Software Defined Networking Architecture, Security and Energy Efficiency: A Survey

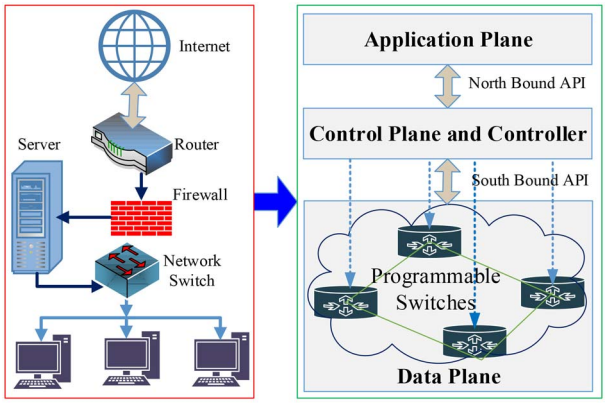
Danda B. Rawat, *Senior Member, IEEE*, and Swetha R. Reddy, *Member, IEEE*

***Abstract*—Software-defined networking (SDN) is an emerging paradigm, which breaks the vertical integration in traditional networks to provide the flexibility to program the network through (logical) centralized network control. SDN has the capability to adapt its network parameters on the fly based on its operating environment. The decoupled structure of SDN serves as a solution for managing the network with more flexibility and ease. In SDN, the centralized cost effective architecture provides network visibility which helps to achieve efficient resource utilization and high performance. Due to the increasingly pervasive existence of smart programmable devices in the network, SDN provides security, energy efficiency, and network virtualization for enhancing the overall network performance. We present various security threats that are resolved by SDN and new threats that arise as a result of SDN implementation. The recent security attacks and countermeasures in SDN are also summarized in the form of tables. We also provide a survey on the different strategies that are implemented to achieve energy efficiency and network security through SDN implementation. In an effort to anticipate the future evolution of this new paradigm, we discuss the main ongoing research efforts, challenges, and research trends in this area. With this paper, readers can have a more thorough understanding of SDN architecture, different security attacks and countermeasures, and energy efficiency.**

***Index Terms*—Software defined network, network virtualization, energy efficiency, SDN security, OpenFlow network.**

I. Introduction

THE INFORMATION and Communication Technology (ICT) infrastructures are expanding continuously with the increase in a number of devices and applications for Internet-of-Things (IoT) [1] and cyber physical systems [2]. Software-Defined Networking (SDN) is regarded as a technology that is capable of managing the entire network efficiently and transforming the complex network architecture into the simple and manageable one. Recent studies have shown that the traditional networks are not capable of satisfying the growing demands as all components in the network are vertically integrated forming a complex structure that is hard to manage [3]–[6]. Traditional networks are capable of supporting only vendor specific policies and offer no flexibility for dynamic network environment [3], [5], [6].



(a) Traditional Network Architecture (b) SDN Architecture

Fig. 1. Comparison of traditional network architecture and simplified software defined network (SDN) architecture.

Fig. 1 depicts the comparison of high level transformation of traditional networking architecture and SDN architecture. SDN is regarded as the hardware independent next generation networking paradigm in which networking device from any vendors could be controlled through SDN. SDN has decoupled application plane, data plane and control plane. It has two prime components: *controller* and *switches*. SDN controller is responsible for the management of entire network whereas networking switches are responsible for operating based on the instructions deployed through SDN controller. Unlike the traditional networks, where the entire system needs to be reconfigured to upgrade the system, only the software needs to be updated in the SDN, making it more convenient for upgrading and reduction of overall cost [7]. SDN has gained significant attention as it is flexible and can be programmed using any high-level programming languages to serve the purpose of the client devices and users [8]–[11]. Programmable SDN allows to adapt the network parameters based on its operating environment to improve overall network performance as well as detect flaws in the network. Due to the increasingly pervasive existence of smart programmable devices in the network, SDN provides security (e.g., SDN IPS [12], TIPS [13], anomaly attacks [14]), energy efficiency and network virtualization for enhancing the overall network performance.

Manuscript received March 4, 2016; revised May 19, 2016 and September 11, 2016; accepted October 15, 2016. Date of publication October 19, 2016; date of current version February 22, 2017. This work of D. B. Rawat was supported in part by the U.S. National Science Foundation (NSF) CAREER Award under Grant CNS-1650831, and in part by the NSF under Grant CNS-1658972.

D. B. Rawat is with the Department of Electrical Engineering and Computer Science, Howard University, Washington, DC 20059 USA (e-mail: db.rawat@ieee.org).

S. R. Reddy is with the Department of Electrical Engineering, Georgia Southern University, Statesboro, GA 30460 USA.

Digital Object Identifier 10.1109/COMST.2016.2618874

Despite the advantages that programmable SDN technology offers, it can be vulnerable to many kind of attacks such as anomaly attacks, intrusion attacks, Denial-of-Service (DoS) attacks, etc. [15]–[18]. Furthermore, programmable SDN can be used to help reduce energy consumption by network infrastructure and devices while providing network security. According to the reports on the energy consumption, the energy consumption by the ICT infrastructure is proliferating [19] and it is expected to consume up to 14% of the worldwide energy consumption by the end of 2020 [20], [21]. This necessitates an effort and need to reduce the energy consumption in the networks. SDN platform is capable of providing high performance along with energy efficiency and network security. However, the energy efficiency methods have to be implemented in SDN to control the inflating energy consumption while providing network security. The proper traffic management and implementation of sleep-awake mechanisms in SDN can reduce the overall power consumption of the network and thus the energy bills [22]–[25].

Although there are related survey papers [6], [7], [16], [26]–[29] which provide information about the SDN, more up to date activities of rapidly advancing research areas on SDN security and energy efficiency is to be brought to the research community. Furthermore, state-of-the-art literature does not provide recent advances on security threats/attacks and countermeasures in SDN by categorizing them in terms of their types and recent advances in energy management schemes in SDN. The aim of this survey paper is to provide the recent works towards the security aspects of SDN with a more focus on various security threats that are resolved by SDN and new threats that arise as a result of SDN implementation; and various strategies that are implemented to achieve energy efficiency in SDN. Further, we present various security threats that are resolved by SDN and new threats that arise as a result of SDN implementation. We also provide a survey on the different strategies that are implemented to achieve energy efficiency in the networks through SDN implementation. The main contributions of this paper include:

- We present a detailed study on SDN security aspects by categorization them into two parts. The *first* part discusses the threats that can be resolved by the implementation of SDN and the *second* part discusses security threats because of SDN implementation along with their countermeasures.
- We summarize security attacks and countermeasures in SDN in a tabular form for a side-by-side comparison.
- We provide a survey on recent techniques to reduce energy consumption in SDN and present side-by-side comparison of different techniques for energy saving through SDN.
- We present a discussion of research challenges, open problems and recommendations for SDN that are needed to be addressed to realize its full potential.

The organization of the paper is as follows. Section II discusses about the background of SDN. Security and Energy Efficiency are presented in Sections III and IV respectively.

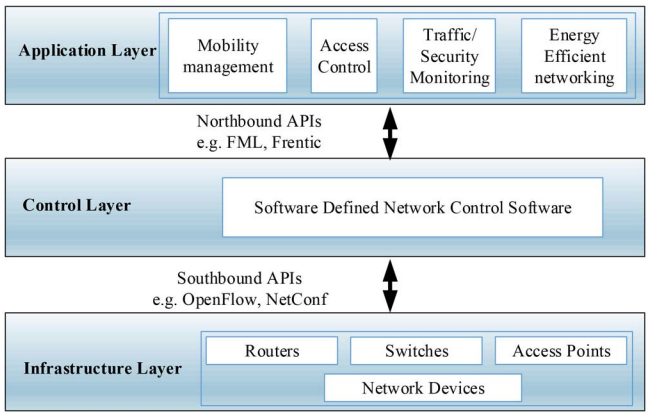


Fig. 2. Block diagram of software defined network (SDN) architecture.

Challenges in SDN are discussed in Section V. Finally, the conclusions are presented in Section VI.

II. Background of SDN

The development of SDN has begun in early 1990's with the ideas obtained from various supporting technologies [30]–[33]. The idea of decoupling of the data plane and control plane was obtained from the network control point implemented in telephone networks where the data and control are fragmented from one another. This proved to be a cost effective and secure solution. Active networks [34], [35] introduced the programmability in the network through an application interface. The capsule model [36], Tempest [37], Virtual Network Infrastructure (VINI) [38], and programmable router switch model [34], [39] are the examples of active networking models that offer the flexibility to control different tasks and events. Though these models were discovered much earlier they could not be implemented due to the lack of proper infrastructure and hardware support. The major contribution for SDN started with the introduction of OpenFlow in 2008 [40].

A. The SDN Architecture

SDN is regarded as an emerging technology where the primary concept is removing the intelligence from the networking devices and managing the entire network functionality with the help of a centralized controller. The basic structure of SDN is represented in Fig. 2. It consists of three different layers: Application layer, Control layer and Infrastructure layer.

1) Infrastructure/Data Layer: Infrastructure layer (*aka* data layer) in SDN comprises of network devices such as router, switch and access point. Both virtual switches such as Open vSwitch, Indigo, Pica8, Nettle, Open Flowj and physical switches coexist in this layer [6], [7], [41]–[43]. The main function of the data plane is forwarding the packets according to the assigned rules/policies.

2) Control Layer: Control layer consists of a controller which controls the overall SDN functions. This layer acts as a mediator for infrastructure layer and application layer. The controller is responsible for managing the entire traffic flow and solely takes decisions on routing, flow forwarding and packet dropping through programming [9], [26], [44]. The controllers in the distributed environment communicate with each

other through east-bound and west-bound interfaces. The control layer and the infrastructure layer communicate with each other through south-bound API such as OpenFlow, NetConf, etc. [7].

3) Application Layer: The Application layer is the foremost layer in the SDN as shown in Fig. 2. It is responsible for handling software related business and security applications. Network virtualization, Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), firewall implementation and mobility management are some examples of applications handled by this layer. This layer communicates with the control layer using Application Control Plane Interface (A-CPI) also called as northbound application interface [45] as shown in Fig. 2.

B. Features of SDN

SDN helps to achieve high performance and flexibility compared to the conventional networks. SDN offers a *centralized control* of entire network through a controller which manages and automates the functions in all networking elements such as switches, routers and firewalls using programs. This feature helps to immediately respond to the changing traffic patterns and dynamically adjust the flows to enhance the overall network performance, better security and energy efficiency. Another feature of SDN is *openness for innovation—open source platform* which helps to improve further exploration, innovation and development of the network. Traditional networks are vendor specific and cannot be modified. Note that the openness of SDN does not affect the performance of the network [46]. *Abstraction through layers* is one of the best features enabled in SDN to reduce the burden on the programmer where different layers are interfaced through APIs [47], [48]. Due to this feature, the application layer has no direct interaction with physical components of the controller. Abstraction tools like Frentic [47] and pyretic [49] are among the most commonly used ones. Virtualization through SDN enables the sharing of physical infrastructures among multiple users and adaptibility in the networks [6], [50]. A typical virtualization architecture is shown in Fig. 3. The virtualization can be storage based, network based or server based [50]. Vmware, Microsoft, Hyper-v, Citrix, Xen server, RHEL are among the most popular companies for providing network virtualization platform [6], [11].

Implementation of virtualization aims at creating a suitable environment for all SDN clients to coexist in a single platform without interfering with each other. The Networking Virtualization Proxy (NVP) by VMware [51] and SDNVE by IBM [52] are network virtualization platforms that are used in current SDN. The NVP specializes in reducing the programming complexity by providing high level abstraction whereas SDN is capable of handling large number of virtual machines [52]. Some of the commonly used virtualization techniques are discussed below. *FlowVisor (FV)* has proposed as a switch virtualization scheme that establishes communication between virtual layer and data plane [53]. Switches in system are embedded with a FlowVisor (FV) slicer and FV classifier [53]. Furthermore, each FV slicer is managed by a different controller [53]. ADvanced Flowvisor (ADvisor) was developed to overcome the limitations of FlowVisor and provide proper isolation among the clients [54]. *AutoSlice* has been proposed in [55] to tackle the issue of scalability and constraints associated with flow. *Carrier Grade Virtualization* has been proposed in [56] to include translation agents in all networking elements present in infrastructure layer. This agents facilitate the direct communication between physical layer and clients' controller without involving hypervisors. *Virtualization Cloud Platform (VCP)* architecture in SDN relies greatly on Network Operating System (NOS) to provide isolation and proper utilization of network resources [57]. *FlowN* has been proposed in [58] for enabling the flexibility in NOX controller. It is a container based virtualization supporting multiple clients on a single platform [59]. HyperFlex has been proposed as a control plane virtualization scheme in SDN which mainly focuses on achieving privacy, flexibility and scalability [60].

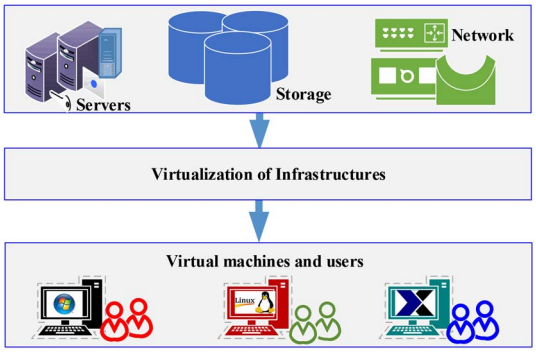


Fig. 3. A typical architecture for server, network, storage virtualization.

C. Applications of SDN

Data centers are the major places used for storage of information/data. The structure of data centers in conventional networks is complex and difficult to manage. The available resources are not properly utilized as it does not have a complete view of the network. The data centers implemented using SDN can dynamically adapt by expanding or shrinking its size based on the requirement. Global view feature in SDN can be used for efficient traffic management to improve the overall network performance and resource utilization, and reduce the energy consumption of the network. Upgrading in traditional networks is not only associated with a cost of replacing the entire network but also involves the chance of losing important information. SDN is capable of upgrading the data-centers without loosing the information, changing the overall work flow in the network and degrading the overall performance [61]–[63].

The usage of SDN in WAN gained significant attention with Google deploying it for managing their data centers. Google employed traffic engineering schemes and were able to get benefits such as fault tolerance, active link utilization up to 100% promoting energy efficiency [64]. Software driven WAN (SWAN) offers flexibility to optimize the network policies and handle flows based on the priority set by the client and

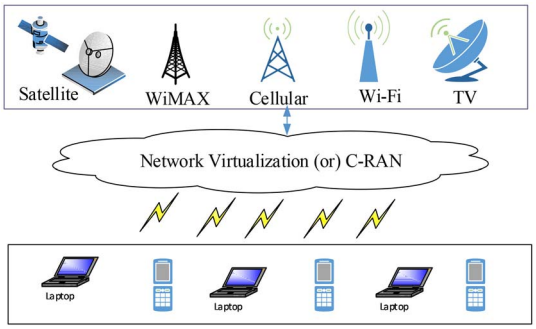


Fig. 4. Top-level wireless virtualization architecture with SDN [71], [81], [82].

was able to offer 38% more throughput than MLPS (Multi Layer Protocol Switching) scheme [65]. Note that the MLPS traffic engineering was the most commonly adopted method in conventional networks. Exponential growth of connected devices for IoT and cyber-physical systems [66] can be handled by SDN by expanding and/or shrinking resources on the fly [6], [67], [68].

SDN in Cellular Networks offers centralized control for cost-effective solutions for reducing Capital expenditures (CAPEX) and operating expenses (OPEX) of the cellular network. Combination of cellular networking technologies and SDN can offer high data rate and better QoS to the users [69], [70]. One of the major benefits is its capability to allow switching between different wireless technologies to provide better connectivity and coverage [70], [71]. Failure recovery, availability, dependability and security can be improved in cellular system [72], [73].

SDN for Wi-Fi Network effectively guarantees airtime shares to network slices using a Time Division Multiple Access (TDMA) like airtime scheduling [74]. Furthermore, SDN provides the flexibility for the deployment of Wi-Fi infrastructure in the rural areas where the traditional networks are hard to deploy and manage [74], [75].

SDN offers network interface for heterogeneous transport networks through virtualization (e.g., [76]) and for optical network virtualization in [77]. The management and allocation of resources to virtual networks/machines is simplified by introducing ANM (Automatic Network Management) systems in the design [78]–[80].

In order to have seamless switching (*aka* wireless virtualization) between different wireless technologies such as satellite, cellular, TV, GPS, Wi-Fi, etc., Cloud-base Radio Access Networks (C-RAN) has been proposed in [81]. A typical block diagram for C-RAN is shown in Fig. 4.

SoftRAN has been proposed to provide reliable connectivity and coverage with available spectrum in [83]. Furthermore, for Self Organizing Network (SON) in C-RAN, Heterogeneous Cloud Radio Access Networks (HC-RAN) are capable of supporting the decoupling without degrading the overall network performance [84], [85]. Similarly, OpenRAN has been proposed for deployment of SDN in C-RAN for meeting the QoS requirements using Wireless Spectrum Resource Pool (WSRP), SDN controller and Cloud Computing Resource Pool (CCRP) [86].

Due to ease to modify the system and adapt various energy efficient mechanisms, SDN provides a better solution for green networking which has become important in network design and deployment for economic and environmental benefits. For green networking, there have been different approaches considered [87]. It is noted that SDN switching devices may not directly help reduce energy consumption in the SDN-based networks [88], however SDN offers configurations that can help to reduce energy consumption [87], [89]. Further details for green networking using SDN are presented in Section IV of this paper.

III. Security and Software-Defined Networks

Initially SDN was introduced for the flexibility to configure the network to improve overall network performance but later SDN is found to be applicable to secure the network. Traditional networks are vulnerable to various security threats some of which cannot be detected easily. SDN is emerging technology that can provide security defense solutions as it is capable of detecting attacks and acting adaptively in a quicker way than traditional networks. However, there are new kind of attacks targeted to SDN [90]. Thus, SDN implementation in the infrastructure has both pros and cons.

The security aspects of SDN are discussed in two sections as depicted in Fig. 5: the *first part* (Section III-A) discusses about the role of SDN to enhance security in existing network infrastructure. The *second part* (Section III-B) presents the security challenges that arise in SDN.

A. SDN as a Security Solution

The various security threats that can be resolved using SDN are summarized in the Table I. In this table, we provide definition of different kinds of threats along with their countermeasure techniques. The detailed description of these countermeasures and applications are discussed below.

1) SDN as Intrusion Detection System (IDS) and Intrusion Prevention System (IPS): An intrusion attack is an unauthorized activity on a network where attacks absorb network resources intended for other uses. Due to the reconfigurablity and programmability of SDN, the SDN can be implemented as IDS and IPS to monitor the network activities continuously to detect intrusion attacks. Most common intrusion attack vectors that can be defended by using adaptability and programmability of SDN are

1) *Asymmetric Routing Attack:* The attacker use more than one route to the targeted network device to bypass certain network segments and intrusion sensors. If networks is not set up for asymmetric routing, they are vulnerable to this attack.
2) *Buffer Overflow Attacks:* This attack overwrites specific sections of device memory of a target network or replaces normal data in certain memory locations with a malware to attack the network with an aim of initiating denial-of-service.
3) *Protocol-Specific Attacks:* The network protocols - such as TCP, UDP, ARP, IP, ICMP etc., may inadvertently

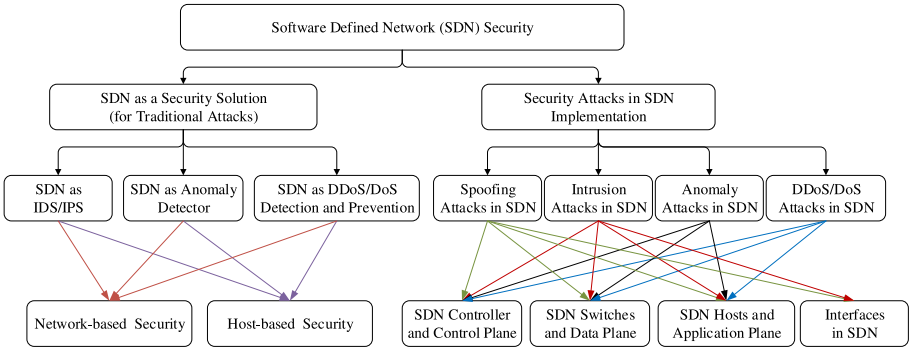


Fig. 5. Software defined network (SDN) as a security solution and security attacks that are unique to SDN.

TABLE I
TYPICAL SECURITY APPLICATIONS OF SDN AS A SECURITY PARADIGM

Attacks	Definition	Applications	Proposed Defense Techniques
Intrusion Attacks	The intrusion attack is creating the access into the system without the valid permissions and compromising the security and stability of the system.	1. Cloud Computing 2. Smart grids 3. Virtual machines	ClOUDWATCHER [91], Snort-SDN [92], TIPS [13], SDNIPS [12]
Anomaly attacks	The attack caused by a unknown user by violating policies and compromising the entire system.	1. Cloud Computing	Graphical based Approach [93]
DDoS attacks	The DDoS attack is a kind of attack where the system is attacked from the multiple sources in the distributed manner creating a denial of service to the valid users	1. Cloud computing 2. Mobile infrastructure	Mobile malware detection [94], RTBH [95], Botnet detection scheme [96], privacy aware mitigation of DDoS attacks [97]

leave a backdoor for network intrusions through spoofing or so with an aim of compromising or even crashing the targeted devices on a network. For instance, while mapping IP network addresses to the hardware addresses, ARP protocol does not perform authentication on messages, allowing attackers to execute "man-in-the-middle" attacks.

4) *Traffic Flooding Attacks:* Attacker can generate traffic loads too heavy for the network to overwhelm the overall network resources. These attacks can be easily controlled using SDN.
5) *Trojan based Attack:* This attack instigates DoS attacks, erase stored data or open back doors to permit system control by outside attackers.

There are different defense solutions based on SDN which are discussed below. CloudWatcher has been proposed in [91] for controlling the traffic flow in SDN with program logic and efficiently routing it through all security components present in the infrastructure such as Network IDS and firewalls. This prevents the entry of malicious packets that may pose a threat to the network.

Network Intrusion detection and Countermeasure sElection in virtual network systems (NICE) has been proposed in [98] for both intrusion detection and prevention. NICE has four modules: NICE-A, VM profiling, attack analyzer and a network controller. The NICE-A works as a network IDS, VM profiling stores the complete activities of VMs (including traffic conditions, open ports, vulnerabilities and security alerts etc.), attack analyzer is responsible for analyzing attacks and providing countermeasures, and the network controller helps the analyzer by reporting the complete information of the network conditions.

On-path detection and off-path detection approaches have been proposed in [99]. In on-path detection, the suspicious packets are detected by attaching the IDS system in the path of travel of packets. It is more effective than the off-path detection where the IDS is attached as a separate physical module to the system. The other security feature is the ability of the IDS to report suspicious activities to the controller using alerts/alarms so that the controller can immediately take an action to mitigate the attacks.

IDS and IPS have been integrated with SDN in [12], [13], and [92] to analyze attacks in a network and supply suitable countermeasures for the attacks. The network controller is utilized to gather the required information for the attack analyzer to detect these threats/attacks.

SDN is implemented along with a prominent IDS system called Snort [100] for the detection of threats in Advanced Metering Infrastructure (AMI) which are popular in smart energy grids. The standalone IDS cannot prevent the malware from entering the system, so SDN is embedded with it to guard and protect the system. Snort detects the malware based on predefined rules. These are different methods of incorporating snort with SDN including mirror implementation and PACKET_IN approach [100]. In mirror implementation, Snort is connected to the OpenFlow switch in SDN where all the traffic is made to pass through both OpenFlow switch and snort for detection of suspicious activity. In PACKET_IN approach, snort runs as a background application connected to the OpenFlow controller and only suspicious activity are reported to the controller. The limitation of these methods is that there might be flooding of traffic in the network. The new method for integration is proposed in [92] where the rules of snort are incorporated into OpenFlow switches and snort

comes into play only when there is some suspicious activity in the system. The additional features are also embedded in OpenFlow which includes management server to controller and policy checking agents in switches. Transparent Intrusion Prevention Systems (TIPS) [13] has been proposed to prevent intrusion attacks by integrating SDN and poll-mode packet processing [13]. SDN-IPS has been proposed in [12] to prevent the intrusion attacks in a network with high efficiency.

2) SDN for Anomaly Detection: Anomaly detection (or outlier detection) in a network is the identification of events or observations which do not conform to an expected pattern. These days attacks are becoming more sophisticated which makes it hard to trace the actual origin of the attack. SDN technology gives us a privilege to configure the devices to serve our need. For instance, home router that is configured using SDN works effectively to detect the malware and spyware attacking the system [14]. A graphical approach has been proposed in [93] that relies on OpenFlow based switches to trace-back the origin of attacks where all paths that are vulnerable to anomaly attacks can be determined.

With the implementation of SDN, collaborative detection can be implemented through already existing centralized SDN controller where each switch or host reports its attack detection decision to the centralized controller. For binary decision variable $d_i \in \{0, 1\}$ of each switch/host $i = 1, 2, \ldots, N$, to make a decision (D) about the attack, SDN controller can use logical AND operation ($\cup$) as

$$D = \cup_{\forall i} d_i \tag{1}$$

or logical OR operation (Π) as

$$D = \Pi_{\forall i} d_i. \tag{2}$$

Note that the AND operator in (1) says there is an attack when $d_i = 1$, $\forall i$ and thus this approach is more restrictive/conservative. Whereas the OR operator in (2) says there is an attack when any one of d_i's is true making it the least conservative. Thus, alternative approach could be the majority based decision that is given as

$$D = 1 \text{ if } \sum_{i=1}^{N} d_i > \frac{N}{2}, \quad d = 0 \text{ otherwise}, \tag{3}$$

which could be a more appropriate scheme to enhance the performance of anomaly detection.

3) SDN for Distributed Denial-of-Service (DDoS) Attack Detection and Prevention: DDoS attacks deny legitimate users to get access to network services. These attacks can cause a significant damage by compromising the entire network [101]. Conventional networks have some methods to detect DDoS attacks and protect the networks but do not offer very reliable and flexible defense solutions [102]. Due to the programmable features and reconfigurable nature of SDN, flexible and robust approaches can be designed, deployed and evaluated to detect and prevent DDoS attacks.

The mobile devices have become more powerful compared to the past and usage of these devices have been exponentially increasing. This increases the chance of attacks including DDoS attacks in the network. Mobile malware detection approach has been proposed in [94] where mobile traffic from the access points is directed to the controller attached with a malware detector. Four algorithms for detecting malware are given below.

1) *IP Blacklisting:* A list of all suspicious IP addresses is maintained in the system. When switches send unmatched packets to OpenFlow controller, it verifies the IP address to see if it is from the blacklist and drops the packet if IP is found in that list.
2) *Connection Success Ratio:* If the number of unsuccessful connections of the users exceeds the fixed threshold value then the user is identified as malicious one.
3) *Throttling Connection:* The malicious device/host trying to attack many systems is identified based on the Recently Accessed Host (RAH) list maintained in the system. If the waiting list of the host exceeds a fixed threshold value then the user is identified as a malicious one.
4) *Aggregate Analysis:* If one host in the network is compromised by malicious activity then security of the other users in the network is also at risk. This algorithm works for detection of other infected hosts based on the similarities (i.e., connection time, destination and single platform).

The integration of SDN for the mobile cloud infrastructure has been further explored in [101] for designing a sophisticated mechanisms for protecting the network. The prime cause for the occurrence of DDoS attack in system is due to botnets. The protocol for easy recovery from botnet DDoS attacks is developed in [96] where the SDN controller is extended with DDoS blocking module. In [95], SDN has been exploited to use Remote Triggered Black Hole (RBTH) approach for the prevention of DDoS attacks. The SDN controller plays a major role in detecting the malicious traffic routed from OpenFlow switch and discards them to prevent further damage to the network. Furthermore, distributed collaborative framework has been proposed in [97] to enable autonomic mitigation of DDoS attacks by avoiding privacy leakage and other legal concerns.

B. Security Attacks in SDN and Countermeasures

As discussed, SDN offers defense solutions for various security attacks through its programmability features. However, there are several new threats that arise as a result of SDN implementation. Note that attacks are prevalent in SDN as it is mostly dependent on the programs/software for defining its behavior which may keep the security of the entire system at stake making it feasible for the attackers to enter the system. In this section, we present security attacks, challenges and countermeasures in SDN. Security vulnerabilities in SDN can jeopardize the entire network and degrade the performance. The attacks on SDN may occur in different modules such as controller, virtual machines and OpenFlow switches [6]. There are several attacks that arise along with SDN implementation [103]. Typical attack vectors and their locations of occurrence in SDN are shown in Fig. 6.

TABLE II
SECURITY ATTACKS AND COUNTERMEASURES IN SDN

Attack	Affected layers	Result of Attack	Proposed Defense Techniques
Spoofing Attacks	1. Data Layer 2. Data control layer Interface	1. Rule modification 2. Leakage of data 3. Malicious applications	OF-RHM [104], TopoGuard [105], Malware switch detection [106], Sphinx [107]
Intrusion Attacks	1. Data Layer 2. Control Layer 3. Data control layer interface 4. Application layer 5. Application control layer interface	1. Malicious Applications 2. Unauthorized entry 3. Leakage of Data 5. Entry of bugs 6. Application failures	Flowguard [108], Flowtag architecture [109], NIMBLE [110], Slick [111], NICE [112], FortNOX [113]
Anamoly attacks	1. Application Layer 2. Control layer 3. Data Layer	1. Leakage of data 2. Modification of data 3. Unauthorized access	FRESCO-DB [15], FortNOX [113], TRW-CB [114], Rate Limiting [115], Maximum entropy [116], NETAD [117]
DoS and DDoS Attacks	1. Control Layer 2. Data Layer 3. Data-control Interface	1. Denial of Service 2. Flooding in Controller 3. Flooding of flow-entries	Entropy method [118], FRESCO-DB [15], Avant guard [17], FortNOX [113], Lightweight DDoS detection [119], Z3 Prover method [120], LineSwitch method [121]

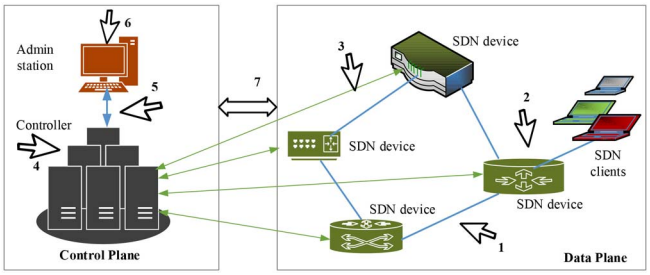


Fig. 6. Threat vectors in SDN.

The description of each threat vector given in Fig. 6 is given below.

- Threat vector **1** represents the fake traffic flows that occur during intercommunication of SDN devices in data plane. The attack may occur using the spoofed identity of legitimate flows or fake identity of the device.
- Threat vector **2** represents the attacks at SDN switches in data plane which may occur from the in-flow and out-flow of traffic.
- Threat vector **3** represents the attacks that may occur during the communication of data plane devices with control plane device (controller).
- Threat vector **4** represents the attacks at the controller.
- Threat vector **5** represents the attacks that occur between controller and application layer devices (including admin systems).
- Threat vector **6** represents the attacks on administrator's station which is linked with the controller.
- Threat vector **7** represents the attacks targeting communications between data layer and application layer.

To protect from these threat vectors, defense solutions have been proposed in [103] but the most of which are still open to be explored. Table II discusses various attacks in SDN occurring in different layers and provides the results of occurrence of these attacks along with mitigation techniques to prevent these attacks. The description of threats in different layers is discussed in detail below.

1) The controller is the most obvious target for attackers as the entire network functionality and behavior depends on controller. Once the controller is compromised it can pose a threat to entire network. The threat vectors 3, 4 and 5 are associated with control plane. The attacks can be directed from various modules which includes switches in data plane layer, northbound interface, southbound interface and application layer. DoS/DDoS, anomaly and intrusion attacks are the possible attacks that can occur in control plane.
2) The switches in SDN are only capable of performing minimal tasks such as forwarding the packets. However, the threat posed to switches may cause a huge damage to entire network. The threat vector 1 and 2 are associated with vulnerabilities in switches. DoS, DDoS, spoofing attacks and intrusion attacks are some attacks associated with data plane.
3) The threat vectors 5, 6 and 7 are related to attacks in application layer. The software related attacks such as bugs, failure of applications, malicious applications injection, anomaly and intrusion attacks are most common threats in this layer.
4) The threat vectors 3, 5 and 7 are associated with the attacks on interfaces. The interfaces play a important role in enabling communication between two planes. Most of the attacks on the interfaces are similar to attacks that occur in other planes. If the interface is compromised it allows back and forth flow of malicious traffic in the system.

The detection and mitigation techniques for the above discussed attacks are presented in detail below.

1) DDoS and DoS Attacks in SDN: The SDN controller plays a crucial role in determining the functionality of SDN architecture, thus the controller has become one of the main targets for DDoS/DoS attacks. Some vulnerabilities in FloodLight based controllers lure DDoS/DoS attacks [121], [122]. The links between switches and controller is the point of interest for conducting these attacks. These attacks can be mitigated by enabling strict Transport Layer Security (TLS) authentication mechanisms in communication links between switches and controllers and also prioritizing the pre-existing connections over the new connection.

Note that DDoS attacks can be detected using techniques proposed for conventional network, however same defense techniques proposed for traditional networks cannot be implemented directly in SDN due to the differences in the architecture. Ideas proposed for traditional networks can be borrowed while designing attack detection techniques for SDN.

Typical DDoS attacks are untraceable as they are carried out by botnets with automated actions. To discover the DDoS attacks, there are different approaches [123], [124] which could easily detect botnets. Protocols and IP addresses can be verified to detect DDoS attacks, however botnets can spoof the identity by faking legit addresses. In this case, the detection system might not be able to detect attacks since protocols and IP addresses are faked by attacker that seem legitimate for malicious legitimate users or botnets. Furthermore, DDoS attacks could occur at random interval and random time, and they are persistent. Important DDoS attack defense solutions are briefly discuss below.

1) *Attack Pattern Recognition [125]:* If the attack is happening at particular intervals, such as at any given date and time and repeating within similar intervals such as year or months, then the pattern of attack can be recognized. We can also gauge the duration of attacks and how long these attacks last. The nature of attack and the packets of attacks can give us a hint of what kind of attack is being carried out. If these information can be logged to create a database from the prior experiences of attack to generate the statistics, pattern of attack can be recognized.
2) *System Clustering for Added Security [126]:* For the provided system, DDoS can be nullified or made complicated to achieve by clustering the system. For each cluster created, user authentication can be added. With user authentication requirement, further credibility is necessary to penetrate and cause havoc. Therefore, clustering of system can provide added layer of security to be able to filter out attacks. Moreover, if an attack is carried out on one cluster of the system, rest of the cluster might be safe and not whole system is prone to DDoS attacks.
3) *Detection of High Speed Flow-level Detection System (HiFIND) [127]:* In order to detect the DDoS attack and provide substantial protection to victim and service provider, HiFIND can be utilized. It is highly secured due to its high volume capacity and immune to DDoS attacks for high density data packets. Thus, HiFIND is less prone and highly stable when it comes to DDoS attacks that target weaker system; highly vulnerable to satisfying only low volume traffic.

Lightweight DDoS attack detection [119] has been proposed which is map based detection scheme inspired from Self Organizing Map (SOM) technique. It is a three stage process consisting of flow collector, feature extractor and classifier. The flow extractor is used for gathering the flow statistics from OpenFlow switches. The feature extractor selects the specific information required for the detection based on which classifier determines legitimate user. The security defense approach called Damask has been proposed in [128] to protect SDN from DDoS attacks.

The entropy method has been proposed in [118] to detect DDoS attack on controller using a threshold value. The entropy of the IP addresses is calculated after every incoming 50 packets and if it is above the threshold value, it declares that there is a suspicious activity in the system. These attacks are mostly directed during the communication with switches. Infected switches send overwhelming requests and controller will be involved in responding to these fake requests denying the requests of legitimate users. This attack creates direct harm to the legitimate users. These attacks can be mitigated by protecting controller from these malicious flows.

TopoGuard has been developed in [105] as a security add-on in an OpenFlow controller to address the vulnerabilities in network topology. Network attacks have become most common in many controllers available in today's market such as flood-light, beacon and POX. It focuses on countermeasures for the vulnerabilities involving host tracking services and link discovery in OpenFlow controller. This architecture maintains the record of host profile which includes MAC address, IP address and location information to provide a seamless service without delay in hand-off mechanism. The host profile is monitored and tracked by the Host Tracking Services (HTS) present in the controller. This can be used for determining the valid user. When the controller cannot match the host profile, the new profile is created and stored. If the location of host varies with the profile then it gets updated automatically using HOST_MOVE event presuming the change of location of host. This kind of functionality is not very secure and creates the gateway for the hijackers and spoofing attacks as the users are not validated with any authentication mechanisms. If the attacker can get the access to the location of the target he can trick the controller by mimicking the host creating the Web impersonation attack. Public key based methods can be implemented to validate the host but would not be very efficient solution as the management of these keys would be a tedious task involving cost factor. TopoGuard uses precondition and post condition techniques for validation of the host migration. The precondition is Port_Down signal before host migration and the post condition refers to verification of post location of the host and making sure that it cannot be reached in that location. The Link Discovery Service (LDS) uses Link Layer Discovery Protocol (LLDP) for discovering the internal links between the switches. The link fabrication attacks occur by injecting fake LLDP packets that are capable of creating DoS attacks and man-in-the-middle-attacks. Methods for resolving these kind of attacks include the additional authentication of LLDP packets using Type Length Variable (TLV) and switch port confirmation [105]. Furthermore, LineSwitch, which is a solution based on probability and blacklisting, offers resiliency against SYN flooding-based control plane saturation attacks and protection from buffer saturation vulnerabilities in SDN [121].

FortNOX technique has been introduced in [101] and [113] to resolve the security threats in application layer and the control layer of SDN. It is a software solution implemented in NOX OpenFlow system to secure the system. It responds to

the requests based on the authorization and privileges granted to the users. This method can help to prioritize valid users over fake users.

Avant guard has been proposed in [17] that focuses on two key aspects i.e., security between the data plane and control plane and increasing the response rate of controller to the data plane requests. These two issues of SDN can be resolved by the addition of some more security features into the system namely connection migration and actuating triggers. Connection migration is used for enhancing more security in data plane using classification, report, migration and relay stage. For providing a strict security, flow packets are allowed to interact with the controller only after passing the TCP handshake mechanism. This method can help in the detection of malicious users. The actuating trigger is enabled to increase the responsiveness between the controller and data plane. These triggers make the data plane report all the existing conditions in switches to the controller asynchronously.

In [129], a model has been proposed to analyze the threats that may occur during the communication with data plane by using the OpenFlow protocol. The analysis is done by combination of STRIDE and attack trees to analyze the attacks such as spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privilege.

2) Anomaly Attacks in SDN: Anomaly attacks are involved with many risks such as unauthorized access, malicious application injection etc. which can affect the security of both applications and networks. Furthermore, these attacks are one of the most dangerous attacks that can occur in any layer of the network and are untraceable and hard to detect. Four anomaly detection techniques are proposed in [14] which are discussed below.

1) Threshold Random Walk with Credit-Based (TRW-CB) Algorithm [114] considers the user to be suspicious if the probability value (i.e., ratio of the number of unsuccessful connection and attempts made by the user) is greater than the fixed threshold value.
2) Rate Limiting Algorithm [115] considers the user to be suspicious if the user tries to establish communication with multiple devices in a given time above the threshold value.
3) Maximum Entropy Detection [116] provides the operator with complete view of network from all dimensions. It is two staged process in which it first categorizes the packets into various classes based on destination and then detects the anomalies based of rapidly varying traffic patterns.
4) Network Advertisement (NETAD) [117] is a two staged process. In the first stage, it filters out all unnecessary data such as non IP packets, leaving flows, etc. In the second stage, it monitors the network and detects rarely occurring events and then reports to the controller.

The programmability features of SDN make it convenient to debug the errors as well as attract the attackers [130]. Similarly, No bugs In the Controller Execution (NICE) has been proposed in [112] which is a debugging tool in OpenFlow networks that monitors the condition of the complete system frequently and determines eleven kinds of bugs such as host unreachable after moving, delayed direct path, excess flooding, next TCP packet always dropped after reconfiguration, TCP packet dropped after reconfiguration, ARP packets forgotten during address resolution, duplicate SYN packets during transitions, packets of a new flow are dropped, packets drop when the load reduces etc. NICE approach provides a report of the policy violations and origin of attack helping the system to retrieve back from these bugs.

FRESCO-DB has been developed in [15] from the click router [131] which contains two important modules that are embedded in NOX controller to detect and countermeasure the suspicious threats. The API module creates different schemes to counter attack the malware using IDS and other anti-malware applications. Security Enforcement Kernel (SEK) module is used for the enforcement of the security related applications specified by the controller.

The confidentiality and authenticity of applications in SDN can be protected by the method of encryption and cryptography [11], [132]. Z3 prover method uses a high level programming language to distinguish a legitimate application from malicious applications for protecting confidentiality and integrity of applications [6], [120].

3) Intrusion Attacks in SDN: The traditional networks have built-in middle boxes (which can integrate IDS, firewall and proxy) and other features to block malicious users. These middle boxes are not available in SDN but are essential to secure the SDN from security attacks in both data plane components and controller. They may not be capable of completely preventing the attacks but can be helpful for enhancing the basic security in SDN. However, integration of these modules in SDN comes with some difficulties as SDN has decoupled structure that relies on centralized controller for all tasks such as updating the policies. Thus, incorporating extra modules may impose the overhead on the controller showing its impact on the entire network. This might result in intrusion attacks in SDN when effect of attacks are overlooked as legitimate overhead on the controller [133]. There have been various methods proposed for detecting intrusion attacks in SDN without affecting overall network performance.

FlowGuard has been proposed in [108] as an SDN firewall and is more sophisticated compared to the firewalls in conventional networks. FlowGuard is associated with a dual functionality to work as packet filter and policy checker. It monitors the network for detecting malicious packets and policy violations in SDN.

FlowTag architecture has been proposed in [109] for optimizing the system by adding the extended architecture along with the middle-box which tags packets passing through it. This makes it easier to track the missed and malware packets present in the network among others [109], [134]. Though the middle boxes have many advantages associated with it, the management of the middle boxes in SDN is the tedious task.

NIMBLE architecture has been proposed in [110] for managing the middle boxes based on the policy rules provided by the administrator. The slick architecture has been proposed

in [111] for OpenFlow based SDN which is capable of supporting various devices such as NetFPGAs, GPUs and NPs. This method uses a separate control plane to control all operations of middle boxes which enhances the flexibility of SDN.

As SDN functions are completely based on the instructions provided by the controller/software, it is more vulnerable to process breaks and the introduction of new bugs can result in reduction of overall performance of SDN [135]. According to the recent standard (version 1.3.0) of the openSwitch, the presence of transport layer security (TLS) is not a mandatory option. However, the southbound API of the network, which is more prone to the threats, requires a security layer such as TLS that authenticates users using encryption techniques before granting the access.

4) Spoofing Attacks in SDN: Spoofing attacks are kind of attacks in which attacker uses the identity of a legitimate user to inject fake packets and malicious applications into the network. Because of flexibility that SDN offers, spoofing attacks are easy to implement in the software defined networks. The OpenFlow switches in SDN are data forwarding devices with no intelligence programs. They can be spoofed and used for sending requests to controller. The controller also cannot block these fake packets as it lacks basic components like middle boxes. The other kind of switches in OpenFlow network are soft switches which are responsible for network virtualization. These switches are connected to the controller directly, becoming the attractive target for the attackers. These switches pave a direct way to the attackers into the controller where the attackers can configure the routing policy of the switches misleading all the packets in the network [107]. The effect caused due to these attacks can be reduced by early detection of malware switches. For this purpose all packets flowing into the network should be carefully inspected.

Two schemes for finding suspicious switches have been proposed in [106]. In the first scheme, malicious switches in the system are detected based on the traffic flow higher than a given threshold value. The second scheme in the network involves embedding the third party server to monitor the switches to find any malicious actions.

A technique called Sphinx has been proposed in [107] for spoofing threats/vulnerabilities detection in network topology and communication interface between data plane and control plane. It takes the advantage of the flow graphs to monitor each and every flow. It also uses four important OpenFlow commands such as FLOW_MOD, PACKET_IN, STATS_REPLY and FEATURES_REPLY to obtain all required data from the switch and alert the controller if any suspicious activity is detected near the switches [107].

All of these attacks can be avoided to some extent by maintaining the privacy of users' contents. Most dangerous attacks in the networks are targeted to IP-based networks. This necessitates the need for protecting the IP as a whole. In SDN the special mechanism called OpenFlow random host mutation (OF-RHM) is implemented which hides the actual IPs and uses the random virtual IP addresses [104] that prevents the attacks to some extend.

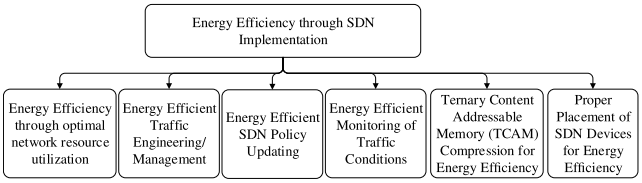
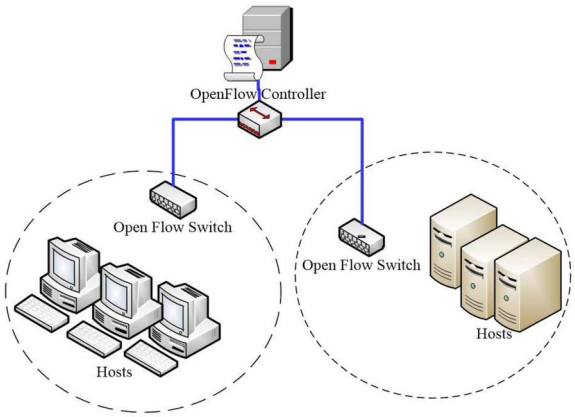


Fig. 7. Different approaches for energy saving through SDN implementation.



(a) Typical OpenFlow based Network

(b) NEC PF5240 OpenFlow 48 port Switch

Fig. 8. Typical OpenFlow based SDN and OpenFlow switch.

IV. Energy Efficiency and Its Tradeoff With Security in Software Defined Networks

Due to ease to modify the network operations and adapt various energy efficient mechanisms, SDN provides a better solution not only for network security but also for green networking which has become important in network design and deployment for economic and environmental benefits. Fig. 7 consists of different approaches for energy saving in SDN. It is worth noting that the security schemes when implemented consume more energy than without them as security schemes consists of computing and communications which consume more energy in the network. Thus we study both security and energy efficiency and their trade-off in this section.

To see the trade-off between energy efficiency and network security, we analyze and evaluate different security schemes when SDN and energy saving techniques are implemented. We created a small scale testbed for SDN as shown in Fig. 8 where OpenFlow controller monitors the overall status of the network. The OpenFlow PF5240- ProgrammableFlow Switch offers forty-eight 10/100/1000 Mbps ports & four 1000/10000 Mbps ports with 176 Gbps switching speeds (Fig. 8(b)); with Flow entry capabilities of 64000-160000. This hybrid controller connects OpenFlow networks to L2/ L3 networks. Having access to this switch will reduce

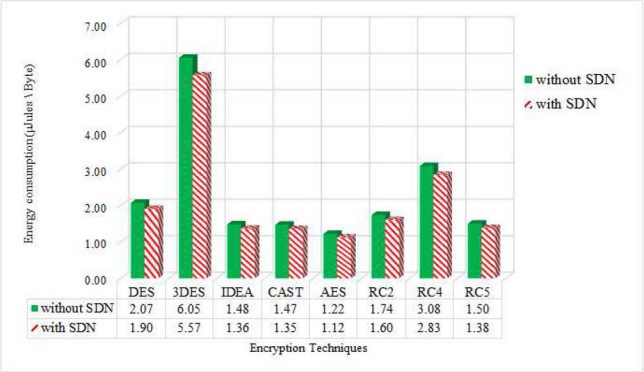


Fig. 9. Comparison of energy consumption with and without SDN implementation for **encryption and decryption** in various symmetric ciphers and transmission.

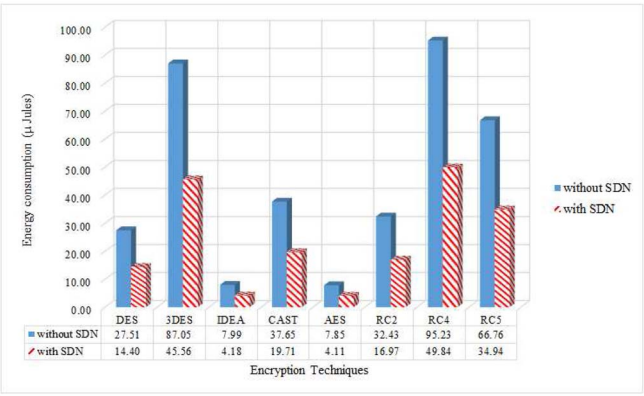


Fig. 10. Comparison of energy consumption with and without SDN implementation for **key setup** in various symmetric ciphers.

network complexity while eliminating the need for traditional network protocols by providing OpenFlow 1.0 and 1.3.1 support. NEC's Virtual Tenant Networks (VTN) technology provides secure multitenant cloud networks that will allow access to various pre-existing devices from other third-parties. Furthermore, we created a traditional network that was equivalent to SDN but networking functions were fixed unlike SDN.

Then, we have implemented different security schemes and energy saving schemes, and collected data to evaluate and compare the performance of SDN and traditional network as well as to study tradeoff between energy efficiency and security.

First, in Fig. 9, we plotted comparison of energy consumption with and without SDN implementation (adaptive configuration for both security and energy efficiency) for encryption and decryption of same file in various symmetric ciphers and transmuted that file from one computer to another. From Fig. 9, we observed that the energy consumption for different symmetric ciphers is much higher in traditional network than that in SDN. SDN consumes less energy (while providing same network security) because of the configurable features of SDN that allows network to adjust both speed of the port (as 10 Mbps port consumes less energy than 100 Mbps or 1000 Mbps ports) and utilization of the link based on the need which is not available in traditional network.

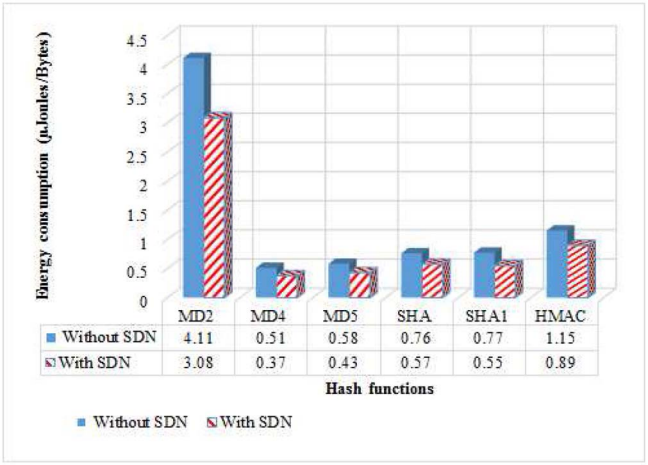


Fig. 11. Comparison of energy consumption with and without SDN implementation for various Hash functions and transmission.

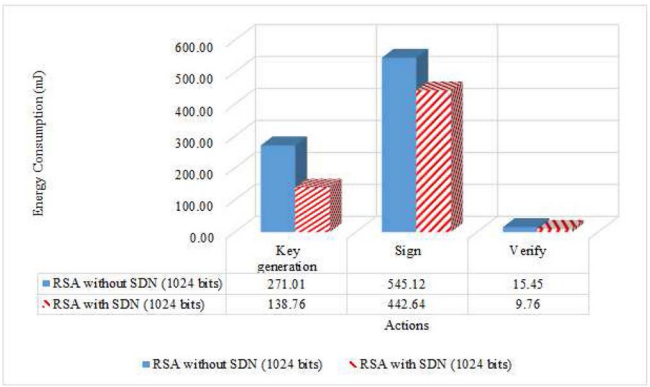


Fig. 12. Comparison of energy consumption with and without SDN implementation with 1024 bits key size for DSA based digital signatures and transmission.

Next, we plotted variation of energy consumption for traditional network and SDN for key setup in various symmetric ciphers as shown in Fig. 10. When key was exchanged, SDN used low speed (e.g., 10 Mbps) link without degrading any network performance as key size is small. But in traditional network, small size key was exchanged with high speed link which consumes more energy. As a result, traditional network consumes more energy compared to SDN as shown in Fig. 10.

In Fig. 11, we plotted the variation of energy consumption with and without SDN implementation for various Hash functions for transmission of same data file. Because of the adaptive nature of SDN, it adjusts the network parameters (speed of the port/link, sleep/on mode based on link activity, etc.) and consumes less energy than that in traditional network as shown in Fig. 11.

Next, we plotted the variation of energy consumption with and without SDN implementation of digital signatures for Digital Signature Algorithm (DSA) in Fig. 12 and for RSA in Fig. 13. In both DSA and RSA, energy consumption is higher in traditional network than that in SDN.

Next, we implemented RC5 security scheme in traditional network and SDN for identical experimental setup. We plotted the variation of energy consumption and security for their

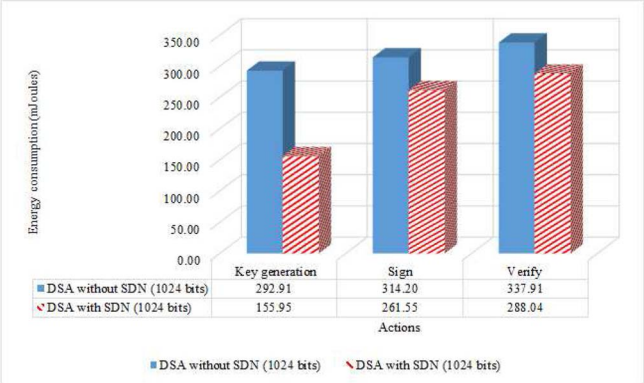


Fig. 13. Comparison of energy consumption with and without SDN implementation with 1024 bits key size for RSA based digital signatures and transmission.

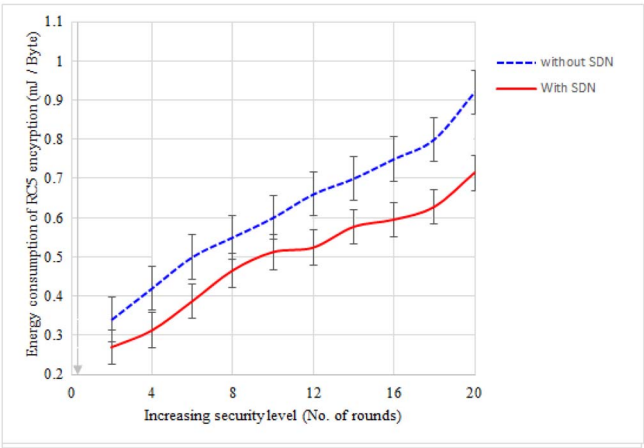


Fig. 14. Comparison of energy consumption and security trade-off for RC5 encryption and transmission.

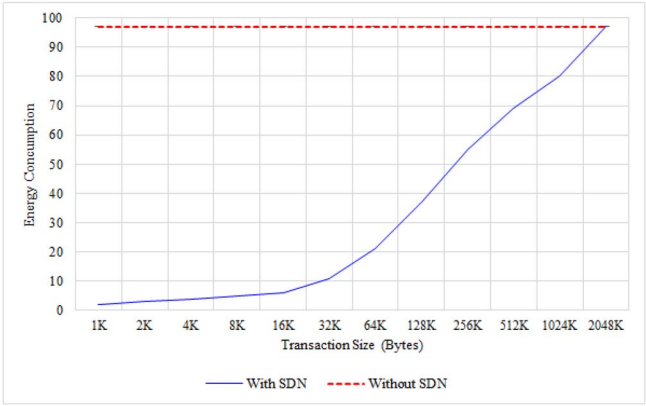


Fig. 15. Comparison of energy consumption for SSL handshake with and without SDN implementation with increasing transaction sizes.

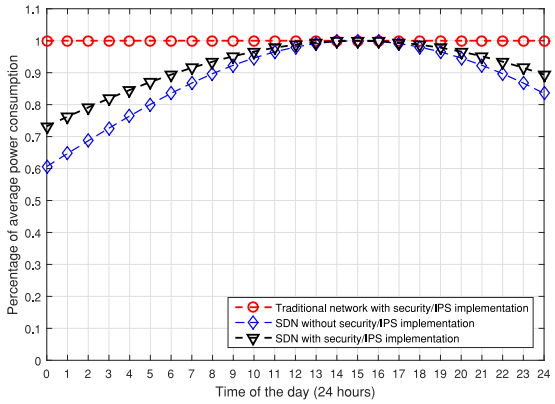


Fig. 16. Variation of percentage of average energy/power consumption by traditional network and SDN with and without security/IPS implementation for a typical small office home office environment as shown in Fig. 8. Experiment was conducted in CWiNs research lab at GSU with NEC OpenFlow switches and controller by mimicking the traffic patterns of the University computer network.

trade-off for RC5 encryption and transmission of the information in Fig. 14. As mentioned earlier, SDN adapts the link parameter and operating parameters, it consumes less power than that in traditional network as shown in Fig. 14. However, when SDN needs higher security (higher rounds for RC5 encryption for better security), energy consumption increases with the level of security (e.g., number of encryption rounds) as shown in Fig. 14.

Then we plotted variation of energy consumption for secure socket layer (SSL) handshake in traditional network and in SDN with increasing data sizes as shown in Fig. 15. We observed that energy consumption increases with the data size as shown in Fig. 15 for both traditional network and SDN. However, energy consumption is lower for SDN than that for traditional network since SDN adapts the link speed based on the information to be transmitted. When SDN uses its maximum speed limit and its link is fully utilized, it consumes same energy as traditional network as shown in Fig. 15. However, when there is a room to utilize lower link speed, it consumes lower energy without degrading the network performance.

Finally, we plotted the variation of average energy/power consumption for the traditional network and the SDN (with and without security/IPS implementation) as shown in Fig. 16. In traditional network, regardless of time, it consumes same power however SDN consumes variable power based on load which is based on the time of the day as shown in Fig. 16. During the peak hours all network consume maximum power however SDN consumes less power during off peak hours. When security scheme such as IPS is implemented, SDN consumes more power than that of without security implementation as shown in Fig. 16. Thus we can sumarize that the security comes with cost and we need to consider trade-off between energy efficiency and security.

We can conclude that the SDN can adapte its network parameters on the fly to provide security with lower energy consumption compared to traditional network. In other words, energy efficient approaches for SDN can help to consume less power while providing better or same level of performance and security than that in (equivalent) traditional network. Thus it is important to study energy efficient approaches for the SDN, which are discussed in the following sub-section to make the paper self-content.

A. Energy Efficiency in SDN

As discussed in earlier sections, one of the major problems faced by many companies around the world is the

amount of energy consumed by networks. The need for continuous availability and its huge architecture make network energy consumption high. The companies pay a significant percentage of their revenue to power their network infrastructures [136]. Usage of the energy efficient networks such as SDN is regarded as a solution for reducing the overall consumption of power in the network [23]. This attracted companies like Google to incorporate SDN in their networks [64]. Implementation of some energy saving strategies in SDN can reduce the overall power consumption which leads to the cost reduction [89]. The openness, feasibility and programmability of SDN reduces the complexity to implement energy efficiency approaches in both hardware and software. It would be more efficient to apply power saving schemes in each module for overall energy saving [137]. For this purpose the power consumed in SDN by each module such as chassis, routers and nodes in the network needs to be known. Furthermore, SDN is regarded as a viable solution where minimal resources could be used to perform a task without degrading overall network performance (such as security) that reduces energy consumption.

Measurement model has been presented in [138], where the power consumed by OpenFlow switches such as OF hardware Switch and OF vSwitch running on the server are considered for the experiment. According to the results obtained, enabling sleeping mechanisms can improvise the energy efficiency of OF vSwitch as the power consumption of network is dependent on number of active links in the network. The additional savings up to 6.6% of overall power can be achieved by setting port configuration rate to 10 Mbps. The obtained power measurements are expected to have error less than 1% in hardware switches and 8% in software. The frequent powering off and on can show its impact on decreasing the life time of networking devices. The efficiency can only be achieved when the adopted schemes do not affect the performance of the system. So, the method to be implemented in network should be selected based on characteristics of the network.

Table III and Table IV provide a comparison between various energy efficiency techniques that have been proposed to implement in SDN. These tables provide working principle of each method along with advantages and disadvantages that they offer when they are implemented in SDN.

The energy efficient schemes that can be implemented in SDN are discussed below.

1) Energy Efficiency Through Optimal Network Resource Utilization: The amount of traffic during the different times of day is not similar, especially during night times the traffic load may be reduced to great extent and most of the nodes in network remain unutilized or underutilized (as shown in Fig. 16). However, when security defense techniques are implemented, power consumption increase significantly. In SDN, due to the flexibility to control the networking devices using a high level programming language, the rerouting techniques can be implemented with ease [88]. The controller in SDN can make decisions according to the traffic load in system promoting green networks and efficient resource utilization. The nodes that have no traffic can be sent in sleep mode and the nodes with low traffic can be rerouted to few active networks to provide the service. This leads to the energy efficient proper utilization of the networks.

Content based method [148] has been implemented in software defined Information Centric Network (ICN) for proper utilization of resources with reduced power consumption in the network. ICN has prior information about the length of content that needs to be delivered and allocates only required amount of resources. This method also keeps track of these resources to ensure proper utilization. Elastic tree model proposed in [89] showed 50% reduction in power usage where optimizer module allocates most suitable link to efficiently handle the traffic load while meeting QoS requirements. The unused links in network are put in sleep mode to save the energy.

Multi Layer Traffic Engineering (MLTE) [149] and GreCO [150] follow the similar approach as elastic tree and these approaches insignificantly save power consumption in SDN [88]. The exclusive routing algorithm (EXR) in [173] routes the traffic based on the time dimension and this method of routing is more effective and quick compared to other energy saving algorithms. In [174], energy efficient routing protocols are proposed to route the network traffic to the most suitable and shortest path to meet the requirements of the users. The queue engineering process has been adopted for energy saving of OpenFlow switch in NETFPGA platform [175]. The clock controller is combined with OpenFlow controller for supporting various modes for power management. This method has a separate module that lowers the frequency to 0 MHz in no traffic conditions for proper usage of power [176]. Reducing the replication of unwanted data can reduce the power consumption of network to some extent [22]. To avoid redundancy in storage of data SMart In-Network Duplication (SMIND) method has been implemented in [177]. This method identifies the redundant data using fingerprinting technique.

2) Energy Efficient Traffic Engineering/Management: Traffic engineering/management for energy optimization is not a new concept. This approach is popular in traditional networks and is used in SDNs for energy efficiency. The Asynchronous Transfer Mode (ATM) network [178] has strict and limited policies for protecting the entire network without degrading Quality of Service (QoS) of the users. These features are beneficial to SDN. Flow management and load balancing techniques can be implemented in both switches and controller for energy efficiency in SDN [65], [179]. Energy usage is minimized in Internet Protocol (IP) based traditional networks using load balancing and efficient routing paths such as shortest path routing protocols [180]. These features are also regarded as energy saving schemes in SDN. Similarly, MultiProtocol Label Switching protocol (MPLS) is mostly focused on implementation of traffic engineering (TE) schemes in the Internet infrastructure [181] for efficient delivery of packets with optimal energy. This method is suitable for traffic engineering in SDN where drawbacks of MPLS can be addressed by the OpenFlow networks.

Hash based ECMP has been proposed in [139] as an Equal Cost Multi Path switch based load balancing scheme that directs the flow to multiple paths in the network to

TABLE III
ENERGY EFFICIENCY TECHNIQUES IN SDN

Category	Method	Principle	Specialties(+) and Limitations(−)
Traffic Engineering	1. Hash Based ECMP [139]	Switch load balancing by distributing flow in multiple paths.	+ Efficient link utilization − Prone to collisions
	2. Hedera [140]	Detects the elephants flows based on threshold value.	+ Capable of handling elephant flows − Prone to overheads
	3. Mahout [141]	Detects the elephant flows using additional layer called shim layer.	+ Capable of handling elephant flows + Overcomes the delay − Prone to overheads
	4. Devo Flow [142]	Assigns set of wildcard rules to switches to reduce the overhead on controller.	+ Reduces the overhead on controller − Reduced network visibility
	5. DIFANE [143]	Introduces authority switches in network to handle basic flows.	+ Reduces the overhead on controller − Hard to manage
	6. Hyperflow [144]	Distribution of controllers around the plane and directing requests to nearest controller.	+ Scalability + Fault tolerance − Complex architecture
	7. Balanceflow [145]	Distributes the flows evenly among all controllers based on reported traffic information.	+ Energy efficient − Complex architecture
	8. Kandoo [146]	Distributes the controllers in two layers to handle traffic.	+ Reduces overhead on controller − Not scalable
	9. SOTE [147]	Combines OSPF and SDN for proper link utilization.	+ Energy efficient − capable of achieving only near optimal performance
Proper utilization of resources	1. Content Based Traffic Engineering [148]	The resources are allocated based on the length of content.	+ Efficient link utilization + Scalability − Cannot be supported by the standard OpenFlow
	2. MLTE [149] , Elastic Tree [148], GreCO [150]	Rerouting the traffic to active links and implementing sleep awake mechanisms.	+ Energy efficiency + Resource utilization − Delay − Not fault tolerant
Monitoring traffic	1. OpenTM [151]	Relies on OpenFlow switch to obtain traffic statistics.	+ Quick estimation − Accuracy depends on selected switch
	2. Payless [152]	Relies on OpenFlow switch to obtain traffic statistics.	+ Cost effective − Prone to overheads
	3. Flowsense [153]	Calculates link utilization based on control messages.	+ Highly accurate − Cannot capture instant utilization
	4. Opensample [154]	Relies on sFlow packet sampling and TCP sequence.	+ Highly accurate +low latency error + Detects elephant flows
	4. Opensketch [155]	Decoupled three stage process used for monitoring traffic.	+ Highly accurate − Delay
	5. MicroTE [156]	Functions as both traffic engineering and monitoring scheme. relies on monitoring agent in server.	+ Dynamic management − Prone to overheads
Updating the policies	1. Net Plumber [157]	Updates only required portion of switches	+Fast Update − None
	2. Safe protocol update [158]	Only stores a single version of rules	+ Removal of redundant rules +Reduced power consumption +Memory and Bandwidth saving − High Processing Time
	3. TIMECONF [159]	Updates based on scheduled time	+ Consistency − Delay
	4. Duplication [48]	Stores both New policies and old policies for a certain time	+ Consistency − More memory consumption
TCAM compression	1. TCAM razor [160]	Reduction of Flowentries	+ Removal of redundant rules +Reduced power consumption − Limited Flowentries
	2. Compact TCAM [161]	Saving the Space in TCAM by using smaller flow id	+ Removal of redundant rules +Reduced power consumption − Limited Flowentries
	3. Bitweaving [162]	Saving the space by merging similar flow entries	+ Removal of redundant rules +Reduced power consumption +High speed − Limited Flowentries
Rule placement	1. Palette [163]	Decomposes flow tables and store the rule under different networks	+Allows more Flowentries − Delay
	2. Joint Optimization [164]	Combines TE with proper rule placement to achieve energy efficiency.	+Fast Update − Delay
	3. ILP [165]	Implements geedy heuristic method to achieve energy efficiency.	+Fast Update − Delay
	4. Cacheflow [166], [167]	Combines hardware and software switches to achieve energy efficiency.	+ Cost effective − Delay

enhance energy efficiency. The major drawbacks of this load balancing technique are computational complexity and low performance. Hedera has been regarded as an intelligent load balancing scheme capable of handling the large flows [140]. In Hedera, controller manages the traffic based on the information obtained from switches and consumes minimum energy.

TABLE IV
ENERGY EFFICIENCY TECHNIQUES IN SDN (CONTD...)

Category	Method	Principle	Specialties(+) and Limitations(−)
Energy saving in datacenters	1. ECDC [168]	Relies on VM to power off the unused servers	+ Easy manageability + Energy Saving −Complex to integrate
	2. Increment Approach [169]	Allocates a space in each switch for handling the rerouted traffic	+ Link utilization + Cost effective − Scalability
	3. MCTEQ [170]	Flows are handled based on the priority set by the operator.	+ High Performance + High data rate − Scalability
Placement	1. Optimal Controller Placement model [171]	Placement of controller to achieve cost reduction meeting some specific criteria.	+ Cost effective − Reduced Performance
	2. EQVMP [172]	Three staged process for proper placement of VM machines for achieving energy efficiency.	+ Energy Efficiency + Resource utilization + Quality of service − Delay −performance

This helps to avoid collisions. Mahout [141], a load balancing scheme, has been implemented in data centers to enhance network performance and reduce energy consumption. DevoFlow load balancing has been implemented in [142] in enterprise networks and data center environments to reduce the burden on controller by providing the switches with a set of additional wild card rules and minimize the overall energy consumption. Other advantages of this method includes performance and scalability. An approach called DIFANE has been capable of achieving controller load balancing by implementing detailed and strict policies in enterprise networks [143]. The goal of the DIFANE is similar to DevoFlow, however this scheme adds additional switches in the network called authority switches which store all the important flow entries. When the packet does not match with the flow table rules in the regular switches they are immediately forwarded to authority switches for making the decision. Hyperflow has been proposed in [144] as an event based distributed control plane platform, which can provide the benefits offered by the OpenFlow and also overcome its limitation of scalability. Balance Flow has been proposed in [145] as a controller load balancing scheme in OpenFlow networks. It has been proposed as an extension in OpenFlow switches called Controller X action. This classifies the flows into various categories based on the switches from which they originated and directs them to different controller. SDN/OSPF Traffic Engineering (SOTE) has been proposed in [147] as a hybrid traffic engineering method with the combination open shortest path first and SDN to lower the link utilization in the network. The main goal of this method is load balancing by directing them evenly through all the SDN nodes and minimize the overall energy consumption.

3) Energy Efficient SDN Policy Updating Including Security Policies: Conventional networks update the network policies once in a while, whereas SDN being an adaptive architecture needs to be updated frequently to adapt itself to updated environment. The constant updates in the system can hinder the performance of network and also increase power consumption. In SDN, the controller is responsible for updating and enforcing the new policies in the network. Whenever a new policy is updated in a network, all switches present in the network receive and store it along with the version number. When multiple policies exist in the switches then the packet is differentiated based on the version number and processed by using new policy or old policy.

A method with per packet consistency and all packet consistency update in SDN has been presented in [48]. The SDN controller updates the system in a timely manner and eliminates the old policies when it stops receiving the packets using the old versions. Some other controllers set expiration date for the older versions and do not support them after this fixed date. The major limitation of this method is it should store both versions in the flow table for certain time. This may overwhelm the flow table entries and consume more space in memory and increase the energy consumption. A new method that deals with single set of rules has been proposed in [158] to address the issues of memory consumption and energy consumption. In TIMECONF method proposed in [159], new policies are updated sequentially in a scheduled time. However this method is associated with some delay as controller updates the next switch only after it receives an acknowledgment from the updated switch. Incremental update method called Net-Plumber [157] has been proposed to enable quick policy updates in network by configuring only the portion of switches that needs an update resulting in lower energy consumption. It is located between data plane and control plane, and enforces policies into switches at the rate of $50 - 500\mu$s [157].

Note that the energy can be saved in SDN by lower the speed of the link while updating the SDN policies. Note that the SDN policies can be easily updated with 10 Mbps links in real-time and consume 4 watts lower than that of 1Gbps link.

4) Energy Efficient Monitoring of Traffic Conditions: Energy saving in the network can be obtained by dynamically programming the network according to traffic conditions. In order to dynamically adjust the system based on traffic flow, it is required to have updated information about the traffic conditions in the network. This necessitates need for traffic monitoring approaches to be implemented in SDN. Though these monitoring schemes are not completely accurate, they can help the controller to have an idea about traffic flow in the network including network security attacks.

An approach called Open TM has been proposed in [151] as a query based monitoring system that relies on the features of OpenFlow switches to measure the network traffic. The information cannot be obtained from all switches as it may impose overhead on the network and consume more energy. The switch from which the traffic statistics are to be collected are intelligently chosen from the routing information present in the controller to minimize the energy consumption and boost the overall network performance. Payless has been presented in [152] as a pull based traffic monitoring scheme that uses adaptive statistics collection algorithm for obtaining highly error free traffic conditions in the entire network. In this method, controller queries the switches in data plane continuously for the updates regarding the traffic flow. Payless proved to be the effective method by lowering the energy consumption in the network. The drawback of this system the querying the controller continuously for maintaining accuracy which may impose overhead on the controller.

FlowSense has been proposed in [153] as a push based monitoring method that focuses on estimating the link utilization. The switches forward the message regarding the detection of new flows to the controller using PacketIn and FlowRemoved commands based on which the controller enforces new policies into the system. These messages could favor FlowSense monitoring scheme to estimate the resource utilization, bandwidth and energy consumption in the network. OpenSketch has been proposed in [155] as a push based traffic flow monitoring scheme which follows the similar concept of decoupling the planes as SDN and uses three major stages namely hashing, filtering and counting. Hashing is used for providing a brief overview on the flows that needs to be measured. The filtering stage eliminates the unnecessary data and statistics are obtained in the counting stage. The results obtained are highly accurate. MicroTE has been proposed in [156] as a traffic monitoring scheme in the network. It dynamically adjust itself to the traffic conditions in the network and immediately respond to the changes in the network. The updates on the recent traffic conditions received from the agent installed in the server and immediately reported to the controller. OpenSample has been proposed in [154] as a push based monitoring scheme that relies on sFlow packet sampling tool for obtaining the packet headers from network.

5) Ternary Content Addressable Memory (TCAM) Compression for Energy Efficiency and Network Security: The rules that are to be implemented in the SDN are stored in flow table present in Ternary content addressable memory (TCAM). It can compare all incoming flows parallelly and enable quick packet processing [182]. The number of the entries in the flow table are limited as the TCAM usage is associated with cost factor. TCAM is expected to impose the burden of 400 times more cost and 100 times more power usage [183] than the traditional memory storage devices such as RAM. The other main concerning issue of TCAM is its update time that is limited to 40−50 rule-tables per second [166], [167], [184].

Compact TCAM has been proposed in [161] that condenses the structure of TCAM by lowering the size of the flow IDs in a flow table. The flows assigned with a specific flow ID for the purpose of identification. The other advantage of the compact TCAM includes the usage of SDN for removal of redundant information. By the implementation of compact TCAM method 80% of the power reduction can be achieved. TCAM razor has been proposed in [160] to reduce number of flow entries in the flow table by implementing four step mechanism that compresses the TCAM by 29.0% which leads to the savings of 54% energy. In this method multidimensional rules are fragmented into many one dimensional rule lists. BitWeaving has been proposed in [162] as a non-prefix ternary classifier implemented for the compression of rules in TCAM using two different approaches: bit swapping and bit merging. In this method flow entries with same decision and just differ by the single bit can be combined together. BitWeaving method was able to achieve the compression ratio of 23.6% with high speed and energy efficiency [162].

Palette distribution [163] has been proposed to offer a solution for rule placement problem by breaking the large SDN tables into small sub-tables using pivot bit decomposition. As all rules cannot be stored under the single network they are divided and distributed among the multiple networks under SDN. Joint optimization method has been proposed in [164] that uses both rule allocation and traffic engineering to achieve energy optimization and security in the network. Integer Linear Program has been proposed in [165] that uses greedy heuristic method for achieving the energy saving. Though the software switches can eliminate the issues such as high cost and can also update the flow tables up to 10 times quicker than the hardware switches, they are associated with some delay in the packet processing. CacheFlow has the hybrid switch embedded with both features of hardware and software switches [166], [167] that offers energy efficient solutions.

6) Proper Placement of SDN Devices for Energy Efficiency and Network Security: As SDN controller is regarded as a brain of the SDN to control behaviors of the network, its placement location has a crucial role. The controller should be capable of managing the provided number of switches in the network. The proper placement of the controller can improve the overall efficiency in the SDN and can also serve as a benefactor for reducing the cost. Deployment of many controllers in SDN has both pros and cons. The significant amount of research has been conducted on SDN controllers that are needed to be integrated in the system. The number of controllers in the SDN should be based on the amount of traffic load and network security requirement, and the size of the network that a given controller needs to control A mathematical model has been proposed in [171] for the placement of controllers with a motive to minimize the energy cost and to boost the network performance. The factors such as the location of the switches, length and bandwidth of the switches and other information is taken into consideration. The controller placement problem for the minimization of cost can be expressed as

$$\text{Minimize} \quad C_c(x) + C_l(v) + C_t(z), \tag{4}$$

where $C_c(x), C_l(v), C_t(z)$ are the cost of installing switches, cost of linking controllers to switches and the cost for linking the controllers together respectively. The equation (4) should

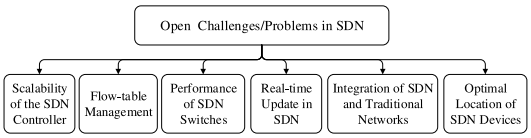


Fig. 17. Typical open challenges and problems in SDN.

satisfy some criteria such as the number of switches connected to the network is always less than the number of ports in the network, the switches in the network have only one particular link to connect to the network, the controller is capable of handling number of packets sent by the switches connected to a given controller.

Next, VM placement in SDN enables proper resource utilization by the infrastructure in the network. Though the VMs are used in the network for the improvement of energy saving. The improper and excessive placement of VMs may degrade the overall performance and network security, and increase the energy consumption. Energy efficient QoS guarantee Virtual Machine Placement (EQVMP) method has been proposed in [172] to find optimal ways for VM placement. This method is implemented in three phases as

a). *Hop Reduction:* Decomposes the VM's into multiple classes and uses protocols such as OSPF to determine the suitable path.

b). *Energy Saving:* The suitable location for placing VM's is decided using best fit algorithm. The position of VM should reduce resource consumption without affecting QoS.

c). *Load Balancing:* Evenly distributing the traffic for achieving energy efficiency without affecting the network performance and security.

Note that suitable placement of SDN devices in a network can help to save significant amount of energy consumed by the network and provide efficient security defense.

V. Other Challenges in SDN Related to Performance, Security and Energy Efficiency

SDN is regarded as an emerging technology for network security and energy efficiency. However, there are many challenges and open problems to realize its full potential. In this section, we discuss about important problems and challenges in SDN. Typical open challenges/problems are summarized in Fig. 17.

A. Scalability of the SDN Controller

In SDN, network functions are centralized to SDN controller. The SDN controller is responsible for handling all important operations in the network. Thus finding optimal number of controllers and their optimal position/placement in the network for better performance, scalability, security and energy efficiency is still an active research topic.

B. Flow-table Management in SDN

SDN is flat network where SDN switches lack efficiency to work independently which makes them dependent on the rules set by the SDN controller. SDN switch matches the packet with the flow entries in the table to make a decision. The flow tables in SDN have limited space. The storage of more number of flow entries could impose overhead on the flow tables thereby increasing cost and degrading the overall performance. This necessitates the need for intelligent flow table management methods that can store more number of rules without increasing the cost and degrading the overall network performance, network security and energy consumption in SDN [6], [185].

C. Performance of SDN Switches

SDN switches are not intelligent and operate based on the rules set by the SDN controller. The performance of switches in a SDN directly impacts the overall performance of the network including network security [7]. The concerning issues in the switches are: maximum output that can be obtained using available OpenFlow switches is limited to $38-1000$ flow-mod per second. Switches need high capacity processors to work efficiently. If more number of CPUs are included in the switch, power consumption will be increased.

D. Real-time Change Update in SDN

SDN is reconfigurable on the fly based on its operating conditions. In any SDN, it has been a real challenge to address *dynamic real-time change* and deployment of rules. As discussed, SDN has an ability to automate the provisioning of new converged infrastructures in minutes and impact multiple devices in real time. This results in gaps in visibility of SDN changes and there will be a huge impact when small things go wrong. In SDN, another challenge is accommodating rapid on-demand growth which poses a risk to SDN monitoring platforms. Defense and monitoring solutions must be able to accommodate the rapid growth of the SDN infrastructures. Otherwise they can quickly become over-subscribed resulting in failure of whole systems.

E. Integration of SDN and Traditional Networks

Traditional network is hierarchical and SDN is flat. Thus, integration of SDN and legacy network is another challenge where defense solutions and monitoring systems should be compatible and robust for both networks to enhance the overall performance, network security and energy saving. Note that one system should not be the bottleneck for other. SDN is expected to be able to work based on a context of a particular customer or tenant of the network to serve them better and meet their QoS requirements. This may degrade the network performance when there is hybrid network with physical and virtual services.

F. Optimal Location of SDN Devices

Optimization for placement of SDN controller, SDN switches and other end devices is another open problem in SDN as placement of different SDN devices impacts the overall network performance and security. It would be more important to enable more efficient network resource sharing and improve services for trillions devices in Internet of Things applications [186], [187].

Note that the selection of different parameters depends on the application that the SDN is envisioned to support. For instance, optimal parameters in SDN for smart grid may not be optimal SDN parameters for traditional data center networks [188], optimal SDN for IoT may not be optimal SDN for cyber-physical systems, and so on.

VI. Conclusion

In this survey, we have explored Software Defined Network (SDN) architecture, various security threats that are resolved by SDN and new threats that arose as a result of SDN implementation as well as energy efficiency. We have summarized the recent security attacks and countermeasures in SDN in a tabular form for side-by-side comparison. We have also provided a survey on different strategies that are implemented to achieve energy efficiency in the networks through SDN implementation and presented in a tabular form. In an effort to anticipate the future evolution of this new paradigm, we discuss the challenges and research efforts in SDN. Note that the selection of different SDN parameters depends on the applications that the SDN is envisioned to support. For instance, optimal SDN parameters for smart grid networking may not be optimal SDN parameters for traditional data center networks and optimal SDN for IoT may not be optimal SDN for cyber-physical systems and so on.

It is noted that the future work should be focused on designing low power consuming security mechanisms that can enhance the overall network performance with high visibility and scalability.

Acknowledgment

However, any opinion, finding, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of NSF. The authors are grateful to the anonymous reviewers for their constructive comments on the paper.

References

[1] S. Jeschke, C. Brecher, H. Song, and D. B. Rawat, *Industrial Internet of Things: Cyber-Manufacturing Systems*. Cham, Switzerland: Springer, 2016.

[2] D. B. Rawat, J. J. Rodrigues, and I. Stojmenović, *Cyber-Physical Systems: From Theory to Practice*. Boca Raton, FL, USA: CRC Press, 2015.

[3] P. Goransson and C. Black, *Software Defined Networks: A Comprehensive Approach*. St. Louis, MO, USA: Elsevier, 2014.

[4] N. McKeown, "Software-defined networking," *INFOCOM Keynote Talk*, vol. 17, no. 2, pp. 30–32, 2009.

[5] T. D. Nadeau and K. Gray, *SDN: Software Defined Networks*. Sebastopol, CA, USA: O'Reilly Media, 2013.

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

[7] D. Kreutz *et al.*, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[8] *Open Networking Foundation*. Accessed on May 8, 2016. [Online]. Available: https://www.opennetworking.org

[9] *Floodlight*. Accessed on May 8, 2016. [Online]. Available: http://www.projectfloodlight.org/ floodlight

[10] *OpenDayLight*. Accessed on May 8, 2016. [Online]. Available: https://www.opendaylight.org

[11] N. McKeown *et al.* (2011). *OpenFlow Switch Specification*. Accessed on Aug. 1, 2016. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf

[12] T. Xing, Z. Xiong, D. Huang, and D. Medhi, "SDNIPS: Enabling software-defined networking based intrusion prevention system in clouds," in *Proc. 10th Int. Conf. Netw. Service Manag. (CNSM)*, Rio de Janeiro, Brazil, 2014, pp. 308–311.

[13] O. Joldzic, Z. Djuric, and D. Vukovic, "Building a transparent intrusion detection and prevention system on SDN," *Norsk informasjonssikkerhetskonferanse*, vol. 7, no. 1, pp. 1–4, 2014.

[14] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Recent Advances in Intrusion Detection*. Heidelberg, Germany: Springer, 2011, pp. 161–180.

[15] S. Shin *et al.*, "FRESCO: Modular composable security services for software-defined networks," in *Proc. NDSS*, San Diego, CA, USA, 2013, pp. 1–16.

[16] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN security: A survey," in *Proc. IEEE SDN Future Netw. Services (SDN4FNS)*, Trento, Italy, 2013, pp. 1–7.

[17] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, Berlin, Germany, 2013, pp. 413–424.

[18] D. B. Rawat and C. Bajracharya, *Vehicular Cyber Physical Systems: Adaptive Connectivity and Security*. Cham, Switzerland: Springer, 2016.

[19] D. B. Rawat and C. Bajracharya, "Software defined networking for reducing energy consumption and carbon emission," in *Proc. IEEE SoutheastCon*, Norfolk, VA, USA, 2016, pp. 1–2.

[20] E. Gelenbe and Y. Caseau, "The impact of information technology on energy consumption and carbon emissions," *Ubiquity*, vol. 2015, p. 1, Jun. 2015.

[21] D. B. Rawat and S. Reddy, "Recent advances on software defined wireless networking," in *Proc. IEEE SoutheastCon*, Norfolk, VA, USA, 2016, pp. 1–8.

[22] R. Wang *et al.*, "Energy-aware routing algorithms in software-defined networks," in *Proc. IEEE 15th Int. Symp. A World Wireless Mobile Multimedia Netw. (WoWMoM)*, Sydney, NSW, Australia, 2014, pp. 1–6.

[23] B. Yan, J. Zhou, J. Wu, and Y. Zhao, "Poster: SDN based energy management system for optical access network," in *Proc. 9th Int. Conf. Commun. Netw. China (CHINACOM)*, Maoming, China, 2014, pp. 658–659.

[24] B. B. Bista, A. Fukushi, T. Takata, and D. B. Rawat, "Reducing energy consumption in wired OpenFlow-based networks," *Int. J. Control Autom.*, vol. 7, no. 6, pp. 401–412, 2014.

[25] B. B. Bista, M. Takanohashi, T. Takata, and D. B. Rawat, "A power saving scheme for open flow network," *J. Clean Energy Technol.*, vol. 1, no. 4, pp. 276–280, 2013.

[26] K. Dhamecha and B. Trivedi, "SDN issues—A survey," *Int. J. Comput. Appl.*, vol. 73, no. 18, pp. 30–35, 2013.

[27] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 623–654, 1st Quart., 2016.

[28] I. Alsmadi and D. Xu, "Security of software defined networks: A survey," *Comput. Security*, vol. 53, pp. 79–108, Sep. 2015.

[29] W. Li, W. Meng, and L. F. Kwok, "A survey on OpenFlow-based software defined networks: Security challenges and countermeasures," *J. Netw. Comput. Appl.*, vol. 68, pp. 126–139, Jun. 2016.

[30] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van Der Merwe, "The case for separating routing from routers," in *Proc. ACM SIGCOMM Workshop Future Directions Netw. Architect.*, Portland, OR, USA, 2004, pp. 5–12.

[31] A. Greenberg *et al.*, "A clean slate 4D approach to network control and management," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 41–54, 2005.

[32] M. Casado *et al.*, "Ethane: Taking control of the enterprise," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 1–12, 2007.

[33] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, 2014.

[34] S. Bhattacharjee, K. L. Calvert, and E. W. Zegura, *An Architecture for Active Networking*. New York, NY, USA: Springer, 1997.

[35] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," in *Proc. DARPA Act. Netw. Conf. Expo.*, San Francisco, CA, USA, 2002, pp. 2–15.

[36] D. J. Wetherall, J. V. Guttag, and D. L. Tennenhouse, "ANTS: A toolkit for building and dynamically deploying network protocols," in *Proc. IEEE Open Architect. Netw. Program.*, San Francisco, CA, USA, 1998, pp. 117–129.

[37] J. E. van der Merwe, S. Rooney, L. Leslie, and S. Crosby, "The tempest-a practical framework for network programmability," *IEEE Netw.*, vol. 12, no. 3, pp. 20–28, May/Jun. 1998.

[38] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI veritas: Realistic and controlled network experimentation," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 3–14, 2006.

[39] D. S. Alexander *et al.*, "The SwitchWare active network architecture," *IEEE Netw.*, vol. 12, no. 3, pp. 29–36, May/Jun. 1998.

[40] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[41] W. Stallings, "Software-defined networks and OpenFlow," *Internet Protocol J.*, vol. 16, no. 1, pp. 1–6, 2013.

[42] *SDN-Ready White Box Data Center*. Accessed on May 8, 2016. [Online]. Available: http://www.pica8.com/open-networking/sdn-ready-white-box-data-center.php

[43] A. Voellmy and P. Hudak, "Nettle: Taking the sting out of programming network routers," in *Practical Aspects of Declarative Languages*. Heidelberg, Germany: Springer, 2011, pp. 235–249.

[44] N. Gude *et al.*, "NOX: Towards an operating system for networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, 2008.

[45] A. Voellmy, H. Kim, and N. Feamster, "Procera: A language for high-level reactive network control," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Helsinki, Finland, 2012, pp. 43–48.

[46] *Defining Openness for Open SDN and NFV*. Accessed on Nov. 16, 2015. [Online]. Available: https://www.sdxcentral.com/articles/featured/defining-open-sdn-nfv-a-primer-network-operators/2014/07/

[47] S. Gutz, A. Story, C. Schlesinger, and N. Foster, "Splendid isolation: A slice abstraction for software-defined networks," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Helsinki, Finland, 2012, pp. 79–84.

[48] M. Reitblatt, N. Foster, J. Rexford, and D. Walker, "Consistent updates for software-defined networks: Change you can believe in!" in *Proc. 10th ACM Workshop Hot Topics Netw.*, Cambridge, MA, USA, 2011, p. 7.

[49] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing software defined networks," in *Proc. NSDI*, Lombard, IL, USA, 2013, pp. 1–13.

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

[51] K. Barr *et al.*, "The VMware mobile virtualization platform: Is that a hypervisor in your pocket?" *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 4, pp. 124–135, 2010.

[52] C. Dixon *et al.*, "Software defined networking to support the software defined environment," *IBM J. Res. Develop.*, vol. 58, nos. 2–3, pp. 1–14, Mar./May 2014.

[53] R. Sherwood *et al.*, "FlowVisor: A network virtualization layer," OpenFlow Switch Consortium, Stanford Univ., Stanford, CA, USA, Tech. Rep., 2009, accessed on Aug. 1, 2016. [Online]. Available: http://archive.openflow.org/downloads/technicalreports/openflow-tr-2009-1-flowvisor.pdf

[54] E. Salvadori, R. D. Corin, A. Broglio, and M. Gerola, "Generalizing virtual network topologies in OpenFlow-based networks," in *Proc. IEEE Glob. Telecommun. Conf. (GLOBECOM)*, Houston, TX, USA, 2011, pp. 1–6.

[55] Z. Bozakov and P. Papadimitriou, "AutoSlice: Automated and scalable slicing for software-defined networks," in *Proc. ACM Conf. CoNEXT Student Workshop*, Nice, France, 2012, pp. 3–4.

[56] P. Skoldstrom and W. John, "Implementation and evaluation of a carrier-grade OpenFlow virtualization scheme," in *Proc. 2nd Eur. Workshop Softw. Defined Netw. (EWSDN)*, Berlin, Germany, 2013, pp. 75–80.

[57] P. Lin, J. Bi, and H. Hu, "VCP: A virtualization cloud platform for SDN intra-domain production network," in *Proc. 20th IEEE Int. Conf. Netw. Protocols (ICNP)*, Austin, TX, USA, 2012, pp. 1–2.

[58] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *IEEE Internet Comput.*, vol. 17, no. 2, pp. 20–27, Mar./Apr. 2013.

[59] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker, "Modular SDN programming with pyretic," USENIX Tech. Rep., 2013, accessed on Aug. 1, 2016. [Online]. Available: https://www.cs.princeton.edu/~jrex/papers/pyretic13.pdf

[60] A. Blenk, A. Basta, and W. Kellerer, "HyperFlex: An SDN virtualization architecture with flexible hypervisor function allocation," in *Proc. IFIP/IEEE IM*, Ottawa, ON, Canada, 2015, pp. 397–405.

[61] L. Velasco, A. Asensio, J. L. Berral, A. Castro, and V. López, "Towards a carrier SDN: An example for elastic inter-datacenter connectivity," *Opt. Exp.*, vol. 22, no. 1, pp. 55–61, 2014.

[62] V. Pandey, "Towards widespread SDN adoption: Need for synergy between photonics and SDN within the data center," in *Proc. IEEE Photon. Soc. Summer Topical Meeting Series*, Waikoloa, HI, USA, 2013, pp. 227–228.

[63] J. Kempf, Y. Zhang, R. Mishra, and N. Beheshti, "Zeppelin—A third generation data center network virtualization technology based on SDN and MPLS," in *Proc. IEEE 2nd Int. Conf. Cloud Netw. (CloudNet)*, San Francisco, CA, USA, 2013, pp. 1–9.

[64] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[65] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Comput. Netw.*, vol. 71, pp. 1–30, Oct. 2014.

[66] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[67] A. Gupta *et al.*, "SDX: A software defined Internet exchange," in *Proc. ACM Conf. SIGCOMM*, Chicago, IL, USA, 2014, pp. 551–562.

[68] V. Kotronis, X. Dimitropoulos, and B. Ager, "Outsourcing the routing control logic: Better Internet routing based on SDN principles," in *Proc. 11th ACM Workshop Hot Topics Netw.*, Redmond, WA, USA, 2012, pp. 55–60.

[69] J. Costa-Requena, "SDN integration in LTE mobile backhaul networks," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Phuket, Thailand, 2014, pp. 264–269.

[70] P. Berthou, "Leveraging SDN for the 5G networks," in *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture*. Hoboken, NJ, USA: Wiley, 2015, pp. 61–80.

[71] D. B. Rawat, S. Shetty, and C. Xin, "Stackelberg-game-based dynamic spectrum access in heterogeneous wireless systems," *IEEE Syst. J.*, vol. 10, no. 4, 2016.

[72] J. Sánchez, I. M. B. Yahia, N. Crespi, T. Rasheed, and D. Siracusa, "Softwarized 5G networks resiliency with self-healing," in *Proc. 1st Int. Conf. 5G Ubiquitous Connectivity (5GU)*, 2014, pp. 229–233.

[73] A. Gember, C. Dragga, and A. Akella, "ECOS: Leveraging software-defined networks to support mobile application offloading," in *Proc. 8th ACM/IEEE Symp. Architect. Netw. Commun. Syst.*, Austin, TX, USA, 2012, pp. 199–210.

[74] J. Lee *et al.*, "meSDN: Mobile extension of SDN," in *Proc. 5th Int. Workshop Mobile Cloud Comput. Services*, Bretton Woods, NH, USA, 2014, pp. 7–14.

[75] K.-K. Yap *et al.*, "The Stanford openroads deployment," in *Proc. 4th ACM Int. Workshop Exp. Eval. Characterization*, Beijing, China, 2009, pp. 59–66.

[76] R. Muñoz, R. Vilalta, R. Casellas, and R. Martínez, "SDN orchestration and virtualization of heterogeneous multi-domain and multi-layer transport networks: The STRAUSS approach," in *Proc. IEEE Int. Black Sea Conf. Commun. Netw. (BlackSeaCom)*, Constanta, Romania, 2015, pp. 142–146.

[77] T. Szyrkowiec *et al.*, "Demonstration of SDN based optical network virtualization and multidomain service orchestration," in *Proc. 3rd Eur. Workshop Softw. Defined Netw. (EWSDN)*, Budapest, Hungary, 2014, pp. 137–138.

[78] Q. Qi, W. Wang, X. Gong, and X. Que, "A SDN-based network virtualization architecture with autonomie management," in *Proc. Globecom Workshops (GC Wkshps)*, Austin, TX, USA, 2014, pp. 178–182.

[79] M. S. Malik, M. Montanari, J. H. Huh, R. B. Bobba, and R. H. Campbell, "Towards SDN enabled network control delegation in clouds," in *Proc. 43rd Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw. (DSN)*, Budapest, Hungary, 2013, pp. 1–6.

[80] R. D. Corin, M. Gerola, R. Riggio, F. De Pellegrini, and E. Salvadori, "VeRTIGO: Network virtualization and beyond," in *Proc. Eur. Workshop Softw. Defined Netw. (EWSDN)*, Darmstadt, Germany, 2012, pp. 24–29.

[81] D. B. Rawat, M. Song, and S. Shetty, *Dynamic Spectrum Access for Wireless Networks*. Cham, Switzerland: Springer, 2015.

[82] D. B. Rawat and N. Sharma, "Wireless network virtualization for enhancing security: Status, challenges and perspectives," in *Proc. IEEE SoutheastCon*, Norfolk, VA, USA, 2016, pp. 1–8.

[83] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "SoftRAN: Software defined radio access network," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, Hong Kong, 2013, pp. 25–30.

[84] M. Peng, C. Wang, V. Lau, and H. V. Poor, "Fronthaul-constrained cloud radio access networks: Insights and challenges," *IEEE Wireless Commun.*, vol. 22, no. 2, pp. 152–160, Apr. 2015.

[85] M. Y. Arslan, K. Sundaresan, and S. Rangarajan, "Software-defined networking in cellular radio access networks: Potential and challenges," *IEEE Commun. Mag.*, vol. 53, no. 1, pp. 150–156, Jan. 2015.

[86] M. Yang *et al.*, "OpenRAN: A software-defined ran architecture via virtualization," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 549–550, 2013.

[87] A. P. Bianzino, C. Chaudet, D. Rossi, and J.-L. Rougier, "A survey of green networking research," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 1, pp. 3–20, 1st Quart., 2012.

[88] D. Staessens, S. Sharma, D. Colle, M. Pickavet, and P. Demeester, "Software defined networking: Meeting carrier grade requirements," in *Proc. 18th IEEE Workshop Local Metropol. Area Netw. (LANMAN)*, Chapel Hill, NC, USA, 2011, pp. 1–6.

[89] B. Heller *et al.*, "ElasticTree: Saving energy in data center networks," in *Proc. NSDI*, vol. 10. San Jose, CA, USA, 2010, pp. 249–264.

[90] L. Schehlmann, S. Abt, and H. Baier, "Blessing or curse? Revisiting security aspects of software-defined networking," in *Proc. 10th Int. Conf. Netw. Service Manag. (CNSM)*, Rio de Janeiro, Brazil, 2014, pp. 382–387.

[91] S. Shin and G. Gu, "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)," in *Proc. 20th IEEE Int. Conf. Netw. Protocols (ICNP)*, Austin, TX, USA, 2012, pp. 1–6.

[92] P.-W. Chi, C.-T. Kuo, H.-M. Ruan, S.-J. Chen, and C.-L. Lei, "An AMI threat detection mechanism based on SDN networks," in *Proc. SECURWARE*, Lisbon, Portugal, 2014, pp. 208–211.

[93] J. François and O. Festor, "Anomaly traceback using software defined networking," in *Proc. Nat. Conf. Parallel Comput. Technol. (PARCOMPTECH)*, Atlanta, GA, USA, 2015, pp. 203–208.

[94] R. Jin and B. Wang, "Malware detection for mobile devices using software-defined networking," in *Proc. 2nd GENI Res. Educ. Exp. Workshop (GREE)*, Salt Lake City, UT, USA, 2013, pp. 81–88.

[95] K. Giotis, G. Androulidakis, and V. Maglaris, "Leveraging SDN for efficient anomaly detection and mitigation on legacy networks," in *Proc. 3rd Eur. Workshop Softw. Defined Netw. (EWSDN)*, Budapest, Hungary, 2014, pp. 85–90.

[96] S. Lim, J.-I. Ha, H. Kim, Y. Kim, and S. Yang, "A SDN-oriented DDoS blocking scheme for botnet-based attacks," in *Proc. 6th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Shanghai, China, 2014, pp. 63–68.

[97] R. Sahay, G. Blanc, Z. Zhang, and H. Debar, "Towards autonomic DDoS mitigation using software defined networking," in *Proc. NDSS Workshop Security Emerg. Netw. Technol. (SENT)*, San Diego, CA, USA, 2015, pp. 1–7.

[98] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "NICE: Network intrusion detection and countermeasure selection in virtual network systems," *IEEE Trans. Depend. Secure Comput.*, vol. 10, no. 4, pp. 198–211, Jul./Aug. 2013.

[99] A. G. P. Lobato, U. da Rocha Figueiredo, and O. C. M. B. Duarte, "An architecture for intrusion prevention using software defined networks," in *Proc. WNetVirt*, Rio de Janeiro, Brazil, 2013, p. 1.

[100] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks," in *Proc. LISA*, vol. 99. Seattle, WA, USA, 1999, pp. 229–238.

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

[102] M. Abliz, "Internet denial of service attacks and defense mechanisms," Dept. Comput. Sci., Univ. Pittsburgh, Pittsburgh, PA, USA, Tech. Rep. TR-11-178, 2011.

[103] D. Kreutz, F. M. V. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, Hong Kong, 2013, pp. 55–60.

[104] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "OpenFlow random host mutation: Transparent moving target defense using software defined networking," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Helsinki, Finland, 2012, pp. 127–132.

[105] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures," in *Proc. NDSS*, 2015, pp. 1–15.

[106] X. Du, M.-Z. Wang, X. Zhang, and L. Zhu, "Traffic-based malicious switch detection in SDN," *Int. J. Security Appl.*, vol. 8, no. 5, pp. 119–130, 2014.

[107] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: Detecting security attacks in software-defined networks," in *Proc. Netw. Distrib. Syst. Security (NDSS) Symp.*, San Diego, CA, USA, 2015, pp. 1–15.

[108] H. Hu, W. Han, G.-J. Ahn, and Z. Zhao, "FLOWGUARD: Building robust firewalls for software-defined networks," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, Chicago, IL, USA, 2014, pp. 97–102.

[109] S. K. Fayazbakhsh, V. Sekar, M. Yu, and J. C. Mogul, "FlowTags: Enforcing network-wide policies in the presence of dynamic middlebox actions," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, Hong Kong, 2013, pp. 19–24.

[110] Z. Qazi *et al.*, "Practical and incremental convergence between SDN and middleboxes," in *Proc. Open Netw. Summit*, Santa Clara, CA, USA, 2013, pp. 1–15.

[111] B. Anwer, T. Benson, N. Feamster, D. Levin, and J. Rexford, "A slick control plane for network middleboxes," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, Hong Kong, 2013, pp. 147–148.

[112] M. Canini, D. Venzano, P. Perešíni, D. Kostić, and J. Rexford, "A NICE way to test OpenFlow applications," in *Proc. NSDI*, vol. 12. San Jose, CA, USA, 2012, pp. 127–140.

[113] P. Porras *et al.*, "A security enforcement kernel for OpenFlow networks," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Helsinki, Finland, 2012, pp. 121–126.

[114] S. E. Schechter, J. Jung, and A. W. Berger, "Fast detection of scanning worm infections," in *Recent Advances in Intrusion Detection*. Heidelberg, Germany: Springer, 2004, pp. 59–81.

[115] J. Twycross and M. M. Williamson, "Implementing and testing a virus throttle," in *Proc. USENIX Security*, 2003, p. 20.

[116] Y. Gu, A. McCallum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," in *Proc. 5th ACM SIGCOMM Conf. Internet Meas.*, Berkeley, CA, USA, 2005, p. 32.

[117] M. V. Mahoney, "Network traffic anomaly detection based on packet bytes," in *Proc. ACM Symp. Appl. Comput.*, Melbourne, FL, USA, 2003, pp. 346–350.

[118] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, Garden Grove, CA, USA, 2015, pp. 77–81.

[119] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proc. IEEE 35th Conf. Local Comput. Netw. (LCN)*, Denver, CO, USA, 2010, pp. 408–415.

[120] C. Schlesinger, A. Story, S. Gutz, N. Foster, and D. Walker, "Splendid isolation: Language-based security for software-defined networks," in *Proc. Workshop Hot Topics Softw. Defined Netw.*, Helsinki, Finland, 2012, pp. 79–84.

[121] M. Ambrosin, M. Conti, F. De Gaspari, and R. Poovendran, "LinesWitch: Efficiently managing switch flow in software-defined networking while effectively tackling DoS attacks," in *Proc. 10th ACM Symp. Inf. Comput. Commun. Security*, Singapore, 2015, pp. 639–644.

[122] J. M. Dover. *A Denial of Service Attack Against the Open Floodlight SDN Controller*. Accessed on Aug. 1, 2016. [Online]. Available: http://dovernetworks.com/wp-content/uploads/2013/12/OpenFloodlight-12302013.pdf

[123] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, 2004.

[124] M. Dražar, M. Vizváry, and J. Vykopal, "Similarity as a central approach to flow-based anomaly detection," *Int. J. Netw. Manag.*, vol. 24, no. 4, pp. 318–336, 2014.

[125] A.-S. Kim, H.-J. Kong, S.-C. Hong, S.-H. Chung, and J. W. Hong, "A flow-based method for abnormal network traffic detection," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp.*, vol. 1. Seoul, South Korea, 2004, pp. 599–612.

[126] P. Casas, J. Mazel, and P. Owezarski, "Unsupervised network intrusion detection systems: Detecting the unknown without knowledge," *Comput. Commun.*, vol. 35, no. 7, pp. 772–783, 2012.

[127] Z. Li, Y. Gao, and Y. Chen, "HiFIND: A high-speed flow-level intrusion detection approach with DoS resiliency," *Comput. Netw.*, vol. 54, no. 8, pp. 1282–1299, 2010.

[128] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "DDoS attack protection in the era of cloud computing and software-defined networking," *Comput. Netw.*, vol. 81, pp. 308–319, Apr. 2015.

[129] R. Kloti, V. Kotronis, and P. Smith, "OpenFlow: A security analysis," in *Proc. 21st IEEE Int. Conf. Netw. Protocols (ICNP)*, Göttingen, Germany, 2013, pp. 1–6.

[130] N. Handigol, B. Heller, V. Jeyakumar, D. Maziéres, and N. McKeown, "Where is the debugger for my software-defined network?" in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Helsinki, Finland, 2012, pp. 55–60.

[131] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, 2000.

[132] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, Hong Kong, 2013, pp. 151–152.

[133] A. Bates, K. Butler, A. Haeberlen, M. Sherr, and W. Zhou, "Let SDN be your eyes: Secure forensics in data center networks," in *Proc. NDSS Workshop Security Emerg. Netw. Technol. (SENT)*, 2014, pp. 1–7.

[134] M. Suh, S. H. Park, B. Lee, and S. Yang, "Building firewall over the software-defined network controller," in *Proc. 16th Int. Conf. Adv. Commun. Technol. (ICACT)*, 2014, pp. 744–748.

[135] W. P. de Jesus, D. A. da Silva, R. T. de Sousa, Jr., and F. V. L. da Frota, "Analysis of SDN contributions for cloud computing security," in *Proc. IEEE/ACM 7th Int. Conf. Utility Cloud Comput. (UCC)*, London, U.K., 2014, pp. 922–927.

[136] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," in *Proc. Green Cloud*, 2010, pp. 1–6.

[137] B. G. Assefa and O. Ozkasap, "State-of-the-art energy efficiency approaches in software defined networking," in *Proc. ICN*, San Francisco, CA, USA, 2015, p. 268.

[138] F. Kaup, S. Melnikowitsch, and D. Hausheer, "Measuring and modeling the power consumption of OpenFlow switches," in *Proc. 10th Int. Conf. Netw. Service Manag. (CNSM)*, Rio de Janeiro, Brazil, 2014, pp. 181–186.

[139] C. E. Hopps. (2000). *Analysis of an Equal-Cost Multi-Path Algorithm*. Accessed on Aug. 1, 2016. [Online]. Available: https://tools.ietf.org/html/rfc2992

[140] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. NSDI*, vol. 10. San Jose, CA, USA, 2010, p. 19.

[141] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection," in *Proc. IEEE INFOCOM*, Shanghai, China, 2011, pp. 1629–1637.

[142] A. R. Curtis *et al.*, "DevoFlow: Scaling flow management for high-performance networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 254–265, 2011.

[143] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 351–362, 2010.

[144] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proc. Internet Netw. Manag. Conf. Res. Enterprise Netw.*, San Jose, CA, USA, 2010, p. 3.

[145] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "BalanceFlow: Controller load balancing for OpenFlow networks," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Intell. Syst. (CCIS)*, vol. 2. Hangzhou, China, 2012, pp. 780–785.

[146] S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Helsinki, Finland, 2012, pp. 19–24.

[147] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in SDN/OSPF hybrid network," in *Proc. IEEE 22nd Int. Conf. Netw. Protocols (ICNP)*, Raleigh, NC, USA, 2014, pp. 563–568.

[148] A. Chanda, C. Westphal, and D. Raychaudhuri, "Content based traffic engineering in software defined information centric networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Turin, Italy, 2013, pp. 357–362.

[149] B. Puype, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester, "Multilayer traffic engineering for energy efficiency," *Photon. Netw. Commun.*, vol. 21, no. 2, pp. 127–140, 2011.

[150] A. Ruiz-Rivera, K.-W. Chin, and S. Soh, "GreCo: An energy aware controller association algorithm for software defined networks," *IEEE Commun. Lett.*, vol. 19, no. 4, pp. 541–544, Apr. 2015.

[151] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "OpenTM: Traffic matrix estimator for OpenFlow networks," in *Passive and Active Measurement*. Heidelberg, Germany: Springer, 2010, pp. 201–210.

[152] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "PayLess: A low cost network monitoring framework for software defined networks," in *Proc. IEEE Netw. Oper. Manag. Symp. (NOMS)*, Kraków, Poland, 2014, pp. 1–9.

[153] C. Yu *et al.*, "FlowSense: Monitoring network utilization with zero measurement cost," in *Passive and Active Measurement*. Heidelberg, Germany: Springer, 2013, pp. 31–41.

[154] J. Suh, T. T. Kwon, C. Dixon, W. Felter, and J. Carter, "OpenSample: A low-latency, sampling-based measurement platform for SDN," in *Proc. ICDCS*, Madrid, Spain, 2014, pp. 1–10.

[155] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with OpenSketch," in *Proc. NSDI*, vol. 13. Lombard, IL, USA, 2013, pp. 29–42.

[156] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proc. 7th Conf. Emerg. Netw. Exp. Technol.*, Tokyo, Japan, 2011, p. 8.

[157] P. Kazemian *et al.*, "Real time network policy checking using header space analysis," in *Proc. NSDI*, Lombard, IL, USA, 2013, pp. 99–111.

[158] R. McGeer, "A safe, efficient update protocol for OpenFlow networks," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Helsinki, Finland, 2012, pp. 61–66.

[159] T. Mizrahi and Y. Moses, "Time-based updates in software defined networks," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, Hong Kong, 2013, pp. 163–164.

[160] C. R. Meiners, A. X. Liu, and E. Torng, "TCAM Razor: A systematic approach towards minimizing packet classifiers in TCAMs," in *Proc. IEEE Int. Conf. Netw. Protocols (ICNP)*, Beijing, China, 2007, pp. 266–275.

[161] K. Kannan and S. Banerjee, "Compact TCAM: Flow entry compaction in TCAM for power aware SDN," in *Distributed Computing and Networking*. Heidelberg, Germany: Springer, 2013, pp. 439–444.

[162] C. R. Meiners, A. X. Liu, and E. Torng, "Bit Weaving: A non-prefix approach to compressing packet classifiers in TCAMs," *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 488–500, Apr. 2012.

[163] Y. Kanizo, D. Hay, and I. Keslassy, "Palette: Distributing tables in software-defined networks," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 545–549.

[164] H. Huang, P. Li, S. Guo, and B. Ye, "The joint optimization of rules allocation and traffic engineering in software defined network," in *Proc. IEEE 22nd Int. Symp. Qual. Service (IWQoS)*, Hong Kong, 2014, pp. 141–146.

[165] F. Giroire, J. Moulierac, and T. K. Phan, "Optimizing rule placement in software-defined networks for energy-aware routing," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Austin, TX, USA, 2014, pp. 2523–2529.

[166] M. Dong, H. Li, K. Ota, and J. Xiao, "Rule caching in SDN-enabled mobile access networks," *IEEE Netw.*, vol. 29, no. 4, pp. 40–45, Jul./Aug. 2015.

[167] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "Rule-caching algorithms for software-defined networks," Tech. Rep., 2014, accessed on Aug. 1, 2016. [Online]. Available: http://www.cs.princeton.edu/~nkatta/papers/cacheflow-long14.pdf

[168] M. Jarschel and R. Pries, "An OpenFlow-based energy-efficient data center approach," in *Proc. ACM SIGCOMM Conf. Appl. Technol. Architect. Protocols Comput. Commun.*, Helsinki, Finland, 2012, pp. 87–88.

[169] D. Kakadia and V. Varma. (2012). *Energy Efficient Data Center Networks—A SDN Based Approach*. Accessed on Aug. 1, 2016. [Online]. Available: http://goo.gl/6o5MBj

[170] J. M. Wang, Y. Wang, X. Dai, and B. Bensaou, "SDN-based multi-class QoS-guaranteed inter-data center traffic management," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, 2014, pp. 401–406.

[171] A. Sallahi and M. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 19, no. 1, pp. 30–33, Jan. 2015.

[172] S.-H. Wang, "Virtual machine placement for energy efficiency and QoS in software defined datacenter networks," Ph.D. dissertation, College Elect. Comput. Eng., Nat. Chiao Tung Univ., Hsinchu, Taiwan, 2013.

[173] D. Li, Y. Shang, and C. Chen, "Software defined green data center network with exclusive routing," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, 2014, pp. 1743–1751.
[174] A. Markiewicz, P. N. Tran, and A. Timm-Giel, "Energy consumption optimization for software defined networks considering dynamic traffic," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, 2014, pp. 155–160.
[175] J. Naous, D. Erickson, G. A. Covington, G. Appenzeller, and N. McKeown, "Implementing an OpenFlow switch on the NetFPGA platform," in *Proc. 4th ACM/IEEE Symp. Architect. Netw. Commun. Syst.*, San Jose, CA, USA, 2008, pp. 1–9.
[176] T. H. Vu, V. C. Luc, N. T. Quan, N. H. Thanh, and P. N. Nam, "Energy saving for OpenFlow switch on the NetFPGA platform based on queue engineering," *SpringerPlus*, vol. 4, no. 1, p. 64, 2015.
[177] Y. Hua, X. Liu, and D. Feng, "Smart in-network deduplication for storage-aware SDN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 509–510, 2013.
[178] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*, vol. 11997. Reading, MA, USA: Addison-Wesley, 1997.
[179] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and principles of Internet traffic engineering," Tech. Rep., 2002, accessed on Aug. 1, 2016. [Online]. Available: https://tools.ietf.org/html/rfc3272
[180] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 118–124, Oct. 2002.
[181] D. O. Awduche and B. Jabbari, "Internet traffic engineering using multi-protocol label switching (MPLS)," *Comput. Netw.*, vol. 40, no. 1, pp. 111–129, 2002.
[182] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "Infinite CacheFlow in software-defined networks," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, Chicago, IL, USA, 2014, pp. 175–180.
[183] E. Spitznagel, D. Taylor, and J. Turner, "Packet classification using extended TCAMs," in *Proc. 11th IEEE Int. Conf. Netw. Protocols*, Atlanta, GA, USA, 2003, pp. 120–131.
[184] X. Jin *et al.*, "Dynamic scheduling of network updates," in *Proc. ACM Conf. SIGCOMM*, Chicago, IL, USA, 2014, pp. 539–550.
[185] S. Sezer *et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, Jul. 2013.
[186] Á. L. V. Caraguay, A. B. Peral, L. I. B. López, and L. J. G. Villalba, "SDN: Evolution and opportunities in the development IoT applications," *Int. J. Distrib. Sens. Netw.*, vol. 2014, 2014, Art. no. 735142.
[187] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the Internet-of-Things," in *Proc. IEEE Netw. Oper. Manag. Symp. (NOMS)*, Kraków, Poland, 2014, pp. 1–9.
[188] N. Dorsch, F. Kurtz, H. Georg, C. Hägerling, and C. Wietfeld, "Software-defined networking for smart grid communications: Applications, challenges and advantages," in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Venice, Italy, 2014, pp. 422–427.

Danda B. Rawat (S'07–M'09–SM'13) received the Ph.D. degree in electrical and computer engineering from Old Dominion University, Norfolk, VA, USA. He is an Associate Professor with the Department of Electrical Engineering and Computer Science, Howard University, Washington, DC, USA. He was with the College of Engineering and Information Technology, Georgia Southern University, Statesboro, GA, USA, as a Faculty Member until 2016. He has published over 120 scientific/technical articles and eight books. His research focuses on wireless communication networks, cybersecurity, cyber-physical systems, Internet-of-Things, big data analytics, wireless virtualization, software-defined networks, smart grid systems, wireless sensor networks, and vehicular/wireless ad-hoc networks. His research is supported by U.S. National Science Foundation, University Sponsored Programs and Center for Sustainability grants. He was a recipient of the NSF Faculty Early Career Development (CAREER) Award and the Outstanding Research Faculty Award (Award for Excellence in Scholarly Activity) in 2015, from Allen E. Paulson College of Engineering and Technology, Georgia Southern University among others, and nominated for the Faculty Award of Excellence in Teaching in 2016 from College of Engineering and Technology, Georgia Southern University. He has been serving as an Editor/Guest Editor for over ten international journals. He serves as the Web-Chair for the IEEE INFOCOM 2016/2017, served as the Student Travel Grant Co-Chair of the IEEE INFOCOM 2015, and the Track Chair for Wireless Networking and Mobility of the IEEE CCNC 2016 and Communications Network and Protocols of the IEEE AINA 2015. He served as the Program Chair, the General Chair, and the Session Chair for numerous international conferences and workshops, and served as a Technical Program Committee Member for several international conferences including the IEEE INFOCOM, the IEEE GLOBECOM, the IEEE CCNC, the IEEE GreenCom, the IEEE AINA, the IEEE ICC, the IEEE WCNC, and the IEEE VTC conferences. He is the Founder and the Director of the Cyber-Security and Wireless Networking Innovations (CWiNs) Research Laboratory. He served as the Vice Chair of the Executive Committee of the IEEE Savannah Section and Webmaster for the section from 2013 to 2017. He is a member of ACM and ASEE.

Swetha R. Reddy (S'15–M'16) received the bachelor's degree from Jawaharlal Nehru Technological University, India, in 2014, and the master's degree in electrical and electronics systems from the Department of Electrical Engineering, Georgia Southern University, Statesboro GA, USA, in 2016. She was a Graduate Research Assistant with the Cybersecurity, Wireless Systems and Networking Innovations Laboratory, College of Engineering and Information Technology, Georgia Southern University, Statesboro, GA, USA. Her research lies in the areas of wireless communication networks, software-defined networks, and network security.