Hindawi
Security and Communication Networks
Volume 2018, Article ID 2459154, 14 pages
https://doi.org/10.1155/2018/2459154

Review Article

A Survey on Security-Aware Measurement in SDN

Heng Zhang,[1] Zhiping Cai,[1] Qiang Liu,[1] Qingjun Xiao,[2] Yangyang Li,[3] and Chak Fone Cheang[4]

[1]*College of Computer, National University of Defense Technology, Changsha, Hunan 410073, China*
[2]*School of Computer Science and Engineering, Southeast University, Nanjing, Jiangsu 211189, China*
[3]*Innovation Center, China Academy of Electronics and Information Technology, Beijing 100041, China*
[4]*Faculty of Information Technology, Macau University of Science and Technology, Macau*

Correspondence should be addressed to Zhiping Cai; zpcai@nudt.edu.cn

Received 7 November 2017; Accepted 26 February 2018; Published 24 April 2018

Academic Editor: Zhen Liu

Software-defined networking (SDN) is one of the most prevailing networking paradigms in current and next-generation networks. Basically, the highly featured separation of control and data planes makes SDN a proper solution towards many practical problems that challenge legacy networks, for example, energy efficiency, dynamic network configuration, agile network measurement, and flexible network deployment. Although the SDN and its applications have been extensively studied for several years, the research of SDN security is still in its infancy. Typically, the SDN suffers from architecture defect and OpenFlow protocol loopholes such as single controller problem, deficiency of communication verification, and network resources constraint. Hence, network measurement is a fundamental technique of protecting SDN against the above security threats. Specifically, network measurement aims to understand and quantify a variety of network behaviors to facilitate network management and monitoring, anomaly detection, network troubleshooting, and the establishment of security mechanisms. In this paper, we present a systematic survey on security-aware measurement technology in SDN. In particular, we first review the basic architecture of SDN and corresponding security challenges. Then, we investigate two performance measurement techniques in SDN, namely, link latency and available bandwidth measurements. After that, we further provide a general overview of topology measurement in SDN including intradomain and interdomain topology discovering techniques. Finally, we list three interesting future directions of security-aware measurement in SDN followed by giving conclusion remarks.

1. Introduction

Recently, software-defined networking (SDN) has been recognized as one of the most prevailing networking paradigms for next-generation networks. Due to the significant characteristic of separating control and data planes, SDN provides an efficient solution for supporting diverse network functions, dynamic network management, and network security [1]. Hence, the SDN technology is becoming a very active research area in academia, and there are several successful application cases in information and communication technology (ICT) industry. For example, Google has already experimented with and deployed B4 [2] network and reported an unprecedented network utilization of 95%. Moreover, open network foundation (ONF) in the ICT industry was established to facilitate technical research, international corporation, system design, and deployment of SDN. What is more, SDN related scientific papers in top conferences and other networking venues suggest that SDN is scalable for adding more intelligent and complex control logic to the centralized control plane for optimized network flow management and status monitoring, which are very important in the practical usage of SDN.

Despite great advances of SDN research and deployment, research on SDN security is still in its infancy [3]. Actually, there are a variety of security threats challenging SDN in practical usage. The reasons mainly are attributed to different security vulnerabilities in the protocol and the implementation of SDN. In the perspective of hardware, both the SDN controller and underlying switches are vulnerable

to security threats due to the centralized architecture and limited network resources of switches. Taking the SDN controller as an example, it functions as the core of a network and takes the responsibility of managing the whole network. It is straightforward that the overall security of the target network would be significantly reduced if the controller is compromised by attackers. On the other hand, adversaries will be able to launch denial-of-service (DoS) attacks against SDN switches due to their limited computational and memory resources, which induces a significant decrease of the availability of SDN. In the perspective of protocol, the widely used OpenFlow protocol suffers from unexpected flaws and security vulnerabilities during its design and evolution. For example, the transport layer security (TLS) procedure is discarded when the SDN controller and underlying switches need to exchange information with each other at the verification stage. Such operation is vulnerable to man-in-the-middle attacks. In rule-match process, since decisions on new flows are made by the centralized controller, the characteristic will induce distributed DoS (DDoS) from multiple adversaries, for example, a number of bots from a botnet. To effectively address the aforementioned security challenges towards SDN, a promising solution is to enhance security awareness during real-time network measurement then to make proper predictions regarding these threats. In this paper, we present a systematic survey on security-aware network measurement technology in SDN. Specifically, we analyze the rationale of different security threats towards SDN by considering its architecture. Then, we investigate two types of performance measurement techniques, namely, link latency and available bandwidth measurements. After that, we provide an overview of topology measurement technology in SDN, including intradomain and interdomain topology discovering techniques. Finally, we present three future directions of security-aware measurement in SDN. We argue that the survey can bring beneficial references to SDN industrial development and corresponding academia research.

The following part of this paper is structured as follows. In Section 2, we review the basic architecture of SDN and corresponding security challenges. In Section 3, we investigate two performance measurement techniques in SDN, namely, link latency and available bandwidth measurements. In Section 4, we provide a general overview of topology measurement in SDN including intradomain and interdomain topology discovering techniques. Three interesting future directions of security-aware measurement are proposed in Section 5, and conclusion remarks are given in Section 6.

2. Architecture of SDN and Security Challenges

To date, SDN overcomes the limitation of legacy network by decoupling network control plane from the forwarding plane and adding programmable functions. This emphasizes the fact that this separation idea provides administrators with ease of resource provisioning and programmability to change and control the characteristics of whole network [4]. Through dynamic, customized, and proprietary-free software written by network operators in SDN structure, they can manage, configure, and optimize network resources in an easy and quick way. From this perspective, SDN should be a very influential and efficient solution of solving current network problems and improving network security.

2.1. Basic Architecture of SDN. Software-defined networking is a new emerging network paradigm in recent years. The novel ideas of decoupling network control and data transfer and ensuring programmability in the network have acquired the most attention in both industry and academia [5]. SDN consists of a set of underlying switches and a centralized control entities, both of which are programmable. The upper layer is set up for users to enforce its policies without considering the detail of underlying network structure through northbound API; the control layer mainly focuses on network management based on a bunch of strategies and the instructions will be installed in switches through OpenFlow channel [6]. In recent years, with the continuous expansion of the network size and complexity of network structure and functionality, multifariousness of new network services has emerged, such as virtual cloud computing, large-scale data center, and various streaming media, which has put forward severe challenges in configuring, operating, and managing traditional network. A simplified view of SDN architecture is shown in Figure 1.

Since the SDN is logically centralized, controllers have a global visibility of the whole network and provide many benefits. For example, controller is a central entity that keeps in continuous contact with all network devices and gather statistics from them. It uses these statistics for routing, load balancing, and so on. Thus, they can dynamically optimize flow management and resources [7]. What is more, SDN provides a complete abstraction of underlying network, which gives network operators an easy way to configure network devices or services by the complexity of the whole network.

OpenFlow is a widely accepted standard protocol defined by the ONF organization. It is designed as a communication interface between control and infrastructure layer. It intellectually takes control and manipulates network hardware to accomplish forwarding actions as required. An OpenFlow switch should carry multiple flow tables, within which a huge amount of flow entries is contained. A switch looks up flow entries for forwarding decisions when flow comes in. Once the flow packet is matched, corresponding instructions will be executed. If no flow entries can match the packet, controller will take the responsibility [8]. To date, various OpenFlow versions have been released, and multiple new functions have been added during the promotion process.

2.2. Security Challenges in SDN. Current attacks have been conducted in a more complex way [9]. According to intrinsic structure of SDN, the security challenges can be concluded from two aspects: hardware-based challenges and protocol-based challenges.

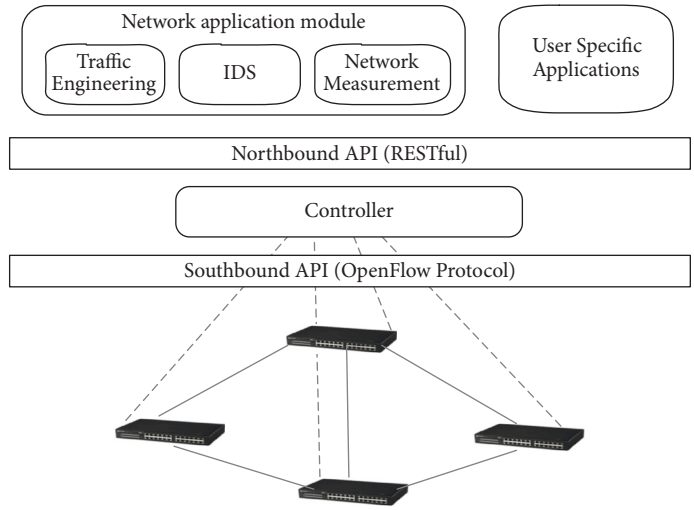

Figure 1: The typical structure of SDN.

Hardware-Based Challenges. Controller might be the primary choice for attackers because controller is the logically centralized device, which functions as the brain of a network [10]. Therefore, many traditional attack approaches and several new attack behaviors are very effective in making a controller disabled. For example, the classic DDoS attack can choose controller as a target. Attacker can produce a huge number of fake packets and send them to switches at the same time. All fake packets are regarded as new in switches, which will generate at least equal amount of fake flow requests to the controller. In this way, controller's computational resources will be drained in no time.

Switch has also been seen as a vulnerability in attacker's mind [11]. Basically, switches have poorer performance in hardware resources. Attackers can first attack the communication channel to cut down the link between controller and switches, so, according to OpenFlow protocol, the switches will change into fail-secure mode or standalone mode. This will dramatically affect the network performance. And attackers can forge a controller when switches tried to restore connection, and this will cause disasters as attackers have taken charge of the underlying network devices and control any flows traversed through switches.

Protocol-Based Challenges. Though there is a boom in OpenFlow development, there are still so many aspects that need to be considered. Take verification mechanism as an example [12]. The subsequent OpenFlow versions except the first edition make mutual authentication optional between controller and switches, and the coerced TLS has been cancelled during the initial time which can lead to data information leak. This loophole is obvious and could introduce very malicious actions such as man-in-the-middle attack. In this way, attackers can steal network information, modify flow rules in the switches, or even forge and insert wrong flow rules so as to take control of the whole network [13].

OpenFlow protocol may also induce system level security challenges. Controller commonly has several modules for efficient network management and monitoring, which can be regarded as third-party applications [14]. Once these modules have been compromised in controller, it will cause detrimental problem and induce unintentionally vulnerability to the whole system.

Therefore, in order to avoid these security challenges and establish a better and wiser network defense mechanism, we should apply newly developed SDN technologies to detect them in the first place and then respond to malicious actions in advance. To do so, we believe SDN measurement technologies should be the prime choice for the following reasons:

(1) Network measurement takes advantages of some certain approaches to understand and quantify network behaviors, which can be very helpful in detecting anonymous behaviors in advance.

(2) Network measurement metrics can also be very useful in making precautions and afterward reactions.

(3) Network measurement helps in understanding the network status in real-time, which can be applied to extensive areas such as network optimization, malfunction detection, and troubleshooting. These are all effective ways to establish a better security mechanism.

To this end, we will survey SDN network measurement technologies to manage the security-based challenges.

3. Performance Measurement in SDN

As current network is being pushed front and center, according to IDG's 2016 state of the network study, network has gotten more complicated. The metrics involved have gotten more important and complicated as well [15]. The handy network metrics that have been gathering all these years do not have to lose their values as they only matter when encountering measurement goals [16]. So, to verify desired network behaviors, aiming at managing security-aware challenges, we recommend link latency, available bandwidth, and network topology as our survey targets.

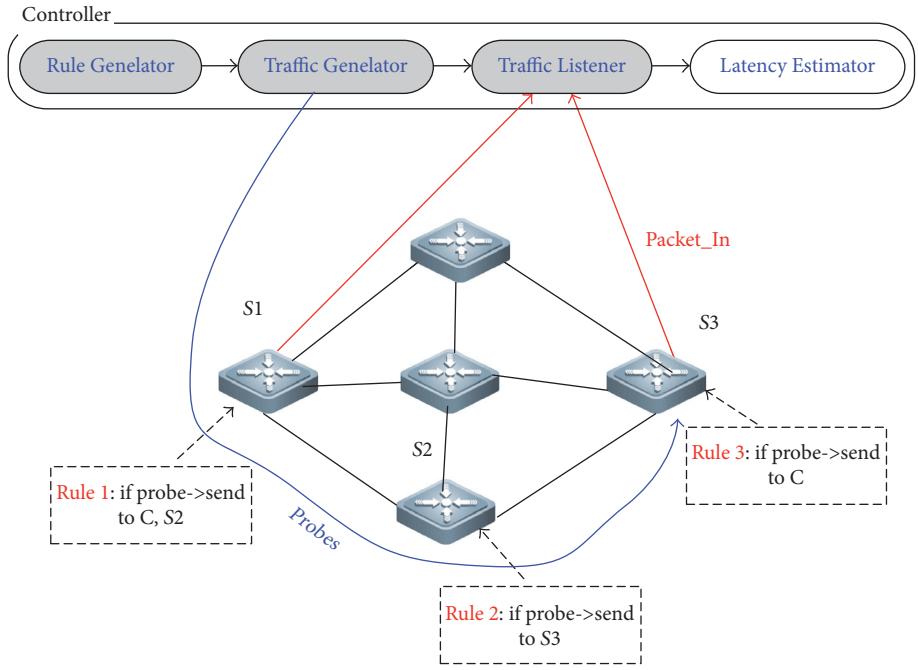

Figure 2: SLAM framework for latency monitoring in SDN.

3.1. Link Latency Measurement. Latency is one of the widely used performance metrics in computer networks, especially for those newly emerging applications which are sensitive to latency [17]. Network administrators could use latency to diagnose and detect abnormalities. For example, in many data center applications, such as e-commercial and e-banking, which have low tolerance to latency, they need to communicate across paths with no delay to realize real-time response, which makes latency a key metric for these latency-sensitive applications and administrators must detect network path latency constantly in order to manage data center networks as well as take actions when latency exceeds the threshold such as rerouting flows from high-delay paths [18].

In time-sensitive conditions, administrators have high demand in latency accuracy within a short time period. But, sometimes, we need network latency in a more general way. These separate delay measurements into two branches.

Active Delay Measurement. SLAM [19] is a typical latency monitoring framework that can measure latency between any two network switches along the path by dynamically sending probe packets to trigger control message (Packet_in and Packet_out message) from the top and end switch of a predefined path to the controller. SLAM then measures latency based on the arrival time of control message at the control plane.

To be specific, SLAM is deployed on centralized controller and combines four components: rule generator, traffic generator, traffic listener, and latency estimator (as shown in Figure 2). The latency computation process is composed of three steps: (1) Preselect end-to-end path and install specific flow monitoring rules on all switches along the path. (2) Controller sends constructed probe packets that match monitoring rules to switches and then the probe will traverse through the monitored path. Once the probe packet arrived at a switch, it will be regarded as a new flow and trigger Packet_in message to controller. (3) Controller estimates path's latency based on the timestamp in these massages. In the meantime, the estimation of delay acquires control link delay, so, in the first step, the first and last switches also generate notifications to controller [20]. SLAM can use Statistics_Request and Statistics_Reply message of OpenFlow to get control link RTT.

SLAM offers several advantages. It requires no switch hardware modifications and the path delay can be arbitrarily selected. It is quick and accurate in identifying high-delay paths by introducing little overhead. Moreover, SLAMs latency measurement framework is well-suited to SDN architecture. However, the accuracy of latency depends on process time of the first and last switches along the path, which varies constantly from time to time.

From SLAM, the premise of measuring latency is the timestamp extracted from the packet, which means that attaching a timestamp to header of a packet is a powerful feature that can be adopted by various SDN applications. DPTH [21] is a flexible and uniform labeling method. it indicates the time at which the packet enters the network. It proposes to timestamp all packets by the following steps (as shown in Figure 3):

(1) The ingress switch (either hardware or software switch) attaches a DPTH to all packets.

(2) Packets are forwarded through the network with the DHCP.

(3) The egress switch will remove the DHCP from every packet header.

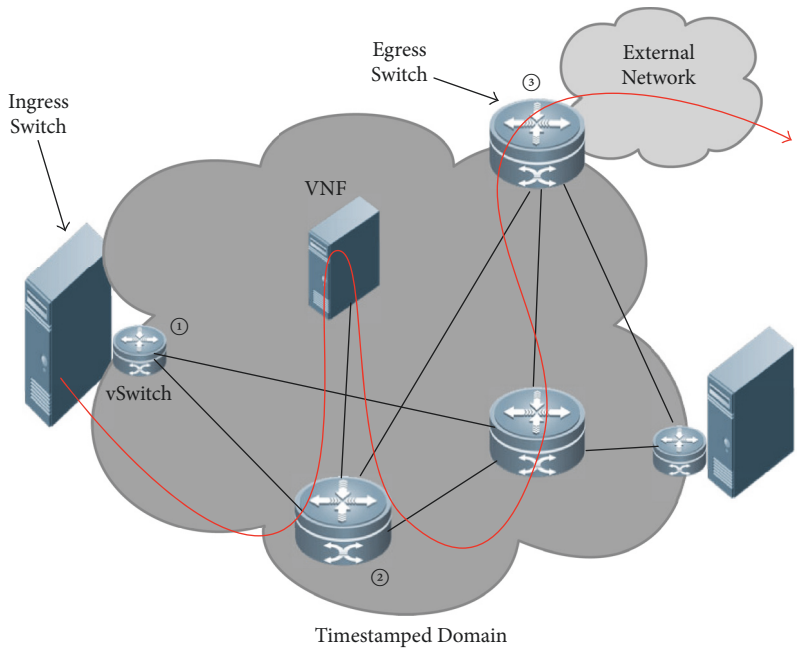

Figure 3: Timestamping all packets in DPTH.

When measuring latency between any two switches, DPTH only needs to send a timestamped from switch S_1 to switch S_2 and directly calculate the time difference as long as these switches have synchronized clocks. Also, DPTH offers a coloring-based method for multiple path latency estimation by alleviating the color bit. So even though DPTH adding additional header to all packets appears to be costly in the first place, the cost can be diverse based on specific application demands.

To make latency more accurate, as mentioned in SLAM, the processing delay of switches should be taken into consideration. MCPL [22] presented a comprehensive exploration of latency measurement and highlights the process delay as inbound latency, outbound latency, and equipment performance discrepancy (as shown in Figure 4)

MCPL also alleviate OpenFlow protocol to estimate the latency. When packet arrived at the switch, the timestamp will be recorded as T_1 and when the host receives the packet_in message, the timestamp will be recorded as T_2. Based on testbed design, the roundtrip time between host and switch can be ignored, so the inbound latency is $T_2 - T_1$. The outbound latency is composed of 4 parts: the OpenFlow agent parses received messages, the addition or deletion of rule, the hardware updating time, and final installation delay. Also, the time can be acquired through timestamp in the packet.

Passive Delay Measurement. FlowTrace [23] is a simple tool to locate every forwarding path of a flow and measure path latencies. There are three components in FlowTrace (as shown in Figure 5). The collector collects flow entries and builds virtual flow table passively without triggering any overhead in control plane. The virtual tables can be updated as soon as flow entries change. The path calculator will simulate forwarding actions of every switch based on aforementioned virtual flow table to calculate flow path. The latency monitor part inserted temporary flow rules along the decided path so long as network administers want to measure the latency of flows.

FlowTrace needs two rounds to measure latency. The probe packet which matched the default flow entry will be sent with a timestamp in its payload. Once it reaches the S_1 switch, it will trigger the responding forwarding action in the default flow entry and will return from the same egress port and path. The RTT can be achieved as T_1. In the second round, a new probe will be sent to S_2 switch through S_1. Then, it will get RTT as T_2. In this way, each hop delay can be achieved round by round.

3.2. Available Bandwidth Measurement. Bandwidth is another useful metric for network performance, especially the end-to-end ABw (available bandwidth) [24]. It is the maximum packet forwarding rate for a new flow to share with other flows that are running in the same path. Conducting bandwidth measurement in SDN can accelerate network management and improve network QoS. When controller is gathering information of the whole network, bandwidth can effectively diagnose and eliminate possible linkage problems, which guarantees that the network functions well [25]. Of all the measurement methods, we can classify them into two parts: one is ABw measurement methods for general condition and the other is designed for specific application conditions.

General Bandwidth Measurement. CSMABw [26] offers a very general and thoughtful method of calculating available bandwidth in software-defined networking. Before illustrating specific measurement details, we give the description of needed parameters in advance (shown as Table 1). The

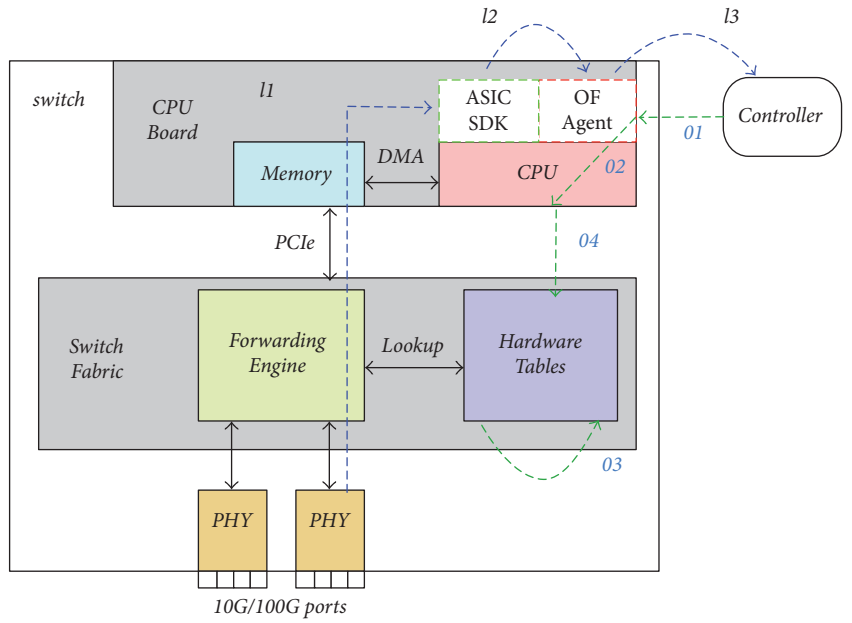

Figure 4: Contributing factors to inbound/outbound latency shown in schematic OpenFlow switch.

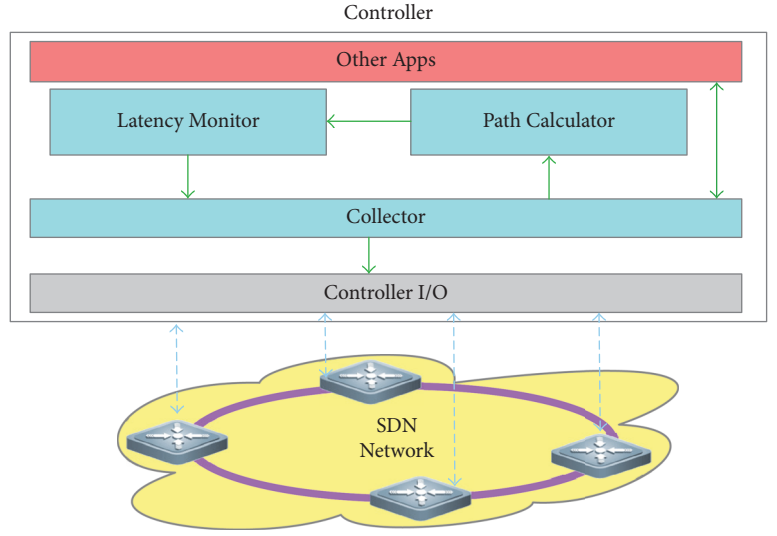

Figure 5: Contributing factors to inbound/outbound latency shown in schematic OpenFlow switch.

first step is to discover network topology graph $G(V, E)$. What needs to be emphasized is that the capacity c_i of every link is known in the network. In the next phase, controller would periodically poll OpenFlow switch counters with FlowStatsReq message to calculate current bandwidth load of a link $b_i(t)$ as

$$b_i(t) = \frac{n_i(t) - n_i(t-T)}{T}. \quad (1)$$

Then the available bandwidth a_i of any link in the network can be estimated as

$$a_i(t) = c_i - b_i. \quad (2)$$

Therefore, the available bandwidth on a given path P should be the minimal value of $a_i(t)$ as

$$\mathrm{ABW}_P = \min_{e_i \in P} a_i. \quad (3)$$

OpenNetMon [27] is a very classic and comprehensive method of implementing monitoring ABw of a path by periodically polling every switch along the distinct path in an active way. Based on OpenFlow protocol, the flow statistics of the first and last switches of a path can be acquired, such as the bytes and duration time, which are necessary for ABw estimation. What is more, to ensure more accurate

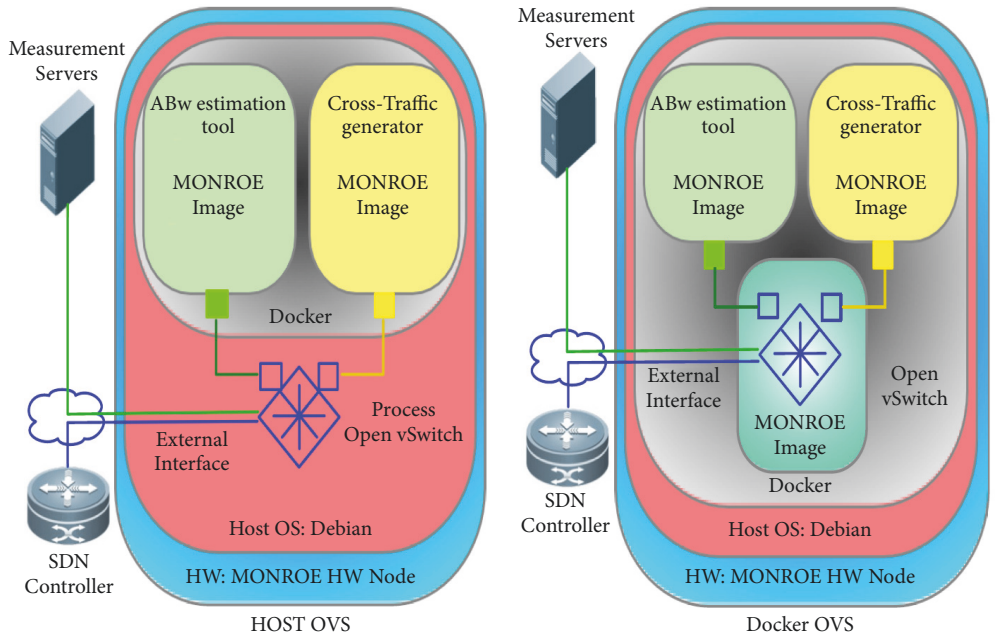

Figure 6: Designed structure for available bandwidth test. (a) is host side, (b) is docker container.

Table 1: Parameters description for available bandwidth measurement.

Parameters	Description
$G(V, E)$	The directed graph representation of the network topology
V	V stands for switch nodes in network
E	E stands for network links between nodes
T	The polling time for a measurement cycle
n_i	Counter values of a switch
c_i	The capacity of e_i
b_i	The current bandwidth load on e_i
a_i	The available bandwidth on e_i, $a_i = c_i - b_i$

information about current throughput per link or flow and avoid staleness of flow information, OpenNetMon uses an adaptive flow characterization. OpenNetMon will increase its sampling rate when new flow comes in and decrease sampling rate when environment is in static condition. As to implementation detail, OpenNetMon designs two module components which are forwarding component and monitoring component. The forwarding component gathers network topology information and configures paths by preinstalling flow rules on every necessary switch. Also, it will generate and forward new flows to the edge switch of a path. The monitoring component did only one thing: requesting flow counters at each time interval, which contain packet information, byte information, and time duration. Then, the delta of these counters can be used to compute ABw and other important network metrics.

Bandwidth Measurement with Specific Settings. SOMETIME [28] is an active method of providing an estimation of ABw by leveraging SDN features. It is designed and implemented in mobile broadband access networks. Due to diversity of devices and usage goals in mobile network, for example, mobile phones and mobile hotspot, a promising ABw measurement approach is needed to account for sharing of communication resources and breaking the performance limitation. In general, SOMETIME can take advantage of OpenFlow protocol to easily isolate targeted flow from other network traffic generated or received by other terminal devices through centralized controller. The detailed structure is shown in Figure 6.

The first necessary step is setup of SDN-enabled testbed for ABw estimation by adding ABw estimation into the measurement metric. Then SDN controller evaluates and mitigates the interference of local traffic and measurement traffic. Also, SDN controller will add a control interface for network operators. Although the presence of control message in the whole testing environment will cause additional overhead and may affect both the measurement flow and local traffic, the estimated ABw result is indeed more accurate in addition to the simplicity and convenience in SDN measurement.

As many cloud providers have been devoted to offering overprovisioning bandwidth resources in order to avoid network performance degradation and guarantee QoS for cloud tenant, MAPLE-Scheduler [29] pointed at improving tenant applications QoS by proposing an empirical estimation technique of EB (effective bandwidth) to assess whether flows need to be rerouted so as to meet the QoS requirements. In MAPLE, network is firstly divided into edge part and core part. Each part has different approaches to conduct EB estimation (as shown in Figure 7).

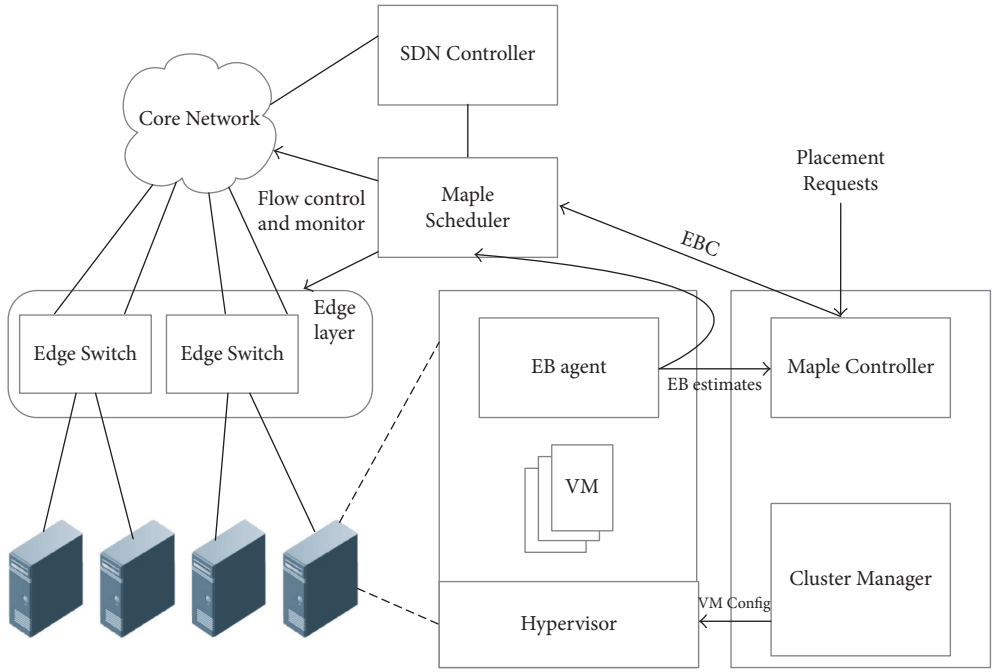

Figure 7: The architecture of MAPLE system.

In Figure 7, we can see that MAPLE only needs to deploy EB agent on edge switches to capture flow statistics which are the first access point of the network. Therefore, the obtained EB estimation can be more accurate. The detailed formulation is as follows:

$$\mathrm{EBC}_i = \frac{R_{\mathrm{eff},i}}{\mathrm{mean}_i}. \tag{4}$$

$R_{\mathrm{eff},i}$ represents the EB of some given link i and mean_i is the current mean throughput of the link. Once we obtained EBC, the residual bandwidth can be calculated as shown in the following two equations:

$$\begin{aligned} \mathrm{residual\_BW}_{\mathrm{sdge}} &= \mathrm{link\_capacity} - \mathrm{effective\_BW} \\ \mathrm{residual\_BW}_{\mathrm{core}} &= \mathrm{link\_capacity} - \widehat{\mathrm{EBC}} \times \mathrm{mean}. \end{aligned} \tag{5}$$

When MAPLE acquired EB and residual bandwidth, it should run its flow schedule algorithm to reroute flows in the network to improve QoS for their tenants.

4. Topology Measurement in SDN

Network topology is a visualized depict for the general state of network; it translates the physical link among all network nodes, which is a fundamental and core task in every network [30, 31]. Measuring and updating topology play a crucial role in providing necessary network functions such as routing, QoS, network management, and malfunction detection and troubleshooting. For example, to secure network and conduct some precautions for network attacks, network topology is one of the most important ones due to its significant role in network management [32]. Tracing packet trajectory through network is one of a useful and convenient way for verifying packet forwarding in data plane and guaranteeing network safety, which means accurate and real-time topology measurement is essential [33]. Therefore, in order to realize aforementioned functions in centralized controller, it needs to have up-to-date information about the state of the network in real-time, which is topology discovery.

Topology discovery is a crucial service at the control plane and will underpin the logically centralized control and management in SDN. In this section, we will focus on intradomain and interdomain topology discovery techniques applied in SDN networks.

4.1. Intradomain Topology Discovery Based on OpenFlow. OpenFlow Discovery Protocol (OFDP) is a reliable and widely used topology discovery method in SDN. It leverages LLDP protocol to discover linkage hop by hop. The whole procedure can be simplified in the sense that all switch nodes will establish TLS with controller by handshake session in the first place. During the session, control and switch exchange vital information and preinstall flow rules so that LLDP packet will finally return back to controller to finish topology information retrieving [34]. Finally, controller will build topology structure of the whole network. Detailed information is component of three aspects.

(1) Initial Stage. Controller obtains crucial information of every node in network. Switches establish TLS session with controller with its own IP address and TCP port numbers. Controller will retrieve active port number and corresponding MAC address of switches and assign unique ID for every switch. The map of ID and its corresponding port information will be stalled in local memories.

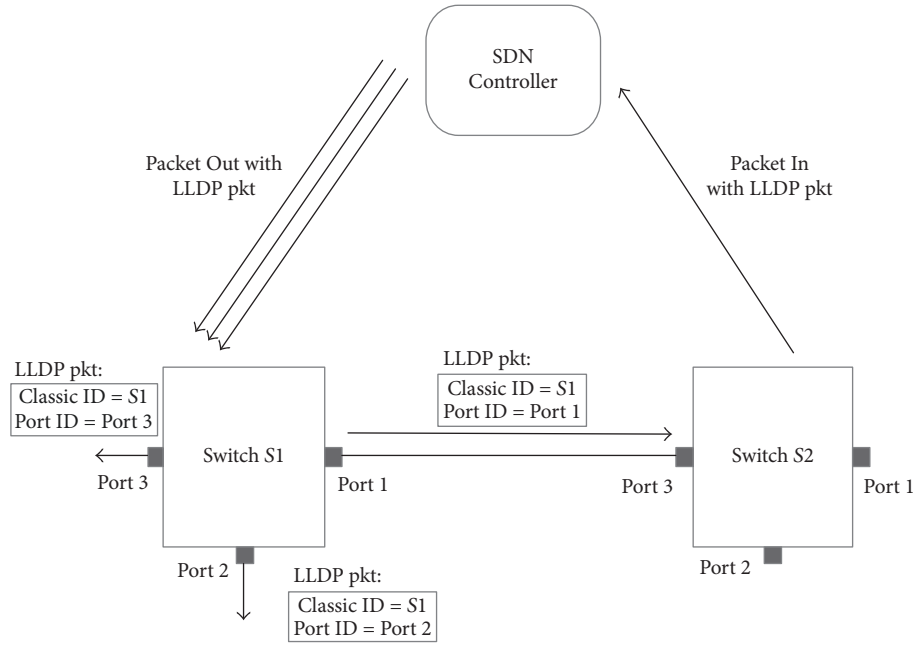

Figure 8: OFDP topology discovery procedure.

(2) Topology Discovery Stage. Controller sent LLDP packet to every active switch port which contains important information like switch ID, port number, and so on. Once arrived, switch transfers the packet to its neighbor node. Due to the preinstalled flow rules, all LLDP packets will be transferred to controller through Packet_In message. Thus, controller resolves received packets, retrieving switch ID and port number and updating mapping table to finish underlying network structure discovery. The whole procedure is shown in Figure 8.

(3) Topology Updating Stage. When network state changes (e.g., link interruption, port state change), this will trigger a synchronization in topology update. Controller will conduct aforementioned process again to obtain latest network state.

OFDP [35] is well designed and easy to implement, but the time cost, computation resource consumption, and introduced control overhead are relatively high. So, in OFDP *v*2 [36], these problems have been improved by reducing controller sending packet numbers which is limiting sent LLDP packet number for every switch to one. When switches received LLDP packet, these packets will be duplicated and modified, adding corresponding MAC address of transferring port number. These duplicated packets then are transferred to all available ports in a switch.

Experimental result proves that, after applying OFDP *v*2, 45% control overhead and 40% CPU resources have been saved compared with the original version. But, in the long run, the limitation of OFDP still exists. When updating the whole network structure, it is inevitable to consume a large amount of computing resources, cost too much time, and squeeze controllers performance with heavy burden. Network QoS can hardly be guaranteed on the one hand and scalability and performance of SDN networks are affected on the other [37]. This calls for a new topology discovery protocol.

SD-TDP [38] is a novel design of discovering network topology. The core idea is to alleviate controllers burden by breaking the whole into parts. The whole network structure is divided into several units, and every unit has an authorized father node (FN) and the rest of the nodes are active nodes (AN). Therefore, a hierarchical ControllerFNAN manage mode has been established to discover topology effectively within a short period of time. As shown in Figure 9, the procedure can be simplified into two parts.

(1) Initial Part. Each AN in the network maintains deactivated state, waiting for a TDP-Request message from controller or neighboring FN. Once a TDP-Request message is received, node status will be changed according to sender state value in the message (i.e., Father state or Children state). Consequently, the hierarchical structure is established. According to the paper, the switches will change their state after receiving the first TDP-Request message, only if they are at standby state.

(2) Discovery Part. During the communication between FN and AN, information is exchanged through TDP-Request messages. AN also transfer messages to neighbor node through other active ports; finally, FN will receive all TDP-Request messages of every AN in the unit. Controller will retrieve the topology structure of all FN and formulate a whole network topology.

Still, some specifications need to be clear. Once network topology changes and a FN has no AN connection, it automatically changes its state to AN. Also, SD-TDP mitigate controllers burden and improve its performance and cut

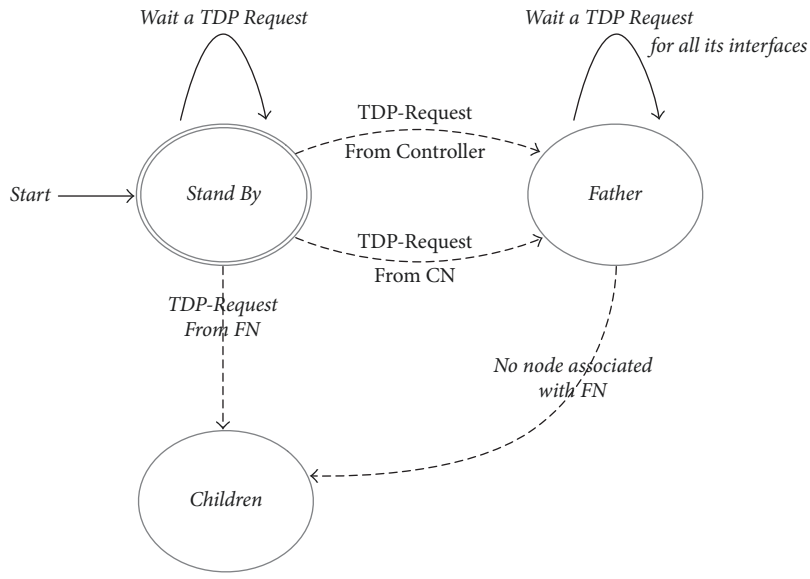

Figure 9: SD-TDP topology discovery procedure.

down time cost and computation resources consumption as well as error rate.

NetMagic [39] is a novel SDN-enabled platform for measuring network topology and scheduling flow routing. It needs to be reprogrammed with hardware language which makes the workflow of processing NMAC and non-NMAC packets in NetMagic more complex. The basic function of NetMagic is to actively carry out network topology detection. The procedure is as follows: (1) The controller generates probe packets to all NetMagics in the network. Once the NetMagic received these probe packets from console port, each NetMagic will copy the packet, insert its device ID into the data field, and broadcast the modified packet to all its neighbors through every active normal port. (2) Once NetMagic received packets from its neighbors, it parses the in-port and device ID of the packet and stores it in the RAM. (3) Controller will leverage read request of NMAC protocol and collect ⟨in-port, NetMagic ID⟩ pairs stored in every NetMagic RAM and build the entire information of network topology. There are two things to be mentioned, NetMagic uses on-board RAM to store history forward table and actively keep them updated. Also, the headers of packets will all be processed by a hash function, which dramatically reduced the use of RAM space. Figure 10 illustrates the whole workflow of topology detection.

4.2. Interdomain Topology Discovery in Large-Scale SDN Networks. Intradomain topology is always a heated and important issue when talking about network management and maintenance. However, as the size of network has been enlarged, which brings a lot of pressure for controller scalability, multidomain is necessary to solve this problem. Hence, interdomain topology discovery technology should be supported in current controllers. For example, administrators cannot know the actual positions of network hosts and switches in other domains from the GUI; when network malfunction happens, controller can only know problems inside the domain, which takes more time and resources to solve problems (hypothetical scenario illustration in Figure 11). When making routing decisions, controller cannot achieve optimized routing path. So, interdomain communication should raise our concerns.

The method in [40] installed a third-party module ENVI and supports a communication mechanism between NOX and Floodlight controller. To be specific, the first part is the modification of NOX for host display. If a host link connects with an interdomain port, it is recognized as an interdomain host link. If it is interdomain host link, NOX will send every host link message to ENVI such as node ID, link type, and interdomain field. If NOX received interdomain host link message from other domain, the link information will be stored into a data structure named interdomain map. The second part is Floodlight extension for support interdomain mechanism. Floodlight has to create a communication channel with ENVI in the first place. Then, two data structures will be added in Floodlight controller to support the communication, which are interDomainNodeMap and interDomainLinkMap [41]. So, new data structure can be stored and processed among NOX, Floodlight, and ENVI. The third part is interdomain communication between NOX and Floodlight. Here, both domains can exchange information by LLDP packet. NOX needs to add a new optional field to store switch ID. Floodlight needs to add a new field to record NOX IP field.

In this paper, NOX and Floodlight build communication mechanism through third-party modules and function well based on experimental results. This works well in simple multidomain networks. However, the big complexity, low communication efficiency, and lack of high identification accuracy of this procedure still need improvement.

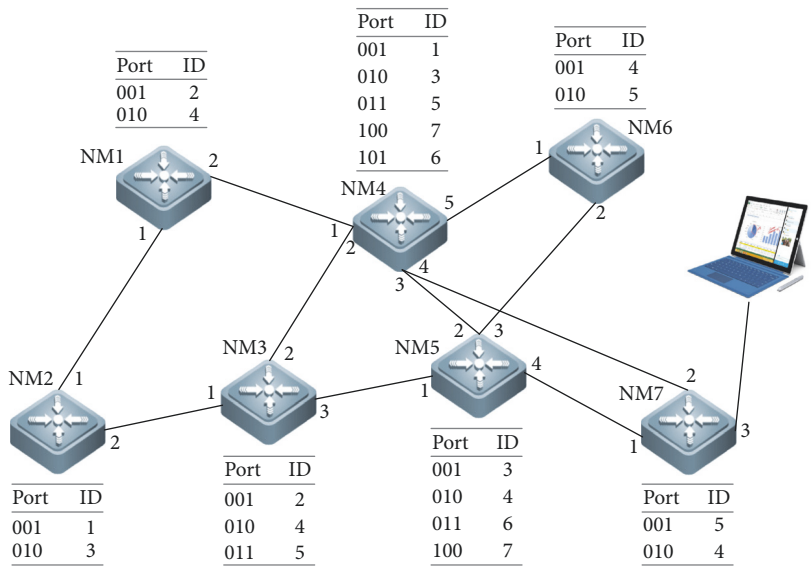

Figure 10: Workflow of topology detection and corresponding trajectory data in NetMagic.

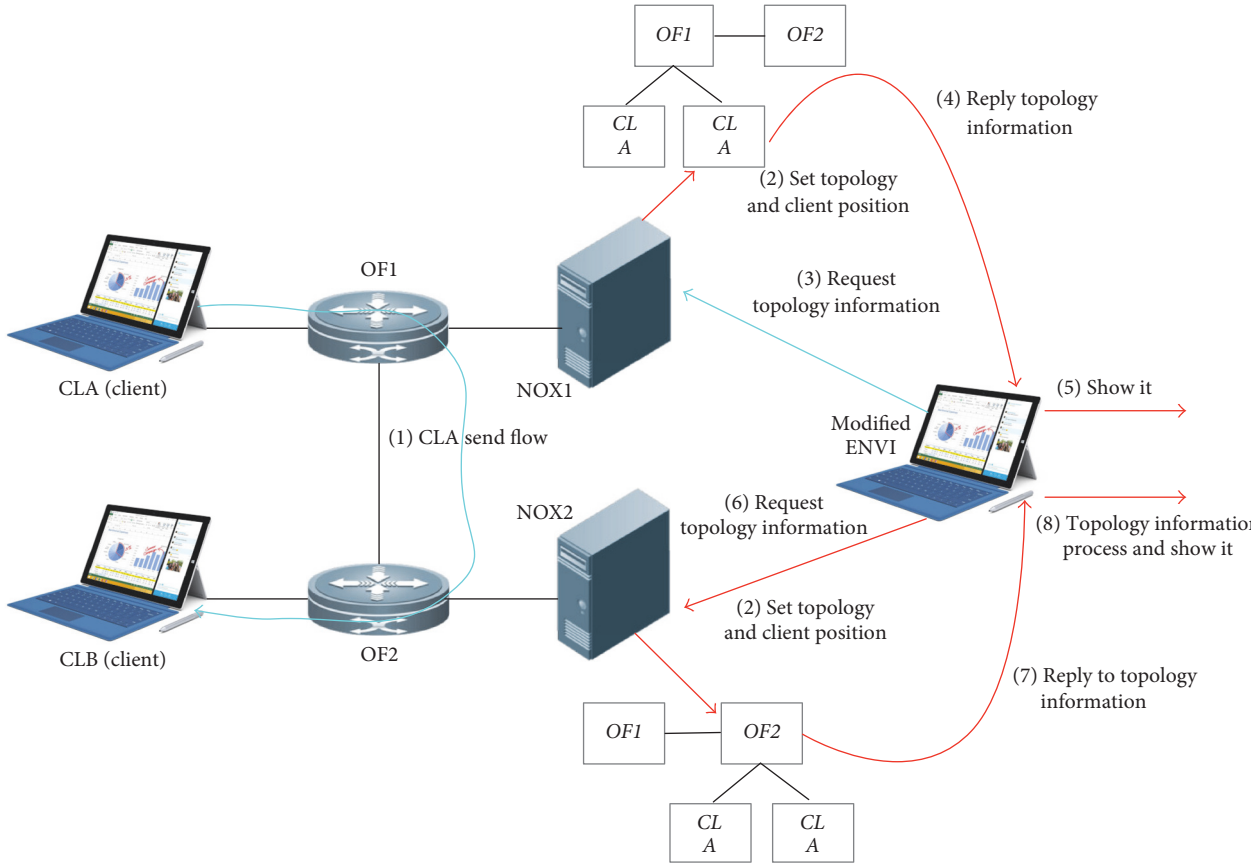

Figure 11: Hypothetical scenario of multidomain communication condition.

5. Future Directions of Security-Aware Measurement in SDN

According to the above background analysis and technical survey, we figure out a great improvement in security-based SDN in the literature [42]. Although SDN has benefits from the network security perspective, we still find several noticeable weak points which bring new attack vectors. These should be the future research issues.

First, due to the excessive scale of network, it is impossible for one controller to cover all the network services and handle all outbursts of network malfunctions. This comes

to coordination among controllers, which should be an interesting and important research direction. Once a network contains multiple areas, the difficulty of detecting network threats by SDN measurement technologies wound grow exponentially [43]. Thus, researchers should pop out new ingenious approaches suitable for solving this problem.

Second, most current measurement technologies have their own scope of application. And judging standard for these acquired metrics is vague during the whole process. This calls for a comprehensive SDN security framework that can handle as many security threats as possible [44]. For example, every security case should set a triggering condition, the SDN measurement framework should be integrated with many measurement modules to offer essential network metrics. By analyzing achieved metrics and matching pre-set threshold, we can detect network threats in near real-time.

Third, every measurement approach has its apparent advantages to catch one's eyes, but it is always accompanied with disadvantages [45]. Take timeliness as an example; real-time measurement is doomed for active measure pattern and consumes more computation and storage resources. However, resources and performance are permanent rivals for researchers. Therefore, to take everything into consideration and come up with a balancing method should be another research direction, too.

6. Conclusions

With the separation of control and data plane and network programmability, SDN appears to be the most fascinating evolution structure for future network. In spite of its impressive benefits, SDN still encounters many challenges, especially when it comes to network security problem. Under this condition, we tried to detect network threats by analyzing core network metrics, which leads us to survey security-aware network measurement in SDN.

In this paper, we briefly reviewed the SDN structure and OpenFlow protocol characteristics in the first place. After which, we analyze current security-based network situations and believe SDN network management technologies must be a reliable and efficient way to deal with those challenges. Then we introduced current network security condition based on SDN technologies and decided to pick link latency, available bandwidth, and network topology as key entries to detect and resist network threats. Afterwards, we separately surveyed several latest measurement approaches of these metrics and conducted a little comparison. Furthermore, we discussed possible future research directions for security in SDN. We first discussed the extension of SDN measurement technologies to multiple controllers case. Then, we suggest to build up a comprehensive security framework to cater for increasing network attacks. Last but not least, we focus on promoting a more balancing measurement method instead of evident shortage. Thus, there is much work to do before these visions are realized.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants nos. 61379145, 61501482, and 61762033 and the Joint Funds of CETC under Grant no. 20166141B08020101.

References

[1] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: detecting security attacks in software-defined networks," in *Proceedings of the Network and Distributed System Security Symposium*, 2015.

[2] S. Jain, A. Kumar, S. Mandal et al., "B4: experience with a globally-deployed software defined WAN," in *Proceedings of the ACM SIGCOMM 2013 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM 2013*, vol. 43, pp. 3–14, August 2013.

[3] S. Shin, L. Xu, S. Hong, and G. Gu, "Enhancing network security through software defined networking (SDN)," in *Proceedings of the 25th International Conference on Computer Communications and Networks, ICCCN 2016*, August 2016.

[4] W. Li, W. Meng, and L. F. Kwok, "A survey on OpenFlow-based Software Defined Networks: Security challenges and countermeasures," *Journal of Network and Computer Applications*, vol. 68, pp. 126–139, 2016.

[5] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN security: a survey," in *Proceedings of the 2013 Workshop on Software Defined Networks for Future Networks and Services, SDN4FNS 2013*, Trento, Italy, November 2013.

[6] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, "Towards a secure controller platform for OpenFlow applications," in *Proceedings of the 2013 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN 2013*, pp. 171-172, August 2013.

[7] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. B. Godfrey, and S. T. King, "Debugging the data plane with anteater," in *Proceedings of the ACM SIGCOMM Conference (SIGCOMM '11)*, pp. 290–301, Toronto, Canada, August 2011.

[8] W. Sun, Z. Cai, Y. Li, F. Liu, S. Fang, and G. Wang, "Security and privacy in the medical internet of things," *Security and Communication Networks*, vol. 2018, 2018.

[9] L. Schehlmann, S. Abt, and H. Baier, "Blessing or curse? Revisiting security aspects of software-defined networking," in *Proceedings of the 10th International Conference on Network and Service Management (CNSM '14)*, pp. 382–387, Rio de Janeiro, Brazil, November 2014.

[10] C. Jeong, T. Ha, J. Narantuya, H. Lim, and J. Kim, "Scalable network intrusion detection on virtual SDN environment," in *Proceedings of the 2014 3rd IEEE International Conference on Cloud Networking, CloudNet 2014*, pp. 264-265, Luxembourg, October 2014.

[11] S. T. Ali, V. Sivaraman, A. Radford, and S. Jha, "A survey of securing networks using software defined networking," *IEEE Transactions on Reliability*, vol. 64, no. 3, pp. 1086–1097, 2015.

[12] M. F. Akbas, E. Karaarslan, and C. Güngör, "A preliminary survey on the security of software-defined networks," in *Proceedings of the in 3rd International Conference on Advanced Technology & Sciences (ICAT '16)*, 2016.

[13] J. Tang, A. Liu, M. Zhao, and T. Wang, "An Aggregate Signature Based Trust Routing for Data Gathering in Sensor Networks,"

Security and Communication Networks, vol. 2018, Article ID 6328504, 30 pages, 2018.

[14] C. Yu Hunag, T. Min Chi, C. Yao Ting, C. Yu Chieh, and C. Yan Ren, "A novel design for future on-demand service and security," in *Proceedings of the 2010 IEEE 12th International Conference on Communication Technology, ICCT'2010*, pp. 385–388, November 2010.

[15] C. Lee, C. Park, K. Jang, S. Moon, and D. Han, "DX: Latency-Based Congestion Control for Datacenters," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 335–348, 2017.

[16] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: trading a little bandwidth for ultra-low latency in the data center," in *Proceedings of the in Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pp. 19-19, 2012.

[17] C. Lee, C. Park, K. Jang, S. B. Moon, and D. Han, "Accurate latency-based congestion feedback for datacenters," in *Proceedings of the in USENIX Annual Technical Conference*, pp. 403–415, 2015.

[18] B. Vamanan, J. Hasan, and T. N. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, ACM SIGCOMM 2012*, pp. 115–126, Finland, August 2012.

[19] C. Yu, C. Lumezanu, A. Sharma, Q. Xu, G. Jiang, and H. V. Madhyastha, "Software-defined latency monitoringin data center networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 8995, pp. 360–372, 2015.

[20] M. Wu, Y. Wu, X. Liu, M. Ma, A. Liu, and M. Zhao, "Learning-based synchronous approach from forwarding nodes to reduce the delay for Industrial Internet of Things," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, 2018.

[21] T. Mizrahi and Y. Moses, "The case for data plane timestamping in SDN," in *Proceedings of the 35th IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2016*, pp. 856–861, USA, April 2016.

[22] K. He, J. Khalid, A. Gember-Jacobson et al., "Measuring control plane latency in SDN-enabled switches," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, SOSR 2015*, June 2015.

[23] S. Wang, J. Zhang, T. Huang, J. Liu, Y.-J. Liu, and F. R. Yu, "Flow-Trace: measuring round-trip time and tracing path in software-defined networking with low communication overhead," *Frontiers of Information Technology and Electronic Engineering*, vol. 18, no. 2, pp. 206–219, 2017.

[24] P. Megyesi, A. Botta, G. Aceto, A. Pescapè, and S. Molnár, "Available bandwidth measurement in software defined networks," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing, SAC 2016*, pp. 651–657, April 2016.

[25] L. Popa, P. Yalagandula, S. Banerjee, J. C. Mogul, Y. Turner, and J. R. Santos, "ElasticSwitch: Practical work-conserving bandwidth guarantees for cloud computing," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, ACM SIGCOMM 2013*, pp. 351–362, August 2013.

[26] P. Megyesi, A. Botta, G. Aceto, A. Pescapé, and S. Molnár, "Challenges and solution for measuring available bandwidth in software defined networks," *Computer Communications*, vol. 99, pp. 48–61, 2017.

[27] N. L. M. Van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: network monitoring in openflow software-defined networks," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World, NOMS 2014*, May 2014.

[28] G. Aceto, V. Persico, A. Pescapé, and G. Ventre, "SOMETIME: Software defined network-basec Available Bandwidth measurement in MONROE," in *Proceedings of the 1st Network Traffic Measurement and Analysis Conference, TMA 2017*, Ireland, June 2017.

[29] R. Wang, S. Mangiante, A. Davy, L. Shi, and B. Jennings, "QoS-aware multipathing in datacenters using effective bandwidth estimation and SDN," in *Proceedings of the 12th International Conference on Network and Service Management, CNSM 2016 and Workshops, 3rd International Workshop on Management of SDN and NFV, ManSDN/NFV 2016 and International Workshop on Green ICT and Smart Networking, GISN 2016*, pp. 342–347, Canada, November 2016.

[30] K. Bakshi, "Considerations for Software Defined Networking (SDN): Approaches and use cases," in *Proceedings of the 2013 IEEE Aerospace Conference, AERO 2013*, USA, March 2013.

[31] R. A. Alhanani and J. Abouchabaka, "An overview of different techniques and algorithms for network topology discovery," in *Proceedings of the 2nd World Conference on Complex Systems, WCCS 2014*, pp. 530–535, Morocco, November 2014.

[32] F. Liu and T. Li, "A clustering-anonymity privacy-preserving method for wearable iot devices," *Security and Communication Networks*, vol. 2018, 2018.

[33] G. Tarnaras, E. Haleplidis, and S. Denazis, "SDN and ForCES based optimal network topology discovery," in *Proceedings of the 1st IEEE Conference on Network Softwarization, NETSOFT 2015*, UK, April 2015.

[34] M. Huang, A. Liu, T. Wang, and C. Huang, "Green data gathering under delay differentiated services constraint for internet of things," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 9715428, 2018.

[35] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient topology discovery in software defined networks," in *Proceedings of the 8th International Conference on Signal Processing and Communication Systems, ICSPCS 2014*, Australia, December 2014.

[36] S. Khan, A. Gani, A. W. Abdul Wahab, M. Guizani, and M. K. Khan, "Topology Discovery in Software Defined Networks: Threats, Taxonomy, and State-of-the-Art," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 303–324, 2017.

[37] Y. Li, Z. Cai, and H. Xu, "LLMP: Exploiting LLDP for Latency Measurement in Software-Defined Data Center Networks," *Journal of Computer Science and Technology*, vol. 33, no. 2, pp. 277–285, 2018.

[38] L. Ochoa-Aday, C. Cervelló-Pastor, and A. Fernández-Fernández, "Discovering the network topology: an efficient approach for SDN," *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 5, no. 2, p. 101, 2016.

[39] M. Dai, G. Cheng, and Y. Wang, "Detecting network topology and packet trajectory with SDN-enabled FPGA Platform," in *Proceedings of the 11th International Conference on Future Internet Technologies, CFI 2016*, pp. 7–13, June 2016.

[40] W.-Y. Huang, T.-Y. Chou, J.-W. Hu, and T.-L. Liu, "Automatical end to end topology discovery and flow viewer on SDN," in *Proceedings of the 28th IEEE International Conference on*

Advanced Information Networking and Applications Workshops, IEEE WAINA 2014, pp. 910–915, Canada, May 2014.

[41] J. Xia, Z. Cai, G. Hu, and M. Xu, "An active defense solution for arp spoofing in openflow network," *Chinese Journal of Electronics*, 2018.

[42] Z. Hu, M. Wang, X. Yan, Y. Yin, and Z. Luo, "A comprehensive security architecture for SDN," in *Proceedings of the 2015 18th International Conference on Intelligence in Next Generation Networks, ICIN 2015*, pp. 30–37, IEEE, Paris, France, February 2015.

[43] C. C. Lamb and G. L. Heileman, "Towards robust trust in software defined networks," in *Proceedings of the 2014 IEEE Globecom Workshops, GC Wkshps 2014*, pp. 166–171, USA, December 2014.

[44] S. Bian, P. Zhang, and Z. Yan, "A survey on software-defined networking security," in *Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications*, Xi'an, China, June 2016.

[45] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient OpenFlow-based networking," in *Proceedings of the 2012 IEEE Network Operations and Management Symposium, NOMS 2012*, pp. 933–939, USA, April 2012.