovation through network programmability. This paper surveys the state-of-the-art in programmable networks with an emphasis on SDN. We provide a historic perspective of programmable networks from early ideas to recent developments. Then we present the SDN architecture and the OpenFlow standard in particular, discuss current alternatives for implementation and testing SDN-based protocols and services, examine current and future SDN applications, and explore promising research directions based on the SDN paradigm.

Summary. This survey begins by referring to current network devices as "vertically integrated black boxes", and the static architecture of the internet referred to as "Internet Ossification", implying these Internet devices are not programmable and are inflexible. The early attempts at programmable networks are then described, these include: Open Signaling (OPENSIG), Active Networking, 4D Project, NETCONF, ForCES, and Ethane. The elements of software-defined network are described, beginning with switches. The paper then proceeds to describe the software-defined networking architecture, including the switches used for forward traffic, and the controller used to manage the operation of those switches. SDN development tools are then described, these include emulation and simulation tools, switching platforms, and available controller platforms. Helpful in identifying use cases, the potential applications of SDN are listed, to include the following environments: enterprise networks, data centers, infrastructure-based wireless access networks, and home and small business. Finally, the

future directions of SDN are considered in the areas of: (a) controller and switch design, (b) software-defined-internetworking, (c) controller-service interaction, (d) virtualization and cloud services, (e) information-centric networking, and (f) heterogeneous network support.

Monsanto, C., Reich, J., Foster, N., Rexford, J., & Walker, D. (2013). Composing software-defined networks. *NSDI, Apr*. Retrieved from https://www.usenix.org/system/files/conference/nsdi13/nsdi13-final232.pdf

Abstract. Managing a network requires support for multiple concurrent tasks, from routing and traffic monitoring, to access control and server load balancing. Software-defined networking (SDN) allows applications to realize these tasks directly, by installing packet-processing rules on switches. However, today's SDN platforms provide limited support for creating modular applications. This paper introduces new abstractions for building applications out of multiple, independent modules that jointly manage network traffic. We define a new abstract packet model that allows programmers to extend packets with virtual fields that may be used to associate packets with high-level meta-data. We realize these abstractions in Pyretic, an imperative, domain-specific language embedded in Python.

Summary. The paper opens by introducing software-defined networking and the concepts of composition operators, topology abstraction, and the Pyretic programming language and system. Of particular importance is the concept of topology abstraction, which introduces network objects that divide the network into modules, enabling objects to be hidden or protected from view. Of secondary importance is the development of languages that allow programmers to build large, sophisticated controller applications out

of small, self-contained modules, as in the case of the Pyretic language. This facilitates the development of SDN applications following modular programming practices, such as modules that perform discrete functions, allowing for the reuse of code, and the ability for modules to be maintained separately.

Category 2: Examples of SDN Use Cases

Azodolmolky, S., Wieder, P., & Yahyapour, R. (2013). SDN-based cloud computing networking. In *Transparent Optical Networks (ICTON), 2013 15th International Conference on* (pp. 1–4). Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6602678

Abstract. Software Defined Networking (SDN) is a concept which provides the network operators and data centers to flexibly manage their networking equipment using software running on external servers. According to the SDN framework, the control and management of the networks, which is usually implemented in software, is decoupled from the data plane. On the other hand cloud computing materializes the vision of utility computing. Tenants can benefit from on-demand provisioning of networking, storage and compute resources according to a pay-per-use business model. In this work we present the networking issues in IaaS and networking and federation challenges that are currently addressed with existing technologies. We also present innovative software-defined networking proposals, which are applied to some of the challenges and could be used in future deployments as efficient solutions. Cloud computing networking and the potential contribution of software-defined networking along with some performance evaluation results are presented in this paper.

Summary. This paper describes how software-defined networking fits into the cloud computing phenomena. SDN is described as an appealing platform for network virtualization, because control logic can run on a controller, rather than physical network devices. The challenges involved with traditional cloud services involve the complexity of multi-layer architecture and connectivity between the cloud networks of different providers. The advantages of SDN-based cloud network involve a comprehensive view of cloud resources and network availability. It will also facilitate multi-vendor networks between enterprises and data centers.

Hampel, G., Steiner, M., & Bu, T. (2013). Applying software-defined networking to the telecom domain. Retrieved from http://moritzsteiner.de/papers/SDNtelecom.pdf

Abstract. The concept of Software-Defined Networking (SDN) has been successfully applied to data centers and campus networks but it has had little impact in the fixed wireline and mobile telecom domain. Although telecom networks demand fine granular flow definition, which is one of SDN's principal strengths, the scale of these networks and their legacy infrastructure constraints considerably limit the applicability of SDN principles. Instead, telecom networks resort to tunneling solutions using a plethora of specialized gateway nodes, which create high operation cost and single points of failure. We propose extending the concept of SDN so that it can tackle the challenges of the telecom domain.

Summary. This paper opens by introducing software-defined networking and suggesting that it has not played a major role in the telecommunications domain. Telecommunications companies currently use overlay networks using tunnel solutions provided via multiprotocol label switching (MPLS), native internet protocol (IP), and

Ethernet. The paper proposes the use of *vertical forwarding* for telecommunications, which controls the flow of traffic across network layers, where *horizontal forwarding*, the present focus of SDN, is limited to controlling traffic within a single layer. The paper provides a number of use cases for SDN within the telecom domain, including (a) IETF mobility protocols such as mobile IP, (b) mobile network architectures such as universal mobile terrestrial system (UMTS) and system architecture evolution (SAE), (c) wireline broadband networks, (d) virtual private networks and secured links, and (e) IP protocol transition. Finally, the paper suggests the use of the OpenFlow application programming interface (API) to accomplish the implementation of vertical forwarding.

Mendonca, M., Nunes, B., Obraczka, K., & Turletti, T. (2013) Software-defined networking for heterogeneous networks. *IEEE COMSOC MMTC E-Letter.* Retrieved from http://www.comsoc.org/~mmc/

Abstract. Motivated by a vision of a fully connected world, we explore how Software-defined networking (SDN) can be utilized to support heterogeneous environments consisting of both infrastructure-based and infrastructure-less networks. To make the case for SDN in heterogeneous networks, or Heterogeneous SDN (H-SDN), we examine application scenarios in which H-SDN is a key enabling technology.

Summary. The paper opens by conveying the key ideas behind software-defined networking, these include: (a) to remove control decisions from forwarding hardware, (b) to allow forwarding hardware to be programmable via an open interface, and (c) to have a controller define the behavior of the network. Scenarios with both SDN and non-SDN enabled heterogeneous networks are considered and compared. Some requirements and challenges to deploying SDN in a heterogeneous network are described, to include end-

user device limitations, gateway device incentives, resource discovery, control plane, security, and flexible rules and actions.

Category 3: Challenges to SDN Adoption

Alberti, A. M. (2012). Software-defined networking: Perspectives, requirements, and challenges. Retrieved from http://www.researchgate.net/publication/236156555_Software-Defined_Networking_Perspectives_Requirements_and_Challenges/file/e0b495167ed2eeef7e.pdf

Abstract. Software-defined networking is emerging as a new paradigm for network design and implementation. However, many issues regarding this paradigm have been explored only on specific scenarios, e.g. network operating systems, networking customization, open commodity equipment, software-defined radio, data centers, etc. More general and deep studies are necessary to explore better this paradigm as it emerges as a key ingredient for future network architectures. This paper provides a first glance discussion on the perspectives, requirements, and open challenges behind software-defined networking.

Summary. This article begins by providing the following brief definition of software-defined networking: Software-defined networking means to establish networks where equipment functionalities are controlled by software. The implications of implementing SDN are then discussed. Finally, the requirements and their associated challenges are mentioned. The requirements include: (a) genericity, or equipment that is generic enough to be customized by different software instances, thus being fully programmable, (b) performance, (c) isolation, or preventing virtual instances from interfering with one another when using shared hardware resources, (d) reconfigurability, or the ability for the

hardware to change its functionality based upon software input, (e) exposure of the ability of hardware resources, and (f) manageability. The challenge of genericity is achieving the flexibility for a network device to be used by several technologies. The performance challenge involves how to utilize high-performance platforms to facilitate multiple virtual instances. Isolation challenges involve preventing virtual instances from interfering with one another when using shared hardware resources. The challenge of reconfigurability is how to make a live transition from one configuration to another. The exposure of hardware resources has the challenge of how to overcome the developer's lack of knowledge required to expose the hardware resources. The challenge of manageability is how to reduce human intervention in the management of these hardware resources. Isolation challenges involve preventing virtual instances from interfering with one another when using shared hardware resources.

Meyer, D. (2013). The Software-Defined Networking Research Group (SDNRG). Retrieved from http://www.1-4-5.net/~dmm/papers/IC-17-06-Standards.pdf

Abstract. Software-defined networking (SDN) promises to bring radical improvements in both cost and functionality to the networking field. At the same time, SDN poses many fundamental questions, including deep issues such as whether control should be centralized or distributed, and whether control and data planes should be separated or share fate. The Software-Defined Networking Research Group (SDNRG) has been formed to facilitate research into these and other foundational questions.

Summary. This paper opens with a short explanation of the positioning of SDN as a solution to management challenges associated with modern networks. It then presents the goals and objectives of the Software-Defined Networking Research Group (SDNRG),

which is part of the Internet Research Task Force (IRTF). The IRTF is the research arm of the Internet Engineering Task Force (IETF). SDNRG operates as a workshop to allow the SDN community to explore various aspects of SDN, including: (a) classification of SDN models, (b) definitions and taxonomies, (c) SDN model scalability and applicability, (d) multilayer programmability and feedback control, (e) system complexity, (f) languages, abstractions, interfaces, and compilers, (g) verification of correct network/node operations, and (h) security. The paper then discusses the various standards-based organizations involved with SDN, these include: (a) the European Telecommunications Standards Institute (ETSI), (b) the Alliance for Telecommunications Industry Solutions (ATIS), (c) ITU-T, IEEE, the Open Networking Foundation (ONF), (d) the Metro Ethernet Forum (MEF), (e) the Distributed Management Task Force (DMTF), and (f) the Object Management Group (OMG). Finally, the paper discusses the open problems that exist in the SDN space, along with a discussion of what lies ahead with the technology. The open problems include: (a) architectural questions about how SDN networks will scale and evolve, (b) how to build distributed logically centralized control planes, and (c) the degree to which the control plane should be centralized.

Ortiz, S. (2013). Software-defined networking: On the verge of a breakthrough? *Computer*, *46*(7), 10–12. Retrieved from: http://www.computer.org/csdl/mags/co/2013/07/mco2013070010-abs.html

Abstract. Many experts predict that software-defined networking, a technology that's been highly touted for several years, will soon finally begin gaining ground in the marketplace.

Summary. This paper begins by discussing recent developments in information technology, and the need for centralized control of networks. It then describes the current environment in which many SDN implementations are proprietary and are not interoperable with other vendors. The Open Network Foundation (ONF) is trying to overcome these multi-vendor limitations by introducing the OpenFlow protocol, which is a standards-based SDN implementation. The paper continues by discussing current trends in the SDN industry, including current demand. The Open Daylight Project, managed by the Linux Foundation, is described. Its goal is to create an open source platform on which vendors can build SDN products, and thus enable interoperability among heterogeneous vendors. Finally, the barriers to SDN adoption are described, including (a) compromise of survivability and reliability, (b) confusion about products and capabilities, (c) how to migrate to an SDN architecture, (d) securing the controller from attack, and (e) proving to auditors and customers that SDN is as secure as traditional implementations.

Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., & Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, *51*(7), 36–43. doi:10.1109/MCOM.2013.6553676

Abstract. Cloud services are exploding, and organizations are converging their data centers in order to take advantage of the predictability, continuity, and quality of service delivered by virtualization technologies. In parallel, energy-efficient and high-security networking is of increasing importance. Network operators, and service and product providers require a new network solution to efficiently tackle the increasing demands of

this changing network landscape. Software-defined networking has emerged as an efficient network technology capable of supporting the dynamic nature of future network functions and intelligent applications while lowering operating costs through simplified hardware, software, and management. In this article, the question of how to achieve a successful carrier grade network with software-defined networking is raised. Specific focus is placed on the challenges of network performance, scalability, security, and interoperability with the proposal of potential solution directions.

Summary. This article begins by explaining what software-defined networking is. It accomplishes this by noting four key features, these include: (a) separation of the control plan from the data plane, (b) a centralized controller and view of the network, (c) open interfaces between the devices and controllers, and (d) programmability of the network by external applications. The article explains networking prior to SDN, and its emergence as a solution. Lastly, the key challenges to adoption are explained through presentation of four specific concepts and questions. These are:

- Performance versus flexibility: How can the programmable switch be achieved? Performance refers to the processing speed of the network node in terms of throughput and latency. Flexibility is the ability to adapt systems to support new features. This paper discusses a number of initiatives designed to enhance the programmability and performance, including specialized processors and application-specific integrated circuits (ASICs).
- Scalability: How can the controller be enabled to provide a global network view? Scalability can be divided into controller scalability and network node scalability. For controller scalability, the paper proposes the use of a peer-to-peer or

distributed controller. The addition of controllers would also allow the number of network nodes to scale as well.

- Security: How can the software-defined network be protected from malicious attack?

  There has been limited focus to date on the issues surrounding security with SDN. Both the controllers and network devices can be particularly attractive targets for attackers. Solutions to mitigate this include the use of encryption and mutual authentication between the controller and network devices.

- Interoperability: How can SDN solutions be integrated into existing networks?

  This section discusses how it would be easier to deploy a completely new infrastructure based on SDN, but the reality is most deployments will be incremental. SDN must be interoperable with existing networks to enable a smooth transition between the two. The use of open standards, like the OpenFlow protocol, are suggested to improve this interoperability.

Conclusion

The purpose of this annotated bibliography is to define the SDN approach in a way that can be understood by networking professionals, including pre-sales consultants, so that they are better able to explain this approach to potential SDN customers. Network configuration and installation requires highly skilled personnel adept at establishing and maintaining many data network elements (Sezer et al., 2013). SDN is meant to reduce the dependency on these personnel, but many pre-sales consultants working for systems integrators have limited ability to explain the benefits to customers (Westling, personal communication, 2013).

The elements, benefits, and challenges of software-defined networking essential for consultants to discuss with their customers are identified through analysis of 15 selected references, presented in the Annotated Bibliography section of this study. Information relative to the needs of pre-sales consultants is presented utilizing three categories: (a) history and explanation of SDN along with the benefits, (b) examples of SDN use cases for implementation, and (c) an explanation of the challenges associated with the adoption of SDN technology. According to Cockburn (1995), a use case is a collection of possible sequences of interactions between the system under discussion and its external actors, related to a particular goal. The system under discussion for the purpose of this paper is software-defined networking. A benefit is defined as a way to reduce dependency on personnel to establish and maintain network configuration and installation (Sezer et al., 2013). According to Hui and Koponen (2012), a challenge is defined as a hard problem in software defined networks, that negatively influences the SDN adoption process. They describe such problems as the challenges to SDN adoption, which fall into three categories: (a) technical, (b) social, and (c) economic.

History, Explanation, and benefits of SDN

In order to better explain software-defined networking (SDN) and its implications, pre-sales consultants must be able to explain SDN's place within the larger context of network technology (Westling, personal communication, 2013). To accomplish this, they should have some knowledge about its history and the motivation for its inception, the ability to provide a basic definition of what it is and what it does, along with its benefits.

Mendonca, Astuto, Nguyen, Obraczka, and Turletti's (2013) refer to current network devices as "vertically integrated black boxes", and the static architecture of the internet as "Internet Ossification" (p. 1) These descriptions imply that these Internet devices are not programmable and are thus inflexible. According to Li & Liao (2013), the key benefit of SDN, includes the rapid introduction of new network functions at software speed, and more seamless integration of the network with IT processes. Monsanto, Reich, Foster, Rexford, and Walker (2013) state that SDN creates topology abstraction, another benefit which introduces network objects that divide the network into modules, enabling objects to be hidden or protected from view.

According to Ward, network configuration state management, which is the act of managing the configurations on discrete devices, has historically remained largely static, unchanged, and commonly untouchable; as a result, manual configuration on a device-by-device basis has been the norm (as cited in Nadeau & Gray, 2013, p. xi). Feamster, Rexford, and Zegura (2013) identify the difficulties and complexity of managing modern data networks, due in part to routers and switches running complex, distributed control software that can be closed or proprietary. Casado, Koponen, Shenker, and Tootoonchian (2012) note that although multi-protocol label switching (MPLS) made advances in achieving the goals of being (a) simple, (b)

vendor-neutral, (c) future-proof, and (d) flexible, yet still had shortfalls. In order to make managing the network configuration state more dynamic, accessible, and automated, software-defined networking (SDN) is proposed (as cited in Nadeau & Gray, 2013, p. xi). SDN includes the benefit of separating the control plane from the data plane on network devices such as routers and switches, in order to make these network devices more easily programmable from a centralized location, thus improving the ability to manage the network state. (Shinde & Tamhankar, 2013).

According to Feamster, Rexter, and Zegura (2013), although this technology seems to have appeared suddenly, SDN is part of a long history of efforts to make computer networks more programmable. Casado et al. (2012) describe three interfaces relevant to network design (see Table 1).

Table 1

Interfaces for network design

Interface	**Description**
Host – Network	This interface specifies how the host, or end system, informs the network of its requirements. This is typically accomplished using an identification mechanism within the packet header (e.g. MPLS labels, QoS Markings).
Operator – Network	This interface specifies how operators, or network managers, inform the network of their requirements. Traditionally accomplished through a network manager manually configuring a network device.
Packet – Switch	This interface specifies how a packet identifies itself to a switch. This is how a switch knows how to forward a packet through the network.

In addition, multi-protocol label switching (MPLS) was an important catalyst to programmable networks; according to Casado et al. (2012):

MPLS allowed the virtualization of networks though the decoupling of core (i.e. service provider) networks from host (i.e. customer) networks. While MPLS focused on the

host—network and packet—switch interfaces, SDNs focus on the operator—network interfaces. Traditionally network devices running MPLS had to be programmed manually. The introduction of a SDN controller to the network, and application programming interfaces (API) to network devices, allows the entire MPLS network to be programmed from a centralized location (p. 86).

According to Medonca et al. (2013), the precursors to the current SDN paradigm laid the foundation for the SDN of today (see Table 2).

Table 2
Precursors to SDN

Precursor	**Description**	**Inception & Current Status**
Open Signaling (OPENSIG)	The goal was to provide access to the network hardware via open, programmable network interfaces, which led to creation of general switch management protocol (GSMP).	Working group began in 1995, GSMP version 3 published in June 2002.
Active Networking	Two main approaches were considered, namely: (1) user-programmable switches, with in band data transfer and out-of-band management channels; and (2) capsules, which were program fragments that could be carried in user messages	Proposed during mid 1990s, never achieved critical mass.
NETCONF	Management protocol proposed by the IETF Network Configuration Working Group. The protocol allowed network devices to expose an API through which extensible configuration data could be sent and retrieved.	Proposed in 2006 by the IETF. The working group is active and the latest proposed standard was published in June 2011.
ForCES	Proposed by the IETF Forwarding and Control Element Separation (ForCES) working group. Shares some common goals with SDN, but internal network device architecture is redefined, and the control software is kept in closer proximity as opposed to SDN.	Undergoing standardization since 2003. The working group is currently active.
SANE/Ethane	The immediate predecessor to OpenFlow. Focused on using a centralized controller to manage policy and security in a network.	Defined in 2006. Laid the foundation for what would become SDN.

Use Case Examples of SDN

The potential applications of software-defined networks are broad. SDNs can be deployed in data centers, enterprise, campus and wide area networks (WAN) (Yap, Huang, Dodson, Lam, & McKeown, 2010). Google is currently running a highly successful SDN developed in-house, consisting of their proprietary management software and WAN switches running the OpenFlow API (Jain, Kumar, Mandal, Ong, Poutievski, Singh, & Zhu, 2013). According to Jain, Kumar, Mandal, Ong, Poutievski, Singh, and Zhu (2013), this situation supports the notion that SDN can be utilized within a closed environment.

SDN has potential benefits to telecom networks. According Hampel, Steiner, and Bu (2013) SDN has been successfully applied to data centers and campus networks, but it has had little impact in the fixed wireline and mobile telecom domain. They provide a list of the telecom domain use cases which include: (a) IETF mobility protocols such as mobile IP, (b) mobile network architectures such as universal mobile terrestrial system (UMTS) and system architecture evolution (SAE), (c) wireline broadband networks, (d) virtual private networks and secured links, and (e) IP protocol transition.

Azodolmolky, Wieder, and Yahyapour (2013), predict that SDN will provide a boon to cloud-based networks, because it provides a new, dynamic network architecture that transforms traditional network backbones into rich service-delivery platforms.

Also according to Azodolmolky et al. (2013), there are benefits to enterprises that adopt OpenFlow-enabled SDN as the connectivity foundation for cloud connectivity, which include a

logically centralized SDN control plane which provides a comprehensive view of cloud resources and access network availability.

Challenges to SDN adoption.

The determination of what constitutes a challenge is framed by Hui and Koponen (2012), who refer to hard problems in software defined networks that negatively influence the SDN adoption process. They view such problems as falling into three categories: (a) technical, (b) social, and (c) economic. Thirteen challenges to successful adoption of SDN are identified in this annotated bibliography; all but one are technical in nature (see Table 3).

Table 3

Challenges to SDN Adoption

Challenge & Category Type	Fundamental Question	Source
Performance versus flexibility (technical)	How to balance the processing speed of the network node in terms of throughput and latency with the flexibility, which is the ability to adapt systems to support new features.	Sezer, Scott-Hayward, Chouhan, Fraser, Lake, Finnegan, & Rao (2013)
Scalability (technical)	How can the controller be enabled to provide a global network view?	Sezer et al. (2013)
Security (technical)	How can the software-defined network be protected from malicious attack?	Sezer et al. (2013)
Interoperability (technical)	How can SDN solutions be integrated into existing networks?	Sezer et al. (2013)
Building logically centralized control planes (technical)	What is the proper method for building distributed logically centralized control planes?	Meyer (2013)
The degree of centralization (technical)	To what degree should the control plane be centralized?	Meyer (2013)
Migration to SDN (technical)	How can an organization effectively migrate an entire network over to an SDN model?	Das, Caria, Jukan, & Hoffmann, (2013)
Genericity (tehnical)	Does the implementation have the ability to be used by several technologies; is it sufficiently generic?	Alberti (2012)
Performance	How to utilize high-performance platforms to	Alberti (2012)

(technical)	facilitate multiple virtual instances	
Isolation (technical	Can the implementation prevent virtual instances from interfering with one another when using shared hardware resources?	Alberti (2012)
Reconfigurability (technical)	Does the implementation have the ability for the hardware to change its functionality based upon software input?	Alberti (2012)
Lack of developer knowledge about the exposure of hardware resource ability (social)	How can the implementation overcome the developer's lack of knowledge required for exposing the hardware resources?	Alberti (2012)
Manageability (technical)	How can human intervention for management purposes be minimized?	Alberti (2012)

The most common technical challenges to SDN adoption are related to: (a) interoperability, (b) security, (c) logistics, and (d) architecture. A significant challenge to widespread SDN adoption is interoperability. Unlike the Google use case example described above, SDN has yet to be deployed into a large heterogeneous network environment, one that consists of multiple hardware vendors and/or multiple service providers (Mendonca, Nunes, Nguyen, Obraczka, & Turletti, 2013). In order to operate in heterogeneous environments, the SDN hardware and software must be interoperable and integrate easily with existing networks (Sezer et al., 2013). Another technical challenge is securing the SDN network from attack; the controller in particular is vulnerable, because it has the power to reprogram the entire network (Ortiz, 2013). Because of this power, SDN controllers are attractive targets for malicious hackers (Sezer et al., 2013). Network managers will be reluctant to implement SDNs until these concerns about security are addressed by the SDN developers.

There are also logistical challenges to SDN adoption. According to Das, Caria, Jukan, & Hoffmann (2013), a single-step complete overhaul of a fully functional network is impractical,

while at the same time, the immediate benefits of SDN are obvious. Lastly, there are a number of architectural questions about SDN that haven't been fully answered, these include: (a) how SDN networks will scale and evolve, (b) how to build distributed logically centralized control planes, and (c) the degree to which the control plane should be centralized (Meyer, 2013).

According to Guha, Reitblatt, and Foster (n.d.), software-defined networking (SDN) makes it possible to control an entire network in software, by writing programs that tailor network behavior to suit specific applications and environments. SDN offers many potential benefits and some as yet unresolved challenges. These challenges represent significant problems that must be addressed in order to meet the expectation that SDN presents the best option for the future of highly optimized and ubiquitous application-driven networks (Sezer et al., 2013).

References

Alberti, A. M. (2012). Software-defined networking: Perspectives, requirements, and challenges. Retrieved from http://www.researchgate.net/publication/236156555_Software-Defined_Networking_Perspectives_Requirements_and_Challenges/file/e0b495167ed2eeef7e.pdf

Azodolmolky, S., Wieder, P., & Yahyapour, R. (2013). SDN-based cloud computing networking. In *Transparent Optical Networks (ICTON), 2013 15th International Conference on* (pp. 1–4). Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6602678

Bell, C., & Frantz, P. (2013). Critical evaluation of information sources. *University of Oregon.* Retrieved from http://library.uoregon.edu/guides/findarticles/credibility.html

Casado, M., Koponen, T., Shenker, S., & Tootoonchian, A. (2012). Fabric: A retrospective on evolving SDN. In *Proceedings of the first workshop on Hot topics in software-defined networks* (pp. 85–90). New York, NY, USA: ACM. doi:10.1145/2342441.2342459

Cockburn, A. (1995). Structuring use cases with goals. Retrieved from: http://www.uio.no/studier/emner/matnat/ifi/INF5120/v04/undervisningsmateriale/UseCases.pdf

Das, T., Caria, M., Jukan, A., & Hoffmann, M. (2013). A Techno-economic Analysis of Network Migration to Software-Defined Networking (arXiv e-print No. 1310.0216). Retrieved from http://arxiv.org/abs/1310.0216

Dix, J. (2013). Understanding software-defined networking. (2013, May 13). *Network World*. Retrieved November 4, 2013, from http://www.networkworld.com/news/2013/051313-sdn-spotlight-269656.html

Dosanjh, J. (2012). How to sell software-defined networking. *CRN*. Retrieved November 6, 2013, from http://www.crn.com/blogs-op-ed/channel-voices/232601530/how-to-sell-software-defined-networking.htm

Feamster, N., Rexford, J., & Zegura, E. (2013). The road to SDN: An intellectual history of programmable networks. Retrieved from https://www.cs.princeton.edu/courses/archive/fall13/cos597E/papers/sdnhistory.pdf

Florentine, S. (2013, October 17). SDN adoption puts DevOps Pros in high demand. *CIO*. Retrieved November 6, 2013, from http://www.cio.com/article/741556/SDN_Adoption_Puts_DevOps_Pros_in_High_Demand

Guha, A., Reitblatt, M., & Foster, N. (n.d.). Formal foundations for software-defined networks. Retrieved from: http://www.opennetsummit.org/pdf/2013/research_track/poster_papers/final/ons2013-final62.pdf

Hampel, G., Steiner, M., & Bu, T. (2013). Applying software-defined networking to the telecom domain. Retrieved from http://moritzsteiner.de/papers/SDNtelecom.pdf

Hui, P., & Koponen, T. (2012). Software-defined networking. Retrieved from http://vesta.informatik.rwth-aachen.de/opus/volltexte/2013/3789/pdf/dagrep_v002_i009_p095_s12363.pdf

Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A. & Zhu, M. (2013). B4: Experience with a globally-deployed software defined WAN. *In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (pp. 3–14)*. Retrieved from http://dl.acm.org/citation.cfm?id=2486019

Lenrow, D. (2012). Why software-defined networking matters: Single point of management. *SDNCentral*. Retrieved November 6, 2013, from http://www.sdncentral.com/technology/why-sdn-matters-single-point-of-management/2012/04/

Levin, D., Canini, M., Schmid, S., & Feldmann, A. (2013). Incremental SDN Deployment in Enterprise Networks. Retrieved from http://www.net.t-labs.tu-berlin.de/~marco/papers/panopticon.p-sigcomm13.pdf

Li, C.-S., & Liao, W. (2013). Software-defined networks [Guest Editorial]. *IEEE Communications Magazine*, *51*(2), 113–113. doi:10.1109/MCOM.2013.6461194

Lim, J., & Obraczka, K. (n.d.). Software-defined networking and services. Retrieved from http://surf-it.soe.ucsc.edu/sites/default/files/Lim_surfit12_report.pdf

Mendonca, M., Astuto, B. N., Nguyen, X. N., Obraczka, K., & Turletti, T. (2013). A survey of software-defined networking: Past, present, and future of programmable networks. Retrieved from http://hal.inria.fr/hal-00825087/

Mendonca, M., Nunes, B., Obraczka, K., & Turletti, T. (2013). Software-defined networking for heterogeneous networks. *IEEE COMSOC MMTC E-Letter*. Retrieved from http://www.comsoc.org/~mmc/

Merriam-Webster (2013). Definition for challenge. Retrieved from: http://www.merriam-webster.com/dictionary/challenge

Meyer, D. (2013). The Software-Defined-Networking Research Group (SDNRG). Retrieved from http://www.1-4-5.net/~dmm/papers/IC-17-06-Standards.pdf

Monsanto, C., Reich, J., Foster, N., Rexford, J., & Walker, D. (2013). Composing software defined networks. *NSDI, Apr*. Retrieved from https://www.usenix.org/system/files/conference/nsdi13/nsdi13-final232.pdf

Murray, A. (2013). HP predicts significant SDN adoption by 2015. *Network Computing*. Retrieved November 5, 2013, from http://www.networkcomputing.com/data-center/hp-predicts-significant-sdn-adoption-by/240162151

Nadeau, T., & Gray, K. (2013). SDN: Software Defined Networks. Sebastopol, CA: O'Reily Media

Ortiz, S. (2013). Software-defined networking: On the verge of a breakthrough? *Computer*, *46*(7), 10–12. Retrieved from: http://www.computer.org/csdl/mags/co/2013/07/mco2013070010-abs.html

Ren, Y. (2011). Building information modeling integrated with electronic commerce material procurement and supplier performance management system. Retrieved from: http://drum.lib.umd.edu/bitstream/1903/12349/1/Ren_umd_0117N_12803.pdf

Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., & Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, *51*(7), 36–43. doi:10.1109/MCOM.2013.6553676

Shenker, S. (2011). The future of networking, and the past of protocols. *Open Networking Summit*. Retrieved from http://www.opennetsummit.org/archives/apr12/site/talks/shenker-tue.pdf

Shinde, M. B., & Tamhankar, S. G. (2013). Review: Software Defined Networking and OpenFlow. Retrieved from http://www.isroset.org/pub_paper/IJSRNSC/ISRSOET-IJSRNSC-00085.1.pdf

Yap, K.-K., Huang, T.-Y., Dodson, B., Lam, M. S., & McKeown, N. (2010). Towards software-friendly networks. In Proceedings of the first ACM Asia-Pacific workshop on systems (pp. 49–54). Retrieved from: http://doi.acm.org/10.1145/1851276.1851288

Westling, M. (2013). Personal Communication