Network Monitoring in Software-Defined Networking: A Review

Pang-Wei Tsai, Chun-Wei Tsai, Chia-Wei Hsu, and Chu-Sing Yang

Abstract—**Monitoring is an important concept in network management as it helps network operators to determine the behavior of a network and the status of its components. Traffic engineering, quality of service, and anomaly detection also depend on monitoring for decision making. Software-defined networking (SDN) is becoming increasingly popular for network provision and management tasks. This paper surveys the tasks and challenges associated with SDN, providing an overview of SDN monitoring developments. Several design concepts, research directions, and open issues are also discussed.**

Index Terms—**Measurement, monitoring, OpenFlow, software-defined networking (SDN).**

I. Introduction

THE purpose of network monitoring is to support proper management operations [1]. Monitoring provides a view of the network status and illustrates network behavior, which is a basis for further management operations such as traffic engineering [2], quality of service (QoS) [3], and anomaly detection [4]. In computer networks, the operation model [5] is a stackable architecture with different layers, linking numerous hosts for data exchange. To satisfy various management purposes, network monitoring helps network operators to obtain operation and usage statistics.

As the Internet continues to grow at a fast pace, more and more network applications are being leveraged by new technologies to improve our daily life. At present, the traditional network architecture cannot meet all the requirements of new applications. For instance, content delivery services usually require flexible and adaptive controls to ensure high performance in terms of global and regional network transmission. Live broadcasting services may suddenly require a large bandwidth capacity to serve their audiences. After the broadcast has ended, the network resources are no longer required. According to the application requirements, network systems must be intelligent for adaptation and optimization. For this issue, Pras *et al.* [6] listed the existing problems of current network structures. They described the weak scalability of ordinary monitoring methods and the lack of efficiency in gathering monitoring information.

Software-defined networking (SDN) has been proposed as a means of gradually eliminating network problems via adaptation and flexibility [7]. SDN improves network programmability and provides a global view of the entire network by adding separated control and data planes, which distinguish data transmission from control operations. In related studies, Sezer *et al.* [8] discussed implementation issues of SDN, such as the tradeoff between data processing, higher costs of communication, and security issues in the system control. Nunes *et al.* [9] discussed alternatives and possible implementations of SDN, whereas Yassine *et al.* [10] surveyed issues related to monitoring and discussed the challenges of traffic measurement within SDN. They pointed out that balancing resources and providing exact real-time measurements via SDN are difficult tasks. However, owing to the changes in network architectures, traditional optimization methods such as balancing algorithms and traffic matrices may struggle to implement traffic engineering in SDN. Shu *et al.* [11] illustrated this issue and created a research framework for enabling traffic engineering in SDN. Their approach consists of two parts: traffic measurement and traffic management. They also outlined an adaptive optimization method expected to reduce the energy consumption of network maintenance.

To provide perspective, our paper surveys developments in research on SDN monitoring. Fig. 1 outlines the organization of this paper. Section II briefly reviews the background to network monitoring and compares established network systems

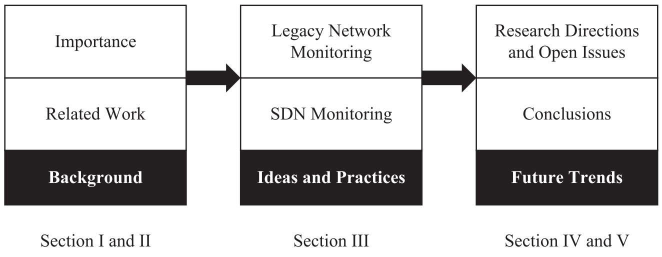


Fig. 1. Steps for interpreting the review in this paper.

Manuscript received March 24, 2017; revised November 7, 2017 and December 16, 2017; accepted December 29, 2017. This work was supported in part by the Ministry of Science and Technology of Taiwan under Contract MOST105-2221-E-005-091, Contract MOST106-2221-E-006-025, Contract MOST106-3114-E-006-003, Contract MOST106-2221-E-005-094, and Contract MOST106-3114-E-005-001, and in part by the TWAREN SDN Research Team and OF@TEIN Community under Grant Asi@Connect-17-094. *(Corresponding author: Chun-Wei Tsai.)*

P.-W. Tsai and C.-S. Yang are with the Institute of Computer and Communication Engineering, Department of Electrical Engineering, National Cheng Kung University, Tainan 701, Taiwan (e-mail: pwtsai@ee.ncku.edu.tw; csyang@ee.ncku.edu.tw).

C.-W. Tsai is with the Department of Computer Science and Engineering, National Chung Hsing University, Taichung 402, Taiwan (e-mail: cwtsai0807@gmail.com).

C.-W. Hsu was with the Institute of Computer and Communication Engineering, Department of Electrical Engineering, National Cheng Kung University, Tainan 701, Taiwan. She is now with the Taiwan Semiconductor Manufacturing Company Limited, Hsinchu 300-78, Taiwan (e-mail: winnie148636@gmail.com).

Digital Object Identifier 10.1109/JSYST.2018.2798060

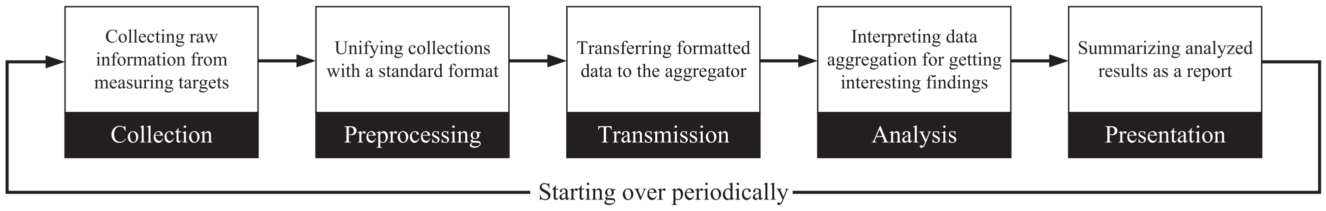

Collecting raw information from measuring targets
Collection
Unifying collections with a standard format
Preprocessing
Transferring formatted data to the aggregator
Transmission
Interpreting data aggregation for getting interesting findings
Analysis
Summarizing analyzed results as a report
Presentation
Starting over periodically

Fig. 2. Reference scheme of operation phases in network monitoring.

with those equipped with SDN. Section III introduces the concept and practice of SDN monitoring, explaining the algorithms, mechanisms, and operation schemes. Section IV collects open issues for research. Finally, the conclusions and research suggestions of SDN monitoring are given in Section V.

II. Background and Related Work

To introduce the process of network monitoring, this section briefly reviews some basic notions. To differentiate currently used network systems from those equipped with SDN, we refer to the former as legacy networks. The different monitoring aspects and implementations between legacy and SDN networks are also compared and discussed.

A. *Basic Notions of Network Monitoring*

The procedure of network monitoring is classified in [12]. The first step is measurement, followed by the aggregation and preprocessing of raw datasets. The datasets are then subjected to various analytics for investigation. Finally, the data can be visualized to enable network operators to realize the status. To interpret the detailed process of network monitoring, Williamson [13] reviewed the fundamental properties of modern networks. Lee *et al.* [14] also analyzed monitoring technologies and identified possible problems. Generally, network monitoring can be roughly classified into data measurement and data processing. Data measurement includes obtaining and preserving data. Based on previous research, we further divide the measurement operation into collection, preprocessing, and transmission. Data processing is responsible for organizing the data as readable information, and we split this into analysis and presentation. The processing scheme is shown in Fig. 2.

1) *Collection:* In this phase, there are three basic considerations: means, target, and frequency. The means refers to how the data are to be collected. The target refers to the devices to be observed, and the frequency denotes the time period before updating information. Furthermore, the measurement methods can be divided [15] into active and passive. In active measurement, agents generate probe instructions to perform a network feature evaluation. The measurement results can provide network behavior information instantly, although there is also a chance of increasing the system load if this operation is performed frequently. In contrast, passive measurement does not add artificial actions. The agents measure network statistics indirectly by receiving output data from network components.
2) *Preprocessing:* In this phase, the raw data are aggregated and turned into some statistical format. The data processing mechanism is helpful in itemizing and tracking the measurement results. For example, the management information base (MIB) [16] is a hierarchical structure that provides such a format to handle the collected information. Each MIB is addressed or identified by a unique object identifier in the MIB hierarchy. In this way, the data collected from each device can be separately itemized.
3) *Transmission:* This phase is responsible for carrying itemized data to the analytics station. For example, the simple network management protocol (SNMP) is a typical protocol used to exchange messages in transmission. It provides a data delivery interaction between agents and the station. Syslog [17] is another example of efficient system message transmission, which uses the connectionless method for log delivery.
4) *Analysis:* This phase generates statistics and identifies particular events. Some methods perform traffic classification based on the payload or host behavior, whereas other methods examine communication patterns [18]. The analysis results provide network status information to traffic engineering and fault management applications.
5) *Presentation:* This phase is used to export the analysis results. For example, MRTG [19] and RRDtool [20] provide data visualization via traffic graphs. Graphing solutions such as Cacti [21] are commonly used to present both long- and short-term traffic statistics. Furthermore, presenting the status of the network topology is important for monitoring the network. The network reachability can be pictured by analyzing routing tables and protocols [22].

B. *Differences Between Legacy Network and SDN Monitoring*

With SDN monitoring, a network can perform different monitoring tasks at multiple spatial and temporal scales [23]. At the spatial scale, SDN can divide larger traffic and aggregate smaller traffic based on address prefixes and port numbers. At the temporal scale, statistics can be collected in different periods of time. Semicentralized agents can also share the overheads. Table I compares the phases of a legacy network with those of SDN monitoring. The details are as follows.

1) *Collection:* In a legacy network, developing agents to collect measurement data from heterogeneous devices is trivial. The agents pull data from devices periodically and store them for further investigation. To improve this, according to a designed scheduler with a global view,

TABLE I
COMPARISON OF LEGACY NETWORK AND SDN IN EACH MONITORING PHASE

Phase	Legacy network	SDN	Comparison
Collection	Managed objects are equipped with agent functions	SDN devices can record traffic information using collecting functions	SDN devices (such as OpenFlow switch) can record traffic information. Additionally, forwarding devices are combined with customized software to record information, offering more flexibility than agent functions.
Preprocessing	Using a standard format for filtering	Using flexible ways to aggregate and examine collection	SDN methods can directly read information kept by forwarding devices rather than simplifying the mathematical and statistical assumptions used in the past
Transmission	Using transmission protocols	Supporting protocols and APIs	SDN replaces legacy network control methods with southbound APIs and SDN protocols
Analysis	Analysis function	Supporting extended module through extensions	The extended modules in SDN are able to support analysis and adaptation functions to automatically adjust SDN devices. Legacy network operators can only manually change the network configuration after analyzing the network status.
Presentation	Visual data/reachability map	Interactive interface/global view	SDN can deploy an interactive interface for further operations using northbound APIs. Additionally, the traditional method of visualizing a network depends on discovery and analysis results, whereas the SDN controller offers an initial global view.

SDN is able to decide which devices should be observed and the precise polling time through the controller.

2) *Preprocessing:* In this phase, the filtering and itemization methods for SDN are similar to those in a legacy network. Because data flows may cross multiple SDN devices in the network, it is necessary to distinguish valid from invalid data. Decisions can be sent back to the collection agents to avoid recording duplicate data from neighborhood network devices.
3) *Transmission:* In a legacy network, standard protocols are commonly used for transmission (e.g., SNMP and syslog). However, in the SDN architecture, the southbound application programming interface (API) allows network designers to define suitable data structures for their requirements. For example, OpenFlow [24] is a practical instance of SDN. The OFPMT METER [25] component of OpenFlow is a built-in controller function for obtaining meter data from OpenFlow devices. Valid data from the agents will be sent for analytics through OpenFlow control communication.
4) *Analysis:* Most of the analytics developed for the legacy network status are also available for SDN and are integrated with the controller. For large-scale and distributed SDN systems, the logging information can be easily explored using the centralized control. It is also convenient for software developers to add interpreting mechanisms [26] for network status analysis.
5) *Presentation:* The existing visualizations can be easily integrated with the SDN system. Moreover, SDN can provide interactive interfaces to support application-level developments for presentation. By obtaining data through the northbound APIs, network behavior and anomalies can be immediately visualized.

C. Discussion

Comparing the legacy network with SDN, the former has advantages in terms of bespoke operations and survivability, whereas the latter offers better adaptation and flexibility in terms of network control. As the network continues to grow, the scalability increases the difficulty of monitoring. The latency and inconsistency of distributed network devices usually lead to inappropriate measurement results [27]. However, without efficient methods in the preprocessing and transmission phases, it is difficult to coordinate the log collection from various devices. Moreover, some new network innovations (e.g., cloud-based data centers and 5G mobile networks) have created more complex environments in terms of system operation. Several studies [28], [29] show that traditional monitoring methods may not be capable of keeping up with fast, on-demand, and dynamic changes in network behavior in such environments. Under this circumstance, using SDN to implement the monitoring mechanism is a possible improvement. In the SDN architecture, the controller can be used for global network monitoring. This centralized control in SDN makes it possible to perform more softwarized operations on the network system, including correlating traffic to specific forwarding destinations, identifying end-hosts, pinpointing critical nodes, showing flow statistics, and determining the topology of and tracking anomalous traffic. With the SDN approach, network monitoring operations can be turned into primitive functions embedded in the controller, utilizing software-based control methods to observe the whole SDN network.

In terms of implementation, there are still several problems in applying the SDN methodology for production network monitoring. As most of the global network is still using the traditional architecture, the first problem occurs at the boundary between the legacy network and SDN. The SDN controller cannot manage or visualize network devices without the support of the SDN protocol. Additionally, because SDN separates the logic of forwarding devices from the data plane, it has an extra communication requirement and increased delay time for centralized control communication. The monitoring tools should properly adjust the connection quality and enable bootstrapping communication to improve efficiency. Even SDN seems to be imperfect, despite its many benefits in developing network applications. For network designers, SDN brings a novel software-control architecture for ameliorating existing problems with legacy networks. From the viewpoint of network operators, SDN provides the possibility of optimally managing network resources via softwarized artificial programs. The open standard of SDN simplifies the network operations, increasing the transparency of exploring the network.

III. Research Developments in SDN Monitoring

According to the various monitoring phases, we have collected various developments in SDN monitoring in Table II. In the collection phase, the monitoring components are instructed to track the network status. In the preprocessing phase, the listed developments include flow header inspection, hashing, programming, and the traffic matrix (TM). In the transmission phase, there are two types of instances: SDN-original methods (e.g., controller polling and switch pushing) and traditional techniques (e.g., sampling and port mirroring). In the analysis phase, traffic statistics, anomaly detection, fault management, and traffic engineering must be considered. In the presentation phase, various developments examine an interactive interface, high-level language, and real-time presentation. The implementation varieties are explained in the following.

A. Collection Methods

The query schedule and time period adaptation are two important concepts in data collection. Narayana *et al.* [44] proposed a method using a query language to enable adaptive strategies for reducing the overheads of SDN monitoring. Moreover, MicroTE [30] instructs servers to track the network traffic through its interface, but there is only one server per rack for summarizing the host-to-host traffic. Mahout [31] focused on detecting elephant flows in networks by observing the socket buffers in end hosts as a proxy for flow measurement. HONE [32] uses an abstract layer to specify the required statistics, employing partitioning among the host agents and the controller for measurement. Living on the Edge [33] explores two-threaded monitoring inside the virtual switches: one thread for NetFlow [60] flow sampling and another for sFlow [61] packet sampling. Experiments using Living on the Edge show that NetFlow gives efficient and accurate measurements, whereas a badly configured sFlow can degrade the network throughput when porting legacy methods into an SDN environment.

B. Preprocessing Methods

The preprocessing phase in SDN is still responsible for gathering and pre-examining the collected data. Most of the developments focus on improving the collection performance and organizing the measurement agents. We categorize these methods into four groups and discuss them in the following.

1) Flow Header Inspection: In this area, different packet fields (e.g., source/destination IP addresses and port numbers) are used to specify the flow statistics. The hardware counters in the ternary content addressable memory (TCAM) entries play an important role in rapid data processing. For instance, Hamad *et al.* [34] proposed a mechanism for obtaining traffic statistics through the OpenFlow protocol and analyzed the impact of the querying frequency on the network load and information accuracy. Their results indicate that the developed method achieves high accuracy. Owing to the limited bandwidth and TCAM entries, the scalability of flow header inspection is limited in practice.

2) Hashing: This methodology extracts the sum of a packet (rather than storing the entire packet body) to provide a tradeoff between memory use and accuracy in data processing. A typical development, OpenSketch [35], uses a hash function to improve the performance of the preprocessing phase. OpenSketch has a three-stage pipeline architecture for storing data by integrating hash functions, classification, and a counting table. One characteristic of OpenSketch is that it makes switches using static random-access memory (SRAM) to store all the counters—this is cheaper and more energy efficient than TCAM. Currently, OpenSketch relies on modifying the commodity switches and has limited availability.

3) Programming: This mechanism can support the monitoring of specific objects with cost-efficient actions based on algorithms. For example, ProgME [36] creates a measurement architecture (flowset) by organizing the collected data. The flowset is specified by the descriptive Flowset Composition Language. Compared with per-flow information, the flowset offers better performance in tracking the flows. Moshref *et al.* [37] proposed an algorithm called Max-Cover HHH to find the critical nodes in the network topology. Their method collects statistics from these nodes for use in existing switch components. Furthermore, some high-level programming language designs (e.g., Frenetic [41] and Pyretic [42]) provide the functionality for monitoring specific traffic. In addition, NetAssay [43] is a preliminary design based on Pyretic that captures traffic with specific characteristics. It uses the domain name, autonomous system number, and user or device information to collect traffic, rather than the IP address or protocol number.

4) Traffic Matrix: The TM has played an important role in traffic analysis and policy-control for many years [62]. Extending TM to the SDN framework is a popular research topic. In recent developments, OpenTM [38] directly reads the statistics from each flow to obtain the TM. This approach is concerned with the accuracy of the collected data, rather than with efficient querying frequency, which may cause scalability problems. DCM [39] uses a novel two-stage filtering method to monitor per-flow information. It includes an admission component to filter flows that are not of interest and uses a Bloom filter to determine the corresponding monitoring actions. iSTAMP [40] separates TCAM entries into two parts: one for using the TM to optimally aggregate parts of the incoming flows and another for per-flow monitoring of the informative flows. Although iSTAMP seems to give a good allocation between resources and accuracy, it ignores the flow aggregation constraints.

C. Transmission Methods

The transmission phase in SDN is responsible for sending data plane statistics from devices to the controller. However, by integrating with SDN, several developments used in legacy

TABLE II
INSTANCES OF RESEARCH DEVELOPMENTS IN SDN MONITORING

Phase	Project/development	Remarks
Collection	MicroTE [30]	MicroTE tracks network traffic at datacenter servers and leverages the existence of short-term predictable traffic to mitigate unpredictable congestion
	Mahout [31]	Mahout uses the shim layer in the end host to monitor socket buffers for elephant flow detection
	HONE [32]	HONE provides a query language for ordering data collection across multiple hosts, which can reduce the controller load
	Living on the edge [33]	Living on the edge employs NetFlow and sFlow inside the vSwitch and compares the performance of their monitoring results
Preprocessing	Getting traffic statistics from network devices in an SDN environment using OpenFlow [34]	Hamad *et al.* proposed a mechanism to obtain traffic statistics using OpenFlow features and analyzed the effect of the querying frequency on the network load and information accuracy
	OpenSketch [35]	OpenSketch is a three-stage pipeline architecture for data collection, offering a good tradeoff between accuracy and usage of memory resources
	ProgME [36]	ProgME is a measurement architecture with a redefined concept of organizing collected data
	Resource/accuracy tradeoffs in software-defined measurement [37]	This paper describes the Max-Cover HHH algorithm, which collects traffic information from HHH to reduce the usage of TCAM memory
	OpenTM [38]	OpenTM directly reads the counters of switches, which is faster and has more accuracy than the TM in legacy networks
	DCM [39]	DCM uses two-stage filters to implement per-flow monitoring
	iSTAMP [40]	The mechanism in iSTAMP separates the TCAM entries into two parts, one for optimally aggregating flows and another for per-flow monitoring
	Frenetic [41]	Frenetic is a high-level language built for classifying and aggregating network traffic; it also offers a functional reactive combinator library
	Pyretic [42]	Pyretic is an imperative domain-specific language embedded in Python to process packets and extend packets with virtual fields
	NetAssay [43]	NetAssay is a preliminary design for capturing traffic more closely and mapping to intent through programming
	Compiling path queries in software-defined networks [44]	The author of this study proposes the use of a query language to enable adaptive strategies to improve the data collection efficiency of SDN monitoring
Transmission	Planck [45]	Planck employs port mirroring to extract traffic without its metadata to enhance performance
	Empowering SDN controller with packet-level information [46]	This study describes a controller that offers packet-level information by sending part of the packets to the controller based on a sampling method
	OpenSample [47]	OpenSample is a traffic measurement mechanism that leverages the packet sampling of sFlow
	OpenNetMon [48]	The development in OpenNetMon periodically queries packet counters from the source and destination switches, which is appropriate for end-to-end measurement
	PayLess [49]	In PayLess, the query data of each flow depend on an adaptive monitoring algorithm, resulting in better measurement of actual utilization
	FlowCover [50]	The polling decisions in FlowCover are made by a polling scheme optimizer that reduces the communication cost
	FlowSense [51]	FlowSense aims to use a completed passive-way to give zero-cost measurements
Analysis	Revisiting traffic anomaly detection using SDN [52]	This research proposes four different anomaly detection algorithms to analyze the connection status, implement rate-limiting, block suspicious packets, and categorize anomalies
	OpenWatch [53]	OpenWatch uses a linear algorithm that dynamically changes the spatial and temporal granularity of measurements
	OpenSAFE [54]	OpenSAFE is an extensible and scalable method for mirroring and measuring large amounts of traffic in network systems
	Scalable fault management for OpenFlow [55]	This research employs scalable fault management to allow integrated operations, administration, management execution, and failure detection in the MPLS network to emit communication messages
	Baatdaat [56]	Baatdaat uses the detour path to mitigate link congestion, but the depth-first algorithm for finding detour paths is limited to depth-3 fat trees
	TinyFlow [57]	In TinyFlow, the edge switch chooses to send packets to different egress ports to break down any detected elephant flows into mice flows
Presentation	Interactive monitoring, visualization, and configuration of OpenFlow-based SDN [58]	This research provides a web interface that visually displays the network topology, traffic rate, and total flow rules
	OF@TEIN SDN testbed [59]	This research provides wide network visibility using sFlow-RT and slices the network into multiple concurrent experiments with SDN-based network virtualization

networks are compatible with SDN environments. Hence, we classify the transmission methods as SDN-original and hybrid methods in the following explanation.

1) SDN-Original Methods: The SDN-original methods only use the SDN protocol to transmit measurement data. To explain this method, we take OpenFlow developments as an example. Controller polling and switch pushing are two approaches provided by OpenFlow for collecting flow statistics from the switch. Controller polling means that the controller sends `Stats_Request` messages to the switches at the polling rate, and the switches send back the counter information with `Stats_Reply` messages. Most of the solutions use the controller polling method, including OpenTM [38] and OpenNetMon [48]. As controller polling incurs large communication overheads, PayLess [49] offers an adaptive scheduling algorithm for polling. The polling decisions of FlowCover [50] are made by polling a scheme optimizer. FlowSense [51] uses a switch pushing approach, whereby the `Flow_Removed` message is triggered whenever a flow entry expires. The link utilization in each interswitch link is then computed. Although FlowSense has zero measurement cost, it is not able to estimate the instantaneous utilization because of its nonreal-time characteristics.

2) Hybrid Methods: Several traditional developments used in legacy networks are compatible with the SDN environment. These methods can be classified as packet based and flow based. There are two types of packet-based methods: port mirroring and packet sampling. Port mirroring is supported by most of the modern switches, enabling a variety of network monitoring and security applications. When port mirroring is enabled, traffic destined for a single port is mirrored to a monitoring port that is connected to a monitoring or intrusion detection system. For instance, Planck [45] employs port mirroring to replicate packets without the metadata, thus enhancing the traffic measurement efficiency. Shirali-Shahreza and Ganjali [46] used packet sampling to select one packet out of every N and send copied parts of the sampled packets to the controller, thus providing packet-level information. OpenSample [47] is another application that leverages sFlow's packet sampling to provide near-real-time measurements of network load and individual flows. The advantage of OpenSample is that it can be deployed without modifying the end-host hardware. OpenSample works with unmodified SDN-enabled switches with sFlow's functionality.

D. Analysis

In SDN environments, the analysis functions are usually integrated into the controller or placed on the application plane. It is common to send feedback to the SDN controller for adaptation. There are four common functions in the analysis phase: traffic statistics, anomaly detection, fault management, and traffic engineering.

1) Traffic Statistics: Throughput, packet loss, latency, and link utilization are fundamental aspects of network evaluation. In SDN, OpenNetMon [48] uses the counters kept by OpenFlow switches to calculate the throughput and packet loss. The measurement of latency in OpenNetMon emulates the transmission of probe packets created by the controller. By comparing the departure time with the arrival time of these packets, the latency can be calculated. As the measuring points are strongly related to the source and destination switches, the scope of OpenNetMon is appropriate for end-to-end measurement in a single-domain network controlled by one SDN controller. Planck [45] tracks the sequence numbers and receiving time of TCP packets to compute the throughput of TCP flows. The controller then totals the throughput of all flows traversing a given link to compute the link utilization. Note that SDN flows match the wildcard rules in flow entries, whereas TCP flows are defined according to five tuples of packets. The traditional flow-based monitoring tools use ossified tuples to define flows, whereas the flow rules are defined by operators. In addition, SDN controllers use the action of flow entries to determine how the switches react to a flow.

2) Anomaly Detection: Intrusions such as denial-of-service attacks, fake packet insertion, and unauthorized programs can manipulate connection requests, thus influencing network operations. To detect possible threats, Mehdi *et al.* [52] proposed several different anomaly detection algorithms to analyze the connection status, implement rate-limiting, block suspicious packets, and categorize anomalies. OpenWatch [53] provides adaptive flow counting methods by using a linear prediction algorithm that dynamically changes the granularity of the anomaly detection measurements. If the traffic varies quickly, linear schemes may not provide accurate estimates. OpenSAFE [54] uses a policy language called ALARMS and a modified SDN controller to manage the programmed network.

3) Fault Management: This function includes two parts: fault detection and fault recovery. Fault detection identifies failures and triggers alarms, whereas fault recovery determines what should be done to troubleshoot the network, such as computing detours or alternate paths to minimize the impact of traffic disruption. Kempf *et al.* [55] proposed a scalable fault management system that allows integrated operations, administration, and management execution. They implemented failure detection in networks by introducing logical group ports that emit monitoring messages without additional processing overheads. Besides the fault management of links and forwarding device failures, the controller failover issue is inevitable. In the SDN architecture, the controller operates the logically centralized control of the enterprise network infrastructure, network policies, and data flows. The forwarding devices in the data plane need to remain connected to the controller in order to accept forwarding decisions for every decision. The failure of the controller can cause critical problems. If the SDN controller crashes, the whole SDN network may stop forwarding packets. As a result, the SDN controller should avoid single-point-of-failure events in handling system and software failures. The design of the open network operating system (ONOS [63]) supports a controller cluster that guarantees the system will continue operating when one of the controllers fails by redistributing work to other remaining controllers. As in ONOS, OpenDaylight [64] uses a cluster composed of multiple controllers to ensure the availability of the system. The difference between the ONOS and OpenDaylight clusters is in the controller relationship. The ONOS controllers operate in equal roles for each switch, whereas the OpenDaylight controllers are setup in master–slave roles.

4) Traffic Engineering: As the equal-cost multipath algorithm [65] occasionally mixes up the elephant and mice flows, Baatdaat [56] uses SDN concepts to offer a hardware-based link measurement module in line with the NetFPGA switch. This enables correct calculations of link utilization. Possible detour paths are constructed by a depth-search algorithm limited to a depth-3 fat tree. The design of TinyFlow [57] records the downstream ports and chooses a different egress port to break up elephant flows into mice flows once the byte count exceeds some threshold. However, as TinyFlow uses all available egress ports for flow management, other existing flows may exceed the threshold. This shortcoming will hopefully be improved in future research.

E. Presentation Developments

In terms of presentation developments, there is a trend toward using interactive and graphic methods of visualizing data. We include three directions for research: the topology graphical user interface (GUI), real-time presentation, and network service interface.

1) Topology GUI: This combines the discovery of the network topology with a display of the network graph. Although there is no official standard that defines a topology discovery method, most of the current controllers (e.g., Floodlight [66], Ryu [67], and Beacon [68]) have already implemented such GUIs. During the topology discovery process, the SDN controller detects the current network topology by leveraging the link layer discovery protocol (LLDP) [69]. LLDP sends packets to each port of each switch, and the switches then separate the received packets to the corresponding ports. All switches receive LLDP packets from each port, except the controller port, and forward the packets to the controller via a Packet-In message. The controller extracts the payload of these packets and adds information to its topology database. By adding a controller module or using applications, we can visualize the topology database of the network.

2) Real-Time Presentation: The sFlow-RT analytics engine [70] aims to deliver real-time visibility for SDN networks. Its agent uses the sFlow protocol to collect packets at the defined sampling rate and sends them to the sFlow-RT collector. sFlow-RT operates in the control plane to receive the sampled packets and process them into matrices of statistics based on the flow. The matrices can be graphically displayed via the representational state transfer (REST) [71] API. Rehman *et al.* [59] developed an sFlow monitoring engine and used sFlow-RT to implement traffic visualization in the OF@TEIN testbed.

3) Management Interface: This represents the managed flow tables, reflects real-time traffic information, and cautions the operators when elephant flows choke the network or unexpected system errors occur. Certain open-source SDN controllers (e.g., ONOS, OpenDaylight, and Ryu) provide an API configuration to add/remove flow entries or handle network events. Isolani *et al.* [58] developed an interactive interface that can visually display the network topology, along with the download and upload traffic rates. The controller polling interval and idle timeout configuration can be set and the corresponding change of traffic rate observed. The use of programming composition to achieve functional flexibility is a current trend in SDN.

F. Summary

Currently, the SDN concept is widely applied in many network fields, such as campus networks [72] and service providers [73]. According to a survey conducted by IHS Markit [74], around 75% of networking carriers have already deployed or will deploy SDN. Many researchers are also focusing on evaluating the production applications of SDN in the real world. For example, Bakalov [75] introduced three major challenges related to shifting real networks into SDN. In their research, the frequent interactions required to update the real-time network status may generate huge management overheads. Fulfilling the necessary services with a frequent new set of use cases is an important consideration when deploying SDN in a production environment. Furthermore, friendly control with software-based methods in SDN is the most attractive benefit to network operators. As SDN monitoring covers many network measurement methods, it is possible to orchestrate the application plane, control plane, and data plane to satisfy management demands. With mechanisms to control forwarding rules and actions, the SDN monitoring results can be extended using many creative network adaptation techniques. Based on the review of research in this section, we now briefly summarize the different uses of SDN monitoring.

1) By User Role: When migrating from a legacy network architecture to SDN, both the network operators and users must embrace the changes. Operators have to use SDN methods to manage the network, whereas users may need to change their perception about the network service. The SDN architecture provides a way of making service-oriented operations for network applications.

a) Operator: SDN takes apart and abstracts the control plane from the switches, routers, and other network components. It is more convenient for network operators to troubleshoot the separate logical and physical layers. Centralized control of a network enables operators to maintain large-scale networks, avoiding the need to configure policies switch-by-switch. Additionally, with the help of monitoring information, operators are able to balance resources in large-scale networks. Therefore, SDN monitoring techniques have already been implemented in enterprise and data center solutions. For instance, B4 [76] demonstrated the practice of bringing the SDN architecture into an enterprise network. The aim is to drive efficiency improvements in terms of managing traffic forwarding over a wide area network (WAN). Furthermore, Hong *et al.* [77] presented software-driven WAN (SWAN), a system that boosts the utilization of data center networks to satisfy transmission demands. According to their experimental results, SWAN is able to carry more traffic than traditional multiprotocol label switching (MPLS) techniques.

b) End user: In legacy networks, users generally have little ability to control the network, whereas network function virtualization (NFV) [78] turns network services into adjustable objects within SDN. For example, by obtaining the network status from presentation developments, network users are able to send back essential reactions to the network controller through NFV. By doing this, the interaction between application and network system becomes more collaborative and adaptive [79].

2) By Deployment Environment: Many academic and commercial institutes have already deployed SDN internally. However, most of the global network is still operating on the traditional architecture. Local SDN networks require the use of transit legacy networks to deliver packets to other SDN networks. In these circumstances, the local SDN controllers can only manage their own regions, while the operation of the intermediate network is managed by non-SDN methods. To find capable developments for network monitoring, we have to divide the deployment environment into two types: pure SDN and hybrid SDN (hSDN). The possible applications for enabling SDN monitoring are described in the following.

a) Pure SDN environment: In this environment, all network nodes are SDN enabled and are fully managed by the SDN controller. Most of the monitoring approaches in the development instances are implemented in pure SDN environments. The only concern about monitoring a pure SDN is the scale of the network. If it is small, then employing a powerful controller can fulfill most control requirements. Using OpenSketch and iSTAMP helps an SDN switch achieve better performance in obtaining data from the switch hardware. However, if the scale of the network is huge (e.g., data center networks and large-scale network testbeds) [80], the monitoring performance becomes an important issue. To reduce the controller load, Phan and Fukuda [81] proposed a network-wide monitoring solution for SDN. Using alternative operations such as HONE may also help to decrease the traffic monitoring load.

b) Hybrid environment: As a combination of legacy and SDN architectures, network managers may have to use both non-SDN and SDN methods to monitor network devices in a hybrid environment. Vissicchio *et al.* [82] thoroughly discussed such a scenario. They classified the hSDN models that combine SDN with legacy networks in network deployment. The challenge of monitoring hybrid environments is to send corresponding instructions to both non-SDN and SDN devices during the collection and transmission phases.

Living on the Edge, OpenSample, and SUMA [83] are reference solutions that may enable compatible measurement solutions. There is also an extension called heterogeneous SDN [84], which aims to apply SDN concepts as a means of improving the controllability of heterogeneous communication systems such as 5G mobile networks [85], VANets [86], sensor networks [87], and the Internet of Things [88]. By integrating SDN management techniques, these systems can achieve intelligent network control. For example, in the 5G SDN mobile network, the traffic statistics reported by small cells and tiny cells can be used to promote traffic engineering and energy saving. The monitoring results are useful in improving the spectrum adaptation of base stations using software-defined control [85].

3) By Operation Purpose: Using the operation purpose to determine the required deployment in SDN monitoring is essential for monitoring developments. Here, we describe three important purposes for deploying SDN monitoring.

a) Improving monitoring performance: Achieving low-cost and high-accuracy measurements is the ideal scenario in network monitoring. To achieve better performance, some compromise in system ability is an option. For example, DevoFlow [89] removes unnecessary flow table entries, which reduces the flow measurement overhead by decreasing the number of active flow entries.

b) Supporting traffic management: SDN has the ability to distinguish between the characteristic issues of security, fault tolerance, and traffic engineering. In security analysis, highly varying traffic or suspicious connections should be detected by anomaly detection algorithms. Fault-tolerant mechanisms offer available paths for failover operations. To meet QoS requirements, traffic engineering mechanisms adjust the priority of flows. Deploying MicroTE or Mahout can help to track mice and elephant flows.

c) Commonality control: Although open-source controllers provide platforms for SDN network programming, the different APIs and control instructions impose limitations. The commonality of controller APIs is not only a requirement, but also a hot research issue in SDN control and monitoring. The use of programmatic languages can bring extended controllability into SDN. For example, Pyretic can be built as an abstract handler for control collaboration, although it is no longer being developed.

IV. Open Issues

A. Supporting Adaptive Measurement

In general cases, network measurement tasks are configured with fixed metrics to sample the traffic regularly, although adaptive metrics may be more efficient during the actual measurements. For instance, according to the observed network behavior, the SDN controller can change the per-flow metrics to approach more fine-grained measurement. There are several studies that focus on this issue. OpenMeasure [90] leverages the SDN controller to update the monitoring rules with learning predictions in real time. Wellem *et al.* [91] implemented the Count-Min sketch algorithm inside the NetFPGA for adaptive measurements. Moreover, HashPipe [92] can track heavy flows with high accuracy on the SDN switches, using pipeline and hash tables to manage flow identifiers and counters to achieve better performance.

B. Toward Real-Time Analytics

For comprehensive management, the designed framework must reduce the monitoring overheads while flexibly allocating network resources in order to process and visualize traffic statistics in real time. Traffic engineering is responsible for improving performance at both the traffic and resource levels. Akyildiz *et al.* [93] laid down the roadmap for traffic engineering in SDN networks. They specified four indexes: flow management, fault tolerance, topology update, and traffic analysis. We believe that the trend of intelligently using flow entries will remain a hot topic, and the powerful SDN analytics engines of the future will be able to process the traffic status in almost-real time.

C. Cyber-Security Support

As SDN decouples data and control, the control plane will become the new target for malicious attacks. Security measures can be enabled by redirecting or filtering traffic flows based on the packet header or payload. For example, FleXam [94] implements packet-level information by redefining sampling as a new action (i.e., `OFPAT_SAMPLING`) to be assigned to each flow. FleXam also implements the so-called threshold random walk, a port scan detection technique. The OrchSec architecture [95] orchestrates network monitoring and SDN control functions to develop security applications. By using the REST API, security applications in OrchSec can periodically query traffic. Once a suspicious activity is detected, the application takes action to drop the harmful packets. Furthermore, Liyanage *et al.* [96] proposed a method for leveraging monitoring and data collection to detect security threats in mobile SDN networks.

D. Cloud Application Integration

With the increasing usage of cloud services, there is a need to enhance network control functionality for handling traffic in the cloud. For example, Neutron [97] is an SDN-enabled development that provides virtual network management in OpenStack [98]. Meridian [99] is a prototype SDN controller platform for network services in cloud environments, while NetGraph [100] uses a topology service module to obtain the topology of the underlying network and up-to-date continuous streams to update this information. CloudWatcher [101] applies security in cloud network monitoring services using four routing algorithms, which detour packets to preinstalled network security devices for inspection.

E. Quality-of-Experience Monitoring

To augment QoS, the concept of quality-of-experience (QoE) [102] has emerged to allow providers to observe user perception, experience, and expectations. Using packet-based or flow-based measurement results, SDN can provide sophisticated QoS management for traffic engineering implementations. As effective QoE control mechanisms require QoS parameters to make decisions, Fiedler *et al.* [103] proposed a generic formula to connect QoE and QoS parameters. Kassler *et al.* [104] also proposed a system architecture for QoE monitoring and management based on two functions: the QoS matching and optimization function residing in the application layer and the path assignment function residing in the control layer. Farshad *et al.* [105] introduced an in-network QoE measurement framework that provided monitoring for HTTP adaptive streaming, and Jarschel *et al.* [106] also implemented QoE approaches based on deep packet inspection (DPI) to enhance video streaming transmission.

F. Application-Aware Networking

To enhance application-driven adaptations and on-demand service, concepts incorporating application programming into SDN have started to appear [107]. Qian *et al.* [108] provided deep insights into the application usage and the relationship between network usage with user mobility in cellular data networks, thus helping network designers and operators to adjust their network capacity management. Atlas [109] is a crowdsourcing approach to detect fine-grained applications from mobile agents. These are then sent to a machine-learning trainer inside the control plane for classification. Mekky *et al.* [110] proposed an extended application-aware SDN architecture that generalizes forwarding abstractions, including layer 4–7 information. For improved efficiency, their implementation leverages the application logic at the switches. Moreover, FlowQoS [111] is a reference design that performs per-flow application-based QoS by delegating the application identification and QoS configuration to the SDN controller.

G. Software-Defined Internet Exchange

The high-level abstraction of network services and management functions gave rise to the software-defined Internet exchange (SDX) [112], [113], which interconnect multiple SDN domains. In this way, operations such as network advertisement and configuration can be software controlled. Such a combination of traffic monitoring in SDXs will bring benefits for managing multidomain networks and monitoring collaborated services. As SDN practices grow at a fast pace, it can be expected that SDX will become an important issue in global network research.

V. Conclusion

This paper has surveyed the current state of the art in SDN monitoring, including design ideas, methods, and possible improvements. Integrated perspectives on monitoring issues have been introduced, and traditional monitoring and SDN monitoring methods have been compared. The development concepts, research directions, and open issues of SDN have been reviewed and discussed. For readers who are planning to study SDN monitoring, several research directions and suggestions are given as follows.

1) *Making monitoring more efficient:* One of the critical tasks in network monitoring is to reduce the operation overheads when measuring the network status and summarizing statistical information. The operation overheads grow according to the scale and complexity of the network. For large-scale SDN networks, there is a need for smart algorithms in data selection, scheduling, and sampling. Balancing performance and accuracy is a critical problem for operators.
2) *Exploring potential uses:* To fulfill various use cases in SDN monitoring, supporting potential utilization scenarios in advance is a desirable aspect of development. As most of the parts of the global network are still using traditional architecture, it is expected that the hSDN mode will be the deployment solution for some time. Therefore, choosing capable designs for application into the monitoring environment is vital. Making good use of programming languages and APIs will help researchers develop various monitoring mechanisms for achieving more flexible, adaptive, and high-level control characteristics in SDN monitoring.
3) *Transparency:* In network virtualization, the virtual network technology provides a powerful technique for utilizing the network. Supervising both physical and logical networks and ensuring that mapping information remains transparent is the challenge in monitoring the networks. To provide a clear view of the physical and virtualized parts in each slice, the transparency functionality is a practical development that supports network monitoring.
4) *Inspection and analysis:* It is important to network operators that the usage of both applications and users can be identified. In terms of inspection, monitoring network packets enables network operators to secure the network, meeting the various and changeable operations safely. However, checking layer 2–4 headers is no longer sufficient to extract the packet. Exploiting multilayer examination and DPI are increasingly important for fine-grained security monitoring.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions on the paper.

References

[1] A. I. Coates, A. O. Hero, III, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal Process. Mag.*, vol. 19, no. 3, pp. 47–65, May 2002.

[2] 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.

[3] A. Campbell, G. Coulson, and D. Hutchison, "A quality of service architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 2, pp. 6–27, 1994.

[4] T. Fawcett and F. Provost, "Activity monitoring: Noticing interesting changes in behavior," in *Proc. ACM Int. Conf. Knowl. Discovery Data Mining*, 1999, pp. 53–62.

[5] H. Zimmermann, "OSI reference model—The ISO model of architecture for open systems interconnection," *IEEE Trans. Commun.*, vol. COM-28, no. 4, pp. 425–432, Apr. 1980.

[6] A. Pras *et al.*, "Key research challenges in network management," *IEEE Commun. Mag.*, vol. 45, no. 10, pp. 104–110, Oct. 2007.

[7] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[8] 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.

[9] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, Third Quarter 2014.

[10] A. Yassine, H. Rahimi, and S. Shirmohammadi, "Software defined network traffic measurement: Current trends and challenges," *IEEE Instrum. Meas. Mag.*, vol. 18, no. 2, pp. 42–50, Apr. 2015.

[11] Z. Shu *et al.*, "Traffic engineering in software-defined networking: Measurement and management," *IEEE Access*, vol. 4, pp. 3246–3256, 2016.

[12] M. Cheikhrouhou and J. Labetoulle, "Efficient instrumentation of management information models with SNMP," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, 2000, pp. 477–490.

[13] C. Williamson, "Internet traffic measurement," *IEEE Internet Comput.*, vol. 5, no. 6, pp. 70–74, Nov./Dec. 2001.

[14] S. Lee, K. Levanti, and H. S. Kim, "Network monitoring: Present and future," *Comput. Netw.*, vol. 65, pp. 84–98, 2014.

[15] V. Mohan, Y. J. Reddy, and K. Kalpana, "Active and passive network measurements: A survey," *Int. J. Comput. Sci. Inf. Technol.*, vol. 2, no. 4, pp. 1372–1385, 2011.

[16] R. Presuhn, "Management information base (MIB) for the simple network management protocol (SNMP)," 2002. [Online]. Available: https://www.ietf.org/rfc/rfc3418.txt

[17] C. Lonvick, "The syslog protocol," 2001. [Online]. Available: https://www.ietf.org/rfc/rfc3164.txt

[18] S. Kandula, R. Chandra, and D. Katabi, "What's going on?: Learning communication rules in edge networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 87–98, 2008.

[19] T. Oetiker and D. Rand, "MRTG: The multi router traffic grapher," in *Proc. Syst. Admin. Conf.*, 1998, pp. 141–148.

[20] T. Oetiker, "RRDtool: Round robin database tool," Mar. 14, 2017. [Online]. Available: https://oss.oetiker.ch/rrdtool/index.en.html

[21] P. Shivakumar and N. P. Jouppi, "Cacti 3.0: An integrated cache timing, power, and area model," Compaq Comput. Corporation, Palo Alto, CA, USA, Tech. Rep. 2001/2, 2001.

[22] B. Cheswick, H. Burch, and S. Branigan, "Mapping and visualizing the internet," in *Proc. USENIX Annu. Tech. Conf.*, 2000, pp. 1–12.

[23] J. A. Wickboldt, W. P. De Jesus, P. H. Isolani, C. B. Both, J. Rochol, and L. Z. Granville, "Software-defined networking: Management requirements and challenges," *IEEE Commun. Mag.*, vol. 53, no. 1, pp. 278–285, Jan. 2015.

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

[25] "OpenFlow specification 1.3." Mar. 14, 2017. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf

[26] "Elasticsearch." Mar. 14, 2017. [Online]. Available: https://www.elastic.co/

[27] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *J. Amer. Statist. Assoc.*, vol. 91, no. 433, pp. 365–377, 1996.

[28] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Comput. Netw.*, vol. 72, pp. 74–98, 2014.

[29] M. Yang, Y. Li, D. Jin, L. Zeng, X. Wu, and A. V. Vasilakos, "Software-defined and virtualized future mobile and wireless networks: A survey," *Mobile Netw. Appl.*, vol. 20, no. 1, pp. 4–18, 2015.

[30] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proc. Conf. Emerg. Netw. Exp. Technol.*, 2011, Art. no. 8.

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

[32] P. Sun, M. Yu, M. J. Freedman, J. Rexford, and D. Walker, "HONE: Joint host-network traffic management in software-defined networks," *J. Netw. Syst. Manage.*, vol. 23, no. 2, pp. 374–399, 2015.

[33] V. Mann, A. Vishnoi, and S. Bidkar, "Living on the edge: Monitoring network flows at the edge in cloud data centers," in *Proc. Int. Conf. Commun. Syst. Netw.*, 2013, pp. 1–9.

[34] D. J. Hamad, K. G. Yalda, and I. T. Okumus, "Getting traffic statistics from network devices in an SDN environment using OpenFlow," in *Proc. Inf. Technol. Syst.*, 2015, pp. 7–11.

[35] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with OpenSketch," in *Proc. USENIX Symp. Netw. Syst. Des. Implementation*, 2013, pp. 29–42.

[36] L. Yuan, C.-N. Chuah, and P. Mohapatra, "ProgME: Towards programmable network measurement," *IEEE/ACM Trans. Netw.*, vol. 19, no. 1, pp. 115–128, Feb. 2011.

[37] M. Moshref, M. Yu, and R. Govindan, "Resource/accuracy tradeoffs in software-defined measurement," in *Proc. ACM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 73–78.

[38] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "OpenTM: Traffic matrix estimator for OpenFlow networks," in *Proc. Int. Conf. Passive Active Netw. Meas.*, 2010, pp. 201–210.

[39] Y. Yu, C. Qian, and X. Li, "Distributed and collaborative traffic monitoring in software defined networks," in *Proc. ACM Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 85–90.

[40] M. Malboubi, L. Wang, C.-N. Chuah, and P. Sharma, "Intelligent SDN based traffic (de)aggregation and measurement paradigm (iSTAMP)," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2014, pp. 934–942.

[41] N. Foster *et al.*, "Frenetic: A network programming language," *ACM Sigplan Notices*, vol. 46, no. 9, pp. 279–291, 2011.

[42] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing software defined networks," in *Proc. USENIX Symp. Netw. Syst. Des. Implementation*, 2013, pp. 1–13.

[43] S. Donovan and N. Feamster, "Intentional network monitoring: Finding the needle without capturing the haystack," in *Proc. ACM Workshop Hot Topics Netw.*, 2014, pp. 1–7.

[44] S. Narayana, J. Rexford, and D. Walker, "Compiling path queries in software-defined networks," in *Proc. Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 181–186.

[45] J. Rasley *et al.*, "Planck: Millisecond-scale monitoring and control for commodity networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 407–418, 2015.

[46] S. Shirali-Shahreza and Y. Ganjali, "Empowering software defined network controller with packet-level information," in *Proc. IEEE Int. Conf. Commun.*, 2013, pp. 1335–1339.

[47] J. Suh, T. T. Kwon, C. Dixon, W. Felter, and J. Carter, "OpenSample: A low-latency, sampling-based measurement platform for commodity SDN," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2014, pp. 228–237.

[48] N. L. Van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network monitoring in OpenFlow software-defined networks," in *Proc. IEEE Netw. Oper. Manage. Symp.*, 2014, pp. 1–8.

[49] 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. Manage. Symp.*, 2014, pp. 1–9.

[50] Z. Su, T. Wang, Y. Xia, and M. Hamdi, “FlowCover: Low-cost flow monitoring scheme in software defined networks,” in *Proc. IEEE Global Commun. Conf.*, 2014, pp. 1956–1961.

[51] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, “FlowSense: Monitoring network utilization with zero measurement cost,” in *Proc. Int. Conf. Passive Active Netw. Meas.*, 2013, pp. 31–41.

[52] S. A. Mehdi, J. Khalid, and S. A. Khayam, “Revisiting traffic anomaly detection using software defined networking,” in *Proc. Int. Workshop Recent Adv. Intrusion Detection*, 2011, pp. 161–180.

[53] Y. Zhang, “An adaptive flow counting method for anomaly detection in SDN,” in *Proc. ACM Conf. Emerg. Netw. Exp. Technol.*, 2013, pp. 25–30.

[54] J. R. Ballard, I. Rae, and A. Akella, “Extensible and scalable network monitoring using OpenSAFE,” in *Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw.*, 2010.

[55] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takács, and P. Sköldström, “Scalable fault management for OpenFlow,” in *Proc. IEEE Int. Conf. Commun.*, 2012, pp. 6606–6610.

[56] F. P. Tso and D. P. Pezaros, “Baatdaat: Measurement-based flow scheduling for cloud data centers,” in *Proc. IEEE Symp. Comput. Commun.*, 2013, pp. 765–770.

[57] H. Xu and B. Li, “TinyFlow: Breaking elephants down into mice in data center networks,” in *Proc. IEEE Int. Workshop Local Metropolitan Area Netw.*, 2014, pp. 1–6.

[58] P. H. Isolani, J. A. Wickboldt, C. B. Both, J. Rochol, and L. Z. Granville, “Interactive monitoring, visualization, and configuration of OpenFlow-based SDN,” in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, 2015, pp. 207–215.

[59] S. U. Rehman, W.-C. Song, and M. Kang, “Network-wide traffic visibility in OF@TEIN SDN testbed using sFlow,” in *Proc. Asia-Pacific Netw. Oper. Manage. Symp.*, 2014, pp. 1–6.

[60] NetFlow, Mar. 14, 2017. [Online]. Available: http://www.cisco.com/go/netflow

[61] sFlow, Mar. 14, 2017. [Online]. Available: http://www.sflow.org/

[62] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, “Traffic matrix estimation: Existing techniques and new directions,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 161–174, 2002.

[63] Open Network Operating System, Sep. 30, 2017. [Online]. Available: http://onosproject.org/

[64] OpenDaylight, Mar. 14, 2017. [Online]. Available: https://www.opendaylight.org/

[65] C. E. Hopps, *Analysis of an Equal-Cost Multi-Path Algorithm*, RFC 2992, 2000.

[66] Floodlight, Mar. 14, 2017. [Online]. Available: http://www.projectfloodlight.org/floodlight/

[67] Ryu, Mar. 14, 2017. [Online]. Available: https://osrg.github.io/ryu/

[68] D. Erickson, “The beacon openflow controller,” in *Proc. ACM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 13–18.

[69] P. Congdon and B. Lane, “802.1ab—Station and media access control connectivity discovery,” 2005. [Online]. Available: http://www.ieee802.org/1/pages/802.1ab.html

[70] sFlow-RT, Mar. 14, 2017. [Online]. Available: http://www.sflow-rt.com/

[71] Richardson, Leonard and Ruby, SamRoy Thomas Fielding, “RESTful web servicesArchitectural Styles and the design of network-based software architectures, representational state transfer (REST),” 20082000. [Online]. Available: https://www.ics.uci.edu/ fielding/pubs/dissertation/rest_arch_style.htm

[72] M. Kobayashi *et al.*, “Maturing of OpenFlow and software-defined networking through deployments,” *Comput. Netw.*, vol. 61, pp. 151–175, 2014.

[73] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, “On controller performance in software-defined networks,” in *Proc. USENIX Workshop Hot Topics Manage. Internet, Cloud, Enterprise Netw. Services*, 2012, p. 10.

[74] M. Howard, “75 percent of carriers surveyed have deployed or will deploy SDN this year,” 2016. [Online]. Available: https://technology.ihs.com/583348

[75] V. Bakalov, “Shifting to SDN? 3 ways it will affects network monitoring,” 2015. [Online]. Available: http://www.networkcomputing.com/applications/shifting-sdn-3-ways-it-will-affect-network-monitoring/1619610103

[76] 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.

[77] C.-Y. Hong *et al.*, “Achieving high utilization with software-driven WAN,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, 2013, pp. 15–26.

[78] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, “Network function virtualization: Challenges and opportunities for innovations,” *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[79] T. Koponen *et al.*, “Network virtualization in multi-tenant datacenters,” in *Proc. USENIX Symp. Netw. Syst. Des. Implementation*, 2014, pp. 203–216.

[80] M. Jarschel, T. Zinner, T. Hoßfeld, P. Tran-Gia, and W. Kellerer, “Interfaces, attributes, and use cases: A compass for SDN,” *IEEE Commun. Mag.*, vol. 52, no. 6, pp. 210–217, Jun. 2014.

[81] X. T. Phan and K. Fukuda, “Toward a flexible and scalable monitoring framework in software-defined networks,” in *Proc. Int. Conf. Adv. Inf. Netw. Appl. Workshops*, 2017, pp. 403–408.

[82] S. Vissicchio, L. Vanbever, and O. Bonaventure, “Opportunities and research challenges of hybrid software defined networks,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 70–75, 2014.

[83] T. Choi, S. Song, H. Park, S. Yoon, and S. Yang, “SUMA: Software-defined unified monitoring agent for SDN,” in *Proc. IEEE Netw. Oper. Manage. Symp.*, 2014, pp. 1–5.

[84] M. Mendonca, K. Obraczka, and T. Turletti, “The case for software-defined networking in heterogeneous networked environments,” in *Proc. ACM Conf. Emerg. Netw. Exp. Technol.*, 2012, pp. 59–60.

[85] R. Trivisonno, R. Guerzoni, I. Vaishnavi, and D. Soldani, “SDN-based 5G mobile networks: Architecture, functions, procedures and backward compatibility,” *Trans. Emerg. Telecommun. Technol.*, vol. 26, no. 1, pp. 82–92, 2015.

[86] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, and E. Cerqueira, “Towards software-defined vanet: Architecture and services,” in *Proc. Annu. Mediterranean Ad Hoc Netw. Workshop*, 2014, pp. 103–110.

[87] T. Luo, H.-P. Tan, and T. Q. Quek, “Sensor OpenFlow: Enabling software-defined wireless sensor networks,” *IEEE Commun. Lett.*, vol. 16, no. 11, pp. 1896–1899, Nov. 2012.

[88] 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. Manage. Symp.*, 2014, pp. 1–9.

[89] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, “DevoFlow: Scaling flow management for high-performance networks,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 254–265, 2011.

[90] C. Liu, A. Malboubi, and C.-N. Chuah, “OpenMeasure: Adaptive flow measurement & inference with online learning in SDN,” in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2016, pp. 47–52.

[91] T. Wellem, Y.-K. Lai, C.-H. Cheng, Y.-C. Liao, L.-T. Chen, and C.-Y. Huang, “Implementing a heavy hitter detection on the NetFPGA OpenFlow switch,” in *Proc. IEEE Int. Symp. Local Metropolitan Area Netw.*, 2017, pp. 1–2.

[92] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, “Heavy-hitter detection entirely in the data plane,” in *Proc. Symp. SDN Res.*, 2017, pp. 164–176.

[93] 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, 2014.

[94] S. Shirali-Shahreza and Y. Ganjali, “FleXam: Flexible sampling extension for monitoring and security applications in OpenFlow,” in *Proc. ACM Workshop Hot Topics Netw.*, 2013, pp. 167–168.

[95] A. Zaalouk, R. Khondoker, R. Marx, and K. Bayarou, “OrchSec: An orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions,” in *Proc. IEEE Netw. Oper. Manage. Symp.*, 2014, pp. 1–9.

[96] M. Liyanage *et al.*, “Security for future software defined mobile networks,” in *Proc. Int. Conf. Next Gener. Mobile Appl., Serv. Technol.*, 2015, pp. 256–264.

[97] Neutron, Sep. 30, 2017. [Online]. Available: https://github.com/openstack/neutron

[98] OpenStack, Mar. 14, 2017. [Online]. Available: http://docs.openstack.org/index.html

[99] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey, and G. Wang, “Meridian: An SDN platform for cloud network services,” *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 120–127, Feb. 2013.

[100] R. Raghavendra, J. Lobo, and K.-W. Lee, "Dynamic graph query primitives for SDN-based cloudnetwork management," in *Proc. ACM Workshop Hot Topics Netw.*, 2012, pp. 97–102.

[101] S. Shin and G. Gu, "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks," in *Proc. IEEE Int. Conf. Netw. Protocols*, 2012, pp. 1–6.

[102] K. Brunnström *et al.*, "Qualinet white paper on definitions of quality of experience," 2014. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00977812

[103] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," *IEEE Netw.*, vol. 24, no. 2, pp. 36–41, Mar./Apr. 2010.

[104] A. Kassler, L. Skorin-Kapov, O. Dobrijevic, M. Matijasevic, and P. Dely, "Towards QoE-driven multimedia service negotiation and path optimization with software defined networking," in *Proc. Int. Conf. Softw., Telecommun. Comput. Netw.*, 2012, pp. 1–5.

[105] A. Farshad, P. Georgopoulos, M. Broadbent, M. Mu, and N. Race, "Leveraging SDN to provide an in-network QoE measurement framework," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2015, pp. 239–244.

[106] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, and P. Tran-Gia, "SDN-based application-aware networking on the example of Youtube video streaming," in *Proc. Eur. Workshop Softw. Defined Netw.*, 2013, pp. 87–92.

[107] T. Zinner, M. Jarschel, A. Blenk, F. Wamser, and W. Kellerer, "Dynamic application-aware resource management using software-defined networking: Implementation prospects and challenges," in *Proc. IEEE Netw. Oper. Manage. Symp.*, 2014, pp. 1–6.

[108] L. Qian, B. Wu, R. Zhang, W. Zhang, and M. Luo, "Characterization of 3G data-plane traffic and application towards centralized control and management for software defined networking," in *Proc. IEEE Int. Congr. Big Data*, 2013, pp. 278–285.

[109] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir, "Application-awareness in SDN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 487–488, 2013.

[110] H. Mekky, F. Hao, S. Mukherjee, Z.-L. Zhang, and T. Lakshman, "Application-aware data plane processing in SDN," in *Proc. ACM Workshop Hot Topics Netw.*, 2014, pp. 13–18.

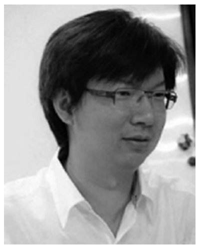
[111] M. S. Seddiki *et al.*, "FlowQoS: QoS for the rest of us," in *Proc. Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 207–208.

[112] A. Gupta *et al.*, "SDX: A software defined internet exchange," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 551–562, 2015.

[113] J. Chung, H. Owen, and R. Clark, "SDX architectures: A qualitative analysis," in *Proc. IEEE SoutheastCon*, 2016, pp. 1–8.

Pang-Wei Tsai received the B.S. degree in electronic engineering and M.S. degree in computer and communication engineering from National Cheng Kung University, Tainan, Taiwan.

His research interests include software-defined networking, cloud computing, virtualization, and network management.



Chun-Wei Tsai received the Ph.D. degree in computer science and engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2009.

In 2017, he joined the Department of Computer Science and Engineering, National Chung Hsing University, Taichung, Taiwan, where he is currently an Assistant Professor. His research interests include computational intelligence, data mining, cloud computing, and the Internet of Things.

Chia-Wei Hsu received the B.S. degree in computer science engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, and the M.S. degree in computer and communication engineering from National Cheng Kung University, Tainan, Taiwan.

She is currently with the Taiwan Semiconductor Manufacturing Company Limited, Hsinchu, Taiwan. Her research interests include software-defined networking, network monitoring, and network management.

Chu-Sing Yang is a Professor of electrical engineering with the Institute of Computer and Communication Engineering, National Cheng Kung University (NCKU), Tainan, Taiwan. He joined the Faculty of the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, in 1988, and the faculty of the Department of Electrical Engineering, NCKU, in 2006. His research interests include software-defined networking, network management, cloud computing, and cyber-security.