A Lightweight Scheme of Active-port Aware Monitoring in Software-Defined Networks

Bong-Hwan Oh, Serdar Vural, *Member, IEEE*, and Ning Wang, *Senior Member, IEEE,*

***Abstract*—Software-defined networking (SDN) is the key technology to enable network softwarization by offering a programable and flexible network control capabilities. In order to dynamically manage and reconfigure the underlying network through SDN, network-based monitoring functionality needs to be in place. However, existing network monitoring schemes are normally heavyweight which can cause substantial monitoring overhead when dealing with entire network infrastructure and complex policies. Such a limitation can be critical in a software-based network system that enables the construction of multiple networks with various network policies designed by a network operator. In this paper, we propose a new lightweight monitoring mechanism referred to as Active-port Aware Monitoring (APAM) in order to support the monitoring of complex networks with substantially reduced overhead. APAM typically monitors active ports which are the switch ports utilized by current flow rules. These active ports are dynamically monitored with reconfigurable monitoring intervals according to their port utilization. The measurement results show that APAM adapts varying traffic route due to a change of flow rules and also adjusts its monitoring performance according to network traffic dynamicity, which reduces the monitoring overhead and also improves monitoring accuracy.**

***Index Terms*—Software-Defined Networking, Network monitoring, Low overhead.**

I. Introduction

SOFTWARE Defined Networking (SDN) [1], [2] has become one of the key enabling technologies for future networks in the sense that it enables dynamic network management and control through its programmable and centralized features. The programmable feature is based on the decoupling of the control plane from the data plane, which brings the ability to dynamically change traffic forwarding rules by an SDN controller. Due to such a feature, SDN requires a signaling mechanism to communicate between the control plane and the data forwarding plane, with OpenFlow [3] being a standard communication protocol for such a purpose.

In addition, due to the necessary communication between SDN switches and SDN controller in charge, a SDN-based network inherently has a centralized feature: an SDN controller has knowledge about the status of the entire network by collecting real-time state information captured in individual SDN switches, thanks to these two features in SDN, dynamic network control can be enabled at the SDN controller side based on the real-time knowledge of the network condition in dynamic environments.

One key prerequisite to efficient network control in SDN-based networks is to obtain network conditions promptly and accurately through the network monitoring system. Such a capability allows SDN controllers to quickly adapt to traffic behavior changes by appropriately reconfiguring data-plane devices. It is worth noting that, the transmission and processing of network monitoring messages on traffic conditions consumes both network and computing resources. Such an overhead may potentially introduce a negative impact on forwarding performances in the data plane [5]. On the other hand, inadequate monitoring capabilities cannot fulfill the task of accurately and promptly reporting network and traffic dynamicity to the control plane, thus losing the agility and accuracy in responding to various events. Therefore, there is an obvious trade-off between monitoring performance and monitoring overhead in SDN-based networks.

Although various monitoring methods [6-21] have been proposed to improve monitoring performance, existing monitoring mechanisms in SDN still have potential risks concerning monitoring overheads. Most of the previous monitoring methods are based on flow monitoring which introduces a heavy monitoring overhead in accordance with the increasing of the number of flow rules in SDN. Thus, applying existing monitoring approaches may cause a substantial overhead when complex network policies based on a large number of flow rules are applied in SDN-based networks. This is particularly the case when concerning macroscopic monitoring which focuses on the entire network infrastructure potentially being large-scale. The basic monitoring strategy in SDN is to periodically collect flow and port statistics from SDN switches which is also known as polling. The main drawback of polling monitoring is it generally uses a static polling period for the entire network regardless of utilization of network infrastructure, which only can be effective at certain network conditions.

In this paper, an active-port aware monitoring (APAM) scheme is presented to mitigate the trade-off of traffic monitoring as a lightweight and macroscopic monitoring function. By using both flow rule information and port statistics, APAM not only dynamically manages the monitoring operation on individual ports but also minimizes unnecessary monitoring overhead. Using flow rule information, APAM scheme retrieves active ports that are SDN ports utilized by current flow rules and monitors a whole network topology by using some (or all if necessary) of active ports. APAM also adapts active ports and a monitoring topology according to a change of flow

This project is partially funded by the 5G Innovation Centre (5GIC).

B. Oh is with the School of Internet of Things, Xi'an Jiaotong-Liverpool University, 111 Ren'ai Road, Suzhou Industrial Park, Suzhou, Jiangsu Province, 215123, P.R. China. (e-mail: BongHwan.Oh@xjtlu.edu.cn).

S. Vural, and N. Wang are with the 5G Innovation Centre (5GIC), Institute for Communication Systems (ICS), James Clerk Maxwell Building, University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom. (e-mail: {s.vural, n.wang}@surrey.ac.uk).

rules. The active ports are then monitored by dynamically collecting port statistics depending on the bandwidth utilization of the active ports for monitoring efficiency. Thus, with APAM scheme, an SDN controller is able to not only concentrate on the active ports but also choose between reducing monitoring overhead and improving monitoring accuracy of the active ports according to their port utilization.

The rest of this paper is organized as follows. Section II briefly describes the monitoring mechanism in SDN and the motivation of this research. This is followed by the description of the APAM framework and its operation in Section III. Section IV presents the measurement results with consideration of the performance evaluation, adaptability and monitoring overhead, and robustness of APAM. Finally, we offer our conclusions in Section V.

II. Background and Motivation

As a monitoring functionality, an SDN controller can obtain network traffic information such as data forwarding rules and the amount of traffic (bytes or packets) from SDN switches. The information for data forwarding rules is obtained from flow tables in each SDN switch and the traffic information is observed by traffic statistics (port statistics and flow statistics). When an SDN switch receives network traffic from an SDN port, the SDN switch measures the amount of the network traffic received. In case of the port statistics, it records the cumulative amount of the traffic at the SDN port as its port statistics. In case of the flow statistics, the SDN switch firstly analyzes packet headers of the received traffic. If there is a flow rule matched by the received traffic, then it records the cumulative amount of the traffic matched by the flow rule as its flow statistics. Note that port statistics and flow statistics are independent of each other.

Although traffic statistics can be properly recorded in SDN switches, the statistics collected should be retrieved from SDN switches to an SDN controller in order to be utilized practically. In other words, for accurate traffic monitoring, traffic statistics should be frequently collected at an SDN controller. However, the operation for collecting traffic statistics generates not only a processing overhead at an SDN controller and SDN switches but also network overhead (management bandwidth): an SDN controller requests traffic statistics using SDN control messages, and then SDN switches send the statistics information to the SDN controller when they receive the request messages. Consequently, there is a trade-off between monitoring performance and monitoring overhead in SDN-based networks.

Various monitoring techniques have been proposed to cope with the monitoring challenge: overhead and accuracy. OpenTM [6] revealed the overhead of flow-based monitoring which is caused by collecting flow statistics from SDN switches to an SDN controller. In order to reduce this overhead, PayLess [7] and OpenNetMon [8] proposed adaptive monitoring methods which manage the rate of flow statistics requests. FlowCover [9] can reduce monitoring overhead using a flow aggregation method with a global view of network topology. In [10], both adaptive pulling rate and aggregation

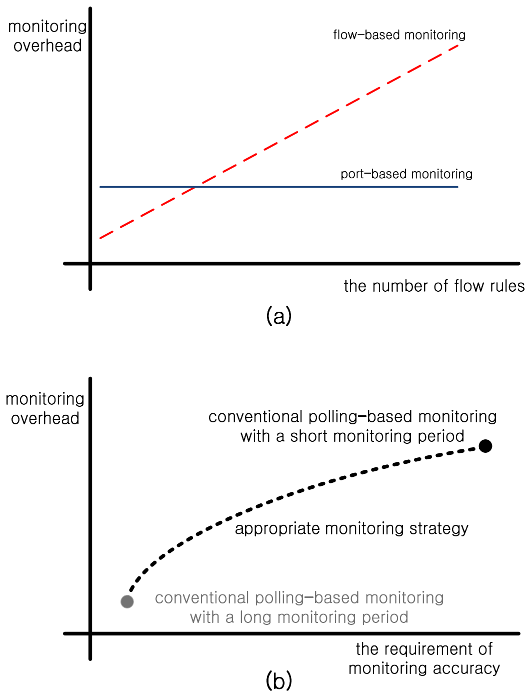


Fig. 1. Monitoring overhead according to monitoring methods.

of flows were considered for anomaly detection. Moreover, FlowSense [11] introduced a passive (pull-based) monitoring method which eliminates control messages for monitoring by using flow expiration messages. In order to mitigate the overhead related with the number of monitored flows, [12] proposed a cost-optimized monitoring scheme using wildcard-based requests and, in [13], a cost-effective monitoring method using distributed controller deployment was proposed. In [14], a self-tunning monitoring method was considered in decentrilized monitoring architecture to reduce both monitoring latency and overhead. For high accuracy (almost real-time) monitoring, sampling-based methods were considered in [15]–[17]. Planck [15] can provide a sampling monitoring with millisecond timescales using a port mirroring. OpenSample [16] proposed a TCP traffic monitoring method which has a 100-millisecond control loop by analyzing sampled TCP packets. In [17], the sampling monitoring which uses duplicated packets generated by SDN was utilized for an intrusion detection. Beyond the control plane operation, several methods were proposed to optimize monitoring information by using P4 [4] which is a language for controlling data plane of P4 switch. FlowStalker [18] subdivides a network into clusters and uses a special packet which collects monitoring information of the clusters. In [19], P4 switches preprocess monitoring information in order to forward requested statistics to an SDN controller. In FlowSpy [20], P4 switches proactively send monitoring information to a controller only when the counter of flow rules exceeds a threshold. The general SDN monitoring work is introduced and organized in [21].

However, although the previous monitoring methods have considered various aspects of monitoring overhead, they fundamentally have potential risks concerning the overhead to monitor a whole network. In the case of flow-based monitoring, a substantial overhead can be caused when there are lots of flow rules in an SDN-based network (as shown in Fig. 1a) due to complex network policies: network virtualization and network slicing techniques can accelerate the increase of flow rules in a physical network infrastructure. Furthermore, in case of a hierarchical SDN structure such as the extended SDN architecture specified by the Open Networking Foundation (ONF) [22], the monitoring burden due to detailed monitoring information can be amplified for a high-level SDN controller. In case of sampling-based monitoring, additional network bandwidth due to traffic sampling and special monitoring functions such as a traffic collector and a traffic analyzer are required. For the simplicity of implementation, most of SDN controllers apply the conventional polling-based monitoring which collects all port (flow) statistics with a static monitoring period as a default monitoring method. Although conventional monitoring is simple and requires minimum deployment cost, it can only be effective in certain situations as shown in Fig. 1b. The conventional monitoring with a short monitoring period is appropriate when high monitoring accuracy is needed. However, it keeps the high monitoring overhead even when the high monitoring accuracy is no more required. In case of the conventional monitoring with a long monitoring period, the opposite results occur.

In order to address the drawbacks, the motivation of this research is, therefore, to design a new monitoring mechanism with the following considerations:

- Monitoring mechanism specialized for macroscopic traffic information for a whole network monitoring: The fundamental information for traffic monitoring is the traffic status of the whole network. If there is a specialized function to monitor the whole network status with minimum overhead, a large amount of monitoring overhead can be reduced by using that function. SDN already supports the port statistics which are aggregated traffic information at a network interface (a link) and are obtained with a static monitoring overhead regardless of the number of flow rules.
- Optimization of polling-based port monitoring: the SDN controller can maintain the knowledge of the whole network topology, which prevents the request (or collection) of duplicated monitoring information. By collecting traffic statistics of the ports which are currently utilized by flow rules, the trade-off of traffic monitoring can be mitigated.
- Dynamic monitoring control for each individual port according to its traffic status: For efficient monitoring, each port should be controlled according to its status and monitoring control can adjust the dynamicity of port traffic conditions.

Based on these considerations, a new traffic monitoring method is proposed in this paper. The proposed monitoring scheme is a macroscopic and polling-based monitoring with traffic flow information. The proposed scheme firstly retrieves the switch ports utilized by current flow rules and these ports are called the active port in this paper. Some (all) of the active ports are then selected and the proposed scheme monitors the whole network traffic with the selected active ports, which enables to minimize monitoring overhead. For the selected active ports, the monitoring period of these ports is adjusted according to their port traffic level. This operation not only supports high monitoring accuracy in high traffic load situation but also minimizes monitoring overhead when network links (ports) are stable. In the case of the non-selected ports, the proposed scheme just keeps these ports as non-active ports or monitors these ports with minimum overhead.

III. Active-port Aware Monitoring (APAM)

The core idea of the proposed scheme, APAM, is to retrieve active ports of SDN switches and then to adjust the monitoring interval of these ports according to their utilization levels. The general operation of APAM is as follows. First, active ports are determined and port utilization of each active port is estimated by collecting its port statistics[1]. For ports with higher utilization levels, the monitoring interval is kept smaller, in order to improve monitoring accuracy. If the port utilization is low, the monitoring interval increases in order to reduce monitoring overhead, i.e. the frequency of the control messages sent to the SDN controller to retrieve port statistics.

A. SDN Architecture and Framework with APAM

The SDN architecture and framework with APAM are described in Fig. 2. The architecture consists of SDN switches, which can communicate with an SDN controller using southbound API. The OpenFlow protocol is used for the southbound API. Although APAM may have more flexibility with P4, this extension is out of scope in this paper, and need further study. The architecture includes some SDN applications which may perform some high-level SDN based control functions and can access the SDN controller via its northbound API, which could be based on Representational State Transfer (REST) or Remote Procedure Call (RPC).

The platform can support several functionalities or network information to applications using several functions in the SDN controller. Among these, a network abstraction function can provide global information on network conditions. In this study, APAM is designed as a basic network abstraction function of the SDN controller. This follows a micro-services approach, in which APAM can be independently deployed and modified when necessary. Applications and/or other functions of the SDN controller can fetch monitoring information from APAM.

B. APAM operation

1) Retrieval of Active Ports: APAM determines which ports of the SDN switches are active by processing current flow rules deployed on the switches[2]. After APAM retrieves the flow

[1]Note that APAM uses port traffic statistics as link traffic statistics because physical link traffic and physical port traffic are identical.

[2]The information on flow rules can be obtained by fetching flow tables in the SDN switches or from other functions of the SDN controller.

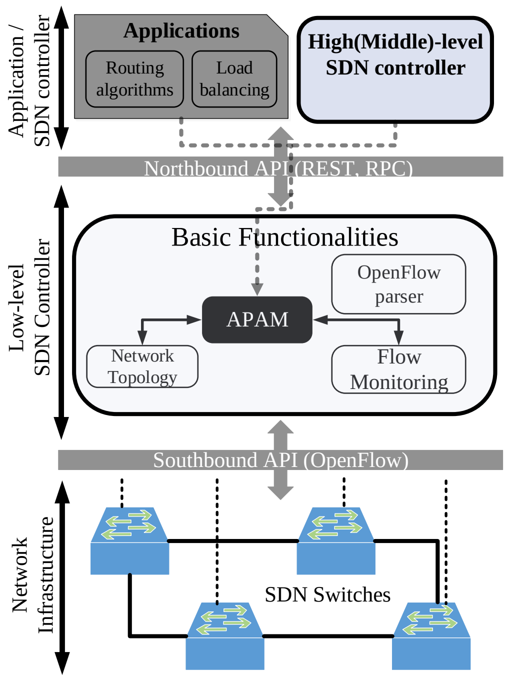


Fig. 2. SDN architecture and framework.

rules, it determines active ports based on some pre-defined conditions that can be derived from the flow rules. These pre-defined conditions can be comprised of elements of flow rules, such as *output_port* and *in_port*. The pre-defined condition used in this paper is to contain output-port elements; i.e. the ports listed as an output port are regarded as active ports. The port number of the active port and the ID of the SDN switch ID where the port is located together uniquely identify an active port. The algorithm of the retrieval operation is given in Alg. 1.

Algorithm 1 Retrieval of active ports

```
1: for $switch$ in $switches$ do
2:     $flow\_rules$ = Fetch flow rules in $switch$
3:     for $flow\_rule$ in $flow\_rules$ do
4:         $condition$ = there is an ‘output_port’ action in $flow\_rule$
5:         if $condition$ and $output\_port \neq controller\_port$ then
6:             active_ports[switch].append ($output\_port$)
7:         end if
8:     end for
9: end for
```

2) Selection of Active Ports to be Monitored: Since an active switch port may be linked to an active port of another switch (rather than a terminal device), to avoid double-counting the link statistics information, only one of these active ports is monitored. In other words, APAM needs to select which of the two switches that a monitored link is connected to should be used to monitor the link. To achieve this, APAM assigns a monitored link to a switch by means of selecting one of these two active ports for monitoring.

For this selection, APAM firstly classifies active ports into two types: the active network port and the active terminal port. The active network ports indicate the active ports connected to another switch and the active terminal ports represent the active ports connected to the terminal. APAM then performs the selection process with the active network ports. The selection process attempts to equalise the number of monitored ports across the network switches as much as possible, so that monitoring traffic load to SDN switches is kept as homogeneous as possible across the network. APAM first sorts the switches in ascending order of active ports. Note that the link information can be obtained by a topology function in the SDN controller. Then, starting from the switch with the least number of active ports, each switch selects one active network port which its neighbor switch has the most number of active ports. While considering a port for selection, APAM checks whether the other end of the active network port (i.e. the active port at the other switch which the link connects to) has already been selected. If so, APAM picks another active network port. This port selection operation continues until all active network ports have been processed; either selected or skipped. After the selection process, APAM assigns the active terminal ports to switches. Each terminal port is assigned to its switch because it can only be monitored by its switch. In the rest of the paper, both active network ports and active terminal ports are denoted by the monitoring active ports. The port selection procedure is provided in Alg. 2.

Algorithm 2 Allocating active ports to SDN switches

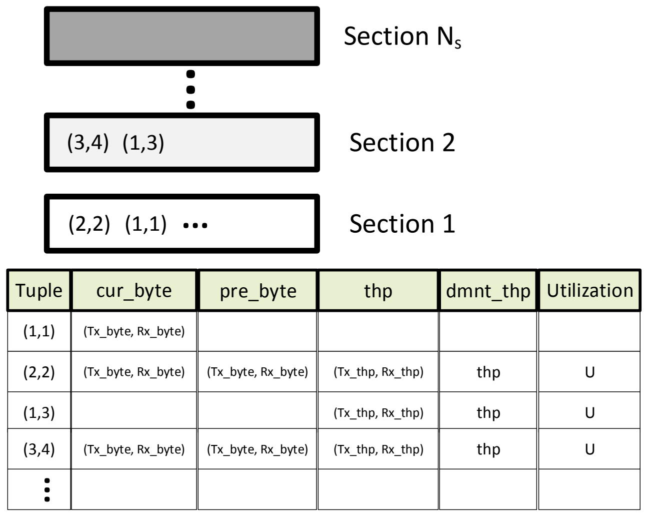
```
Term
* $port_{an}$ = an active network port
* $port_{at}$ = an active terminal port

1: $active\_network\_ports$ = find active network ports from $active\_ports$
       in Alg. 1
2: $N$ = CountActivePorts($active\_ports$)
3: $sorted\_switches$ = Sort ($switches$, ascending, $N$)
4: while $N \neq 0$ do
5:     for $switch$ in $sorted\_switches$ do
6:         $N_{switch}$ = CountActivePorts($active\_network\_ports[switch]$)
7:         if $N_{switch} \neq 0$ then
8:             $port_{an}$ = find a active network port which its neighbor switch has
                   the most number of active ports
9:             if $port_{an}$ is not monitored by other switches then
10:                $monitoring\_active\_ports$.append ($port_{an}$)
11:            end if
12:            $active\_network\_ports[switch]$.pop ($port_{an}$)
13:        end if
14:    end for
15: end while

16: $active\_terminal\_ports$ = find active terminal ports from $active\_ports$
        in Alg. 1
17: for $port_{at}$ in $active\_terminal\_ports$ do
18:     $monitoring\_active\_ports$.append ($port_{at}$)
19: end for
```

3) Adaptive and section-based port monitoring: In APAM, there are monitoring *sections* which have different monitoring intervals and they are used for grouping monitoring active ports according to their port utilization. The number of monitoring sections, $N_{\mathrm{S}i}$, is a predefined value; the more monitoring sections the finer granularity in monitoring intervals. On the other hand, having too many sections may cause high computational overhead in APAM. $N_{\mathrm{S}i}$ is to be set by the network operator.

Each monitoring section, denoted by (Si), has its own monitoring threshold ($thold_{\mathrm{S}i}$) and monitoring interval ($intvl_{\mathrm{S}i}$)



Tuple	cur_byte	pre_byte	thp	dmnt_thp	Utilization
(1,1)	(Tx_byte, Rx_byte)				
(2,2)	(Tx_byte, Rx_byte)	(Tx_byte, Rx_byte)	(Tx_thp, Rx_thp)	thp	U
(1,3)			(Tx_thp, Rx_thp)	thp	U
(3,4)	(Tx_byte, Rx_byte)	(Tx_byte, Rx_byte)	(Tx_thp, Rx_thp)	thp	U
⋮					

Fig. 3. Example of adaptive and section-based port monitoring.

which are defined as:

$$thold_{\mathrm{S}i} = i/N_{\mathrm{S}i}, \qquad i = 1, 2, \cdots, N_{\mathrm{S}i} \tag{1}$$

$$\begin{aligned} intvl_{\mathrm{S}i} =& intvl_{max} - thold_{\mathrm{S}i} * (intvl_{max} - intvl_{min}), \\ & i = 1, 2, \cdots, N_{\mathrm{S}i} \end{aligned} \tag{2}$$

where i indicates a monitoring section number, and $intvl_{max}$ and $intvl_{min}$ indicate the maximum and minimum monitoring intervals, respectively. Both $intvl_{max}$ and $intvl_{min}$ are pre-defined values. All monitoring active ports are dynamically assigned to one of the monitoring sections depending on their port utilization. Fig. 3 depicts an example of section based port monitoring. In the rest of the paper, a tuple (p, s) which is a combination of port number (p) and switch ID (s) is used for the identification of a **m**onitoring **a**ctive port (denoted by $P_{ma}(p, s)$).

The initial monitoring procedure starts at the lowest monitoring section: monitoring active ports are initially monitored at section S1. For each monitoring active port, port throughput and port utilization are estimated using its byte statistics. In the first port monitoring, the byte statistics of a monitoring active port is just recorded as current byte statistics ($byte^{cur}_{(p,s)}(link)$): the tuple $(1, 1)$ in Fig. 3 is an example of this case. The current byte statistics has information on both *received* ($byte^{cur}_{(p,s)}(R_x)$) and *transmitted* ($byte^{cur}_{(p,s)}(T_x)$) bytes. From the second port monitoring, the port throughput ($thp_{(p,s)}(link)$) is calculated by using the current ($byte^{cur}_{(p,s)}(link)$) and the previous ($byte^{pre}_{(p,s)}(link)$) byte statistics from Eq. 3. The $link$ in Eq. 3 can be the forwarding link (T_x) or the backward link (R_x). Between the received port throughput ($thp_{(p,s)}(R_x)$) and the transmitted port throughput ($thp_{(p,s)}(T_x)$), the larger throughput is regarded as the dominant throughput ($thp^{dmnt}_{(p,s)}$) of the monitoring active port (Eq. 4). The dominant port utilization ($U_{(p,s)}$) of the monitoring active port is then estimated from Eq. 5.

$$\begin{aligned} thp_{(p,s)}(link) =& 8 * \frac{(byte^{cur}_{(p,s)}(link) - byte^{pre}_{(p,s)}(link))}{intvl_{\mathrm{S}i}}, \\ & link = T_x \ or \ R_x \end{aligned} \tag{3}$$

$$thp^{dmnt}_{(p,s)} = MAX(thp_{(p,s)}(T_x),\ thp_{(p,s)}(R_x)) \tag{4}$$

$$U_{(p,s)} = thp^{dmnt}_{(p,s)} \ / \ speed_{(p,s)} \tag{5}$$

Note that (p, s), i, and $speed_{(p,s)}$ represent the tuple for port identification, the monitoring section number of the monitoring active port, and the port speed of the monitoring active port, respectively. $byte^{cur}_{(p,s)}(link)$ is the byte statistics of the $link$ collected in the current monitoring and $byte^{pre}_{(p,s)}(link)$ is the byte statistics of the $link$ collected in the previous monitoring iteration.

After the calculation of the dominant port utilization ($U_{(p,s)}$), the monitoring active port ($P_{ma}(p, s)$) may remain current monitoring section (S1) or be assigned to another monitoring section according to its dominant port utilization. The monitoring section assigned the monitoring active port is determined from Eq. 6.

$$\begin{aligned} & P_{ma}(p, s) \subset \mathrm{S1}, \qquad if\ U_{p,s} \ < \ thold_{\mathrm{S1}} \\ & P_{ma}(p, s) \subset \mathrm{S}i, \quad if\ thold_{\mathrm{S}(i-1)} \leq U_{p,s} < thold_{\mathrm{S}i}, \\ & \qquad\qquad i = 2, 3, \cdots, N_{\mathrm{S}i} \end{aligned} \tag{6}$$

If the monitoring active port remains the current monitoring section, it is continuously monitored in the current monitoring section (S1) with its monitoring interval ($intvl_{\mathrm{S1}}$). The tuple (2,2) in Fig. 3 represents this case. If the monitoring active port is moved to another monitoring section, there are two methods to estimate the port throughput. The first method is a seamless estimation method that can estimate the port throughput using Eq. 7 only when the port is moved to another monitoring section.

$$\begin{aligned} thp_{(p,s)}(link) =& 8 * \frac{(byte^{cur}_{(p,s)}(link) - byte^{pre}_{(p,s)}(link))}{t_{new} - t_{pre}}, \\ & link = T_x \ or \ R_x \end{aligned} \tag{7}$$

t_{new} and t_{pre} indicate the time to collect byte statistics at the previous monitoring section and the new monitoring section, respectively. After the byte statistics are measured in the new monitoring section, the port throughput is estimated using Eq. 3. Another method to estimate the port throughput is a reset method that initializes throughput estimation. This method is designed for off-the-shelf SDN switches which cannot estimate byte statistics with a short period. In the reset method, byte information ($byte^{cur}_{(p,s)}(link)$, and $byte^{pre}_{(p,s)}(link)$) are simply reset: the tuple (1,3) in Fig. 3 is the case. In consequence, the initial byte collection should be performed as with the first monitoring whenever a monitoring active port moves into a new monitoring section. After the first monitoring, port throughput and dominant port utilization are estimated by the new monitoring section (Si) with its monitoring interval ($intvl_{\mathrm{S}i}$) like as the tuple (3,4) in Fig. 3. The overall procedure of the adaptive section-based port monitoring is presented in

Algs. 3 and 4.

Algorithm 3 Assigning monitoring active ports to monitoring sections

```
1: for $port$ in $monitoring\_active\_ports$ in Alg. 2 do
2:     if $port$ is NOT in monitoring sections then
3:         S1.append ($port$)
4:     end if
5: end for
```

Algorithm 4 Adaptive section-based port monitoring

```
   Operation in each monitoring section i
1: $ports$ = $monitoring\_active\_ports$ in Alg. 2
2: while True do
3:     for $port$ in $ports$ do
4:         request_port_statistics ($port$)
5:         wait ($intvl_{Si}$)
6:     end for
7: end while

   Below operation is triggered when the port statistics message of a monitoring
   active port ($P_{ma}(p, s)$) arrives at SDN controller
8: port_statistics ($p, s$) = Fetch port statistics of $P_{ma}(p, s)$ from the response
                                 message.
9: if there are no port statistics information of $P_{ma}(p, s)$ in SDN controller then
10:     $byte^{cur}_{(p,s)}(T_x, R_x)$ = $Tx\_byte$ and $Rx\_byte$ in port_statistics ($p, s$)
11: else
12:     $byte^{pre}_{(p,s)}(T_x, R_x)$ = $byte^{cur}_{(p,s)}(T_x, R_x)$
13:     $byte^{cur}_{(p,s)}(T_x, R_x)$ = $Tx\_byte$ and $Rx\_byte$ in port_statistics ($p, s$)
14:     S$i$ = monitoring section number of $P_{ma}(p, s)$
15:     if seamless estimation method AND the first monitoring at S$i$ then
16:         $t_{new}$ = current time
17:         $thp_{(p,s)}(T_x, R_x) = 8 * \frac{(byte^{cur}_{(p,s)}(T_x, R_x) - byte^{pre}_{(p,s)}(T_x, R_x))}{t_{new} - t_{pre}}$
18:     else
19:         $thp_{(p,s)}(T_x, R_x) = 8 * \frac{(byte^{cur}_{(p,s)}(T_x, R_x) - byte^{pre}_{(p,s)}(T_x, R_x))}{intvl_{Si}}$
20:     end if
21:     if $thp_{(p,s)}(T_x) > thp_{(p,s)}(R_x)$ then
22:         $thp^{dmnt}_{(p,s)} = thp_{(p,s)}(T_x)$
23:     else
24:         $thp^{dmnt}_{(p,s)} = thp_{(p,s)}(R_x)$
25:     end if
26:     $U_{(p,s)} = thp^{dmnt}_{(p,s)}$ / $speed_{(p,s)}$
27:     S$k$ = find a new monitoring section using Eq. 6.
28:     if S$i \neq$ S$k$ then
29:         remove $P_{ma}(p, s)$ from S$i$
30:         if seamless estimation method then
31:             $t_{pre}$ = current time
32:         else: reset method
33:             reset $byte^{cur}_{(p,s)}(T_x, R_x)$ and $byte^{pre}_{(p,s)}(T_x, R_x)$
34:         end if
35:         S$k$.append($P_{ma}(p, s)$)
36:     end if
37: end if
```

4) Management of monitoring active ports: Because active ports are determined by flow rules, a change of flow rules directly affects active ports: a new active port can be generated or existing active ports become invalid. Thus, active ports should be properly updated according to a change of flow rules. In order to update active ports, APAM performs some operations which are denoted by the management operations in the rest of paper. Figure 4 shows the logical structure of APAM. When APAM is executed, APAM creates monitoring sections and initially performs management operations. The monitoring sections are continuously running as a background function until APAM terminates. The management operations can be performed as an event triggered by APAM or other functions in a SDN controller in order to update active ports.

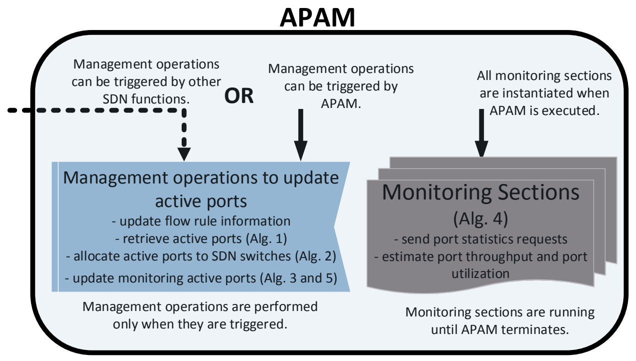

APAM
Management operations can be triggered by other SDN functions.
OR
Management operations can be triggered by APAM.
All monitoring sections are instantiated when APAM is executed.
Management operations to update active ports
- update flow rule information
- retrieve active ports (Alg. 1)
- allocate active ports to SDN switches (Alg. 2)
- update monitoring active ports (Alg. 3 and 5)
Monitoring Sections (Alg. 4)
- send port statistics requests
- estimate port throughput and port utilization
Management operations are performed only when they are triggered.
Monitoring sections are running until APAM terminates.

Fig. 4. Logical structure of APAM.

The details of the management operation are described as follows.

Renewing flow rule information in SDN controller is the first step to update active ports. To be independently operated from other SDN functions, APAM can regard the flow rules information collected and updated by other functions in the SDN controller as up-to-date information, which is called the built-in update method in this paper. On the other hand, for quick adaptation to changing flow rules, APAM can collect flow rule information by itself whenever the SDN controller detects the events which add or modify flow rules (such as "flow_mod"), which is called the dedicated update method. This way enables it to instantly update flow rules only when the flow rule is necessary to be updated although an additional internal process is required: an SDN controller reports flow change events to APAM. In Section IV.E, the impact of update methods for flow rules on updating active ports is explicitly discussed. After the update of flow rules, both active ports and the selection of active network ports are also updated. If there is a change in active network ports, all active network ports (including existing active network ports) are re-assigned to SDN switches using the same selection procedure explained in Section III.B.2. Lastly, APAM updates active ports in monitoring sections according to the update of monitoring active ports. If there is a new monitoring active port, APAM adds the monitoring active port to the monitoring section 1 (Alg. 3). On the contrary, if a monitoring active port monitored in one of monitoring sections changes into non-active port, the monitoring active port is removed from the monitoring section and all information related to the monitoring active port is also deleted. The discarding procedure for invalid active ports is described in Alg. 5.

Algorithm 5 Discarding invalid monitoring active ports from monitoring sections

```
1: $ports\_in\_monitoring\_sections$ = Fetch monitoring active ports from moni-
   toring sections
2: for $port$ in $ports\_in\_monitoring\_sections$ do
3:     if $port$ is NOT $monitoring\_active\_ports$ in Alg. 2 then
4:         remove $port$ from monitoring sections
5:         remove the $port$'s information
6:     end if
7: end for
```

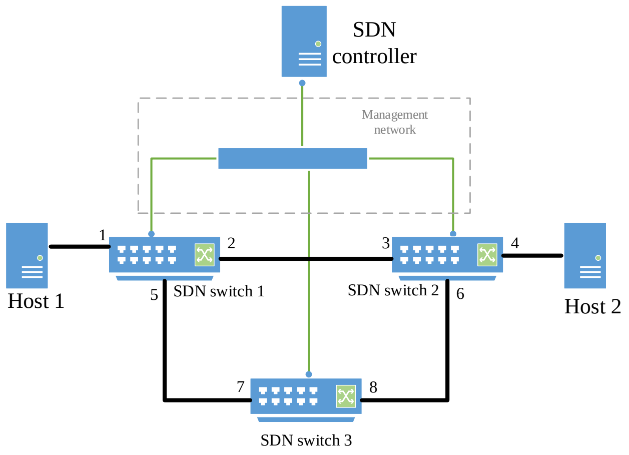


Fig. 5. Measurement environment: network topology.

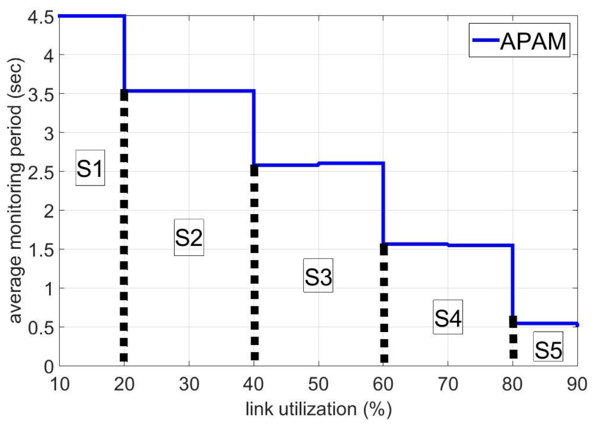


Fig. 6. Average port monitoring period versus link utilization.

Another consideration for APAM is monitoring overhead according to the number of active ports. APAM manages active ports individually; i.e. it generates a request message to gather traffic statistics from each active port separately. Therefore, APAM may have higher monitoring overhead than the conventional monitoring method. In order to restrict the increase of monitoring overhead, APAM operation can switch its operation to the conventional monitoring method when the monitoring overhead of APAM operation exceeds a certain threshold. In this paper, the number of monitoring messages per second is used as an indicator of monitoring overhead. The monitoring overhead of the conventional monitoring method which is the number of network ports (N_{ports}) plus 1 (monitoring request message). In the case of APAM, the monitoring overhead depends on the number of active terminal ports (N_{at}) and the number of active network ports (N_{an}). Active terminal ports require not only N_{at} monitoring request messages but also N_{at} messages for monitoring information (reply messages). In the case of active network ports, half of N_{an} messages are required for each monitoring information messages and monitoring request messages. Thus, the monitoring overhead of APAM is 2*N_{at} + N_{an}. As a result, the switching operation is performed in the case of Eq. 8.

$$\frac{N_{ports}+1}{intvl_{conv}} < \frac{2*N_{at}+N_{an}}{intvl_{avg}} \tag{8}$$

Note that $intvl_{conv}$ indicates the monitoring interval of the conventional monitoring method and $intvl_{avg}$ represents the average monitoring interval of APAM because the monitoring interval of each monitoring port is different in APAM.

IV. Measurement Results

A. Measurement Environment

In this section, we evaluate the performance of APAM using measurement taken using our SDN testbed. We implement APAM in Ryu which is a component-based SDN framework [23] and APAM is executed as a basic function of the SDN controller. In this measurement, the number of monitoring sections (N_{Si}) is 5 and $intvl_{max}$ and $intvl_{min}$ are 5.5 sec and 0.5 sec, respectively. APAM utilizes flow rule information and the whole network topology information of the SDN testbed which are collected by other basic functions in SDN controller. We constructed a network topology using hardware SDN switches, an SDN controller, and end hosts. Two Mellanox switches (SN2100) and one Corsa switch (DP2100) which support OpenFlow 1.3 are applied as SDN switches. The SDN controller and the end hosts are operated on PC (i5-2500U @ 3.3GHz with 8GB of RAM) running 64-bit Ubuntu 14.04. Due to the off-the-shelf SDN switches, the reset method is selected to estimate a port throughput. The network topology is shown in Fig. 5. Each host is connected to one of SDN switches and all SDN switches are connected with the SDN controller through the management network which is used for control messages. All network interfaces in this topology have same bandwidth (1- Gbps port) and UDP traffic is used as a network traffic. Measurements are repeated 10 times. The parameters for the measurements are summarized in Table I.

TABLE I
MEASUREMENT PARAMETERS

APAM parameter	value
N_{Si}	5
$intvl_{max}$	5.5 sec
$intvl_{min}$	0.5 sec
port estimation method	reset method
topology parameter	**value**
SDN switch model	Mellanox SN2100 & Corsa DP2100
SDN controller	Ryu
OpenFlow version	1.3
link bandwidth	1 Gbps
maximum segment size	1500 bytes

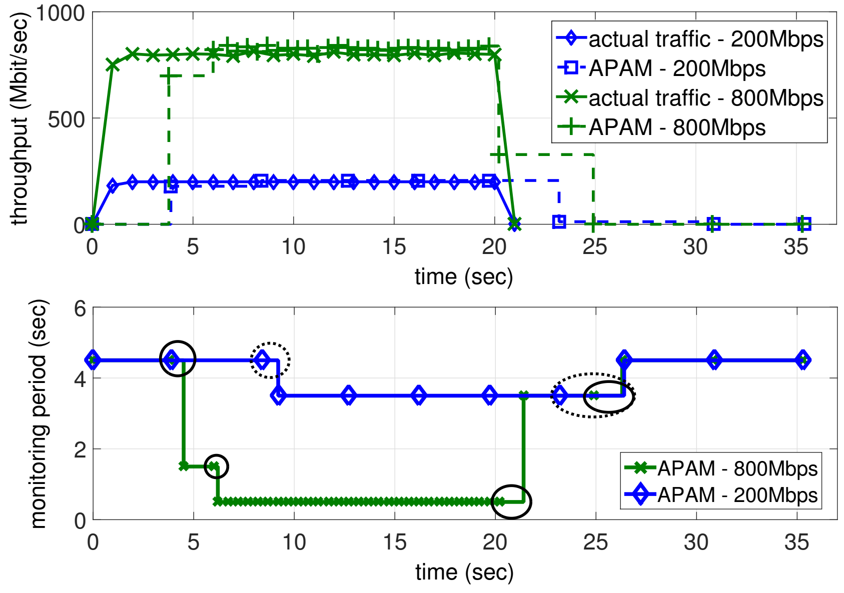


Fig. 7. Adaptive capacity of APAM according to port throughput.

B. Performance Evaluation in Active-port Aware Monitoring

Figure 6 shows the average port monitoring period of APAM for a switch port according to its port (link) utilization. In this measurement, UDP traffic is generated by a client (host 1) and the traffic is forwarded to a server (host 2) via switch 1 and switch 2 as shown in Fig. 5. The monitoring period of the port 2 on switch 1 ($P_{ma}(2,1)$) is recorded as a monitoring active port during the traffic transmission. For each measurement, the client generates UDP traffic for 100 sec and measurements are repeated 10 times for a specific link utilization. The results show that the average monitoring period decreases with increasing the link utilization. This is because the observed port ($P_{ma}(2,1)$) is assigned to a higher monitoring section (Si) whenever the link utilization of the port exceeds the monitoring threshold of its own monitoring section, which makes the port have a shorter monitoring period. Thus, this result indicates that APAM can adapt an active port's monitoring period according to a port's link utilization.

Figure 7 shows the adaptive capacity of APAM according to a port throughput variation. In this measurement, UDP traffic is generated from 0 sec to 20 sec and the traffic is forwarded from the client to the server via switch 1 and switch 2 as shown in Fig. 5. The port 2 ($P_{ma}(2,1)$) is selected to record both its port throughput and its monitoring period. The upper subgraph shows the port throughput results of the port 2. The solid lines indicate the actual traffic results monitored at the server; the solid lines with diamonds represent the results when the port throughput is 200 Mbps and the solid lines with crosses indicate the results when the port throughput is 800 Mbps. Dotted lines indicate the traffic results monitored by APAM; the dotted lines with squares represents the monitored traffic when the port throughput is 200 Mbps and the dotted lines with plus signs correspond to the case that the port throughput is 800 Mbps. The lower subgraph shows the monitoring period of APAM for the port 2. The solid lines with diamonds indicate the monitoring period when the port throughput is 200 Mbps and the solid lines with crosses represent the results when the port throughput is 800 Mbps.

The case when the port throughput is 800 Mbps is firstly described. In the upper subgraph, the results show that APAM recognizes the port traffic at 3.8 seconds although the traffic is generated from 0 seconds. This is because that there is no traffic before the generation of UDP traffic, which places the previous monitoring section in S1. The initial monitored port throughput by APAM is 699 Mbps. This inaccurate measurement is because the traffic is generated between monitoring periods. After the traffic detection, the monitoring active port is moved to S4 according to Eq. 6. After the monitoring active port is assigned to S4, there is a monitoring loss time which is the interval between the time (3.8 sec) that the port is moved to S4 and the time (4.5 sec) that the port monitoring starts at S4. This loss time is generated whenever the monitoring of active port is moved to a new monitoring section because monitoring sections are not synchronized with each other. This generates a latency time to start the monitoring process at a new monitoring section. All these monitoring loss time events are marked by solid circles in the lower subgraph and, during these monitoring loss times, the monitoring period is represented by the previous monitoring period. Because the calculated throughput is 822 Mbps at S4 (at 6 sec), the monitoring active port is moved to S5 and the following monitoring loss time is 0.2 seconds. After the port traffic disappeared, APAM detects that port throughput is changed to 328 Mbps at 20.2 seconds, which makes the monitoring port moved to S2. After the monitoring loss time (1.2 sec), the monitored throughput becomes 0 Mbps from 24.9 seconds, which makes the monitoring active port moved to S1.

The case that the port throughput is 200 Mbps is now

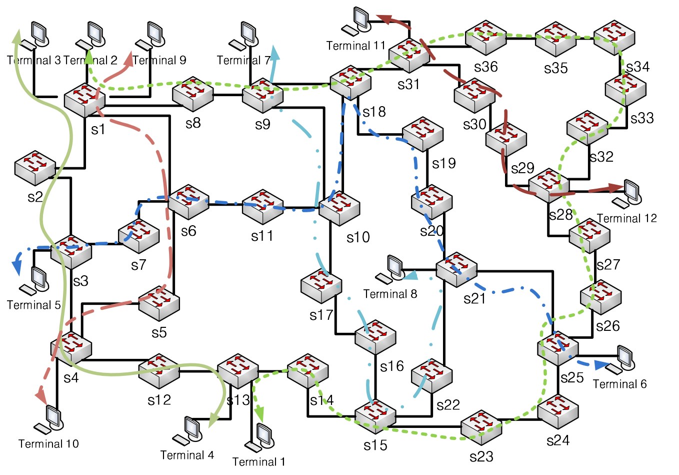


Fig. 8. Virtual network topology.

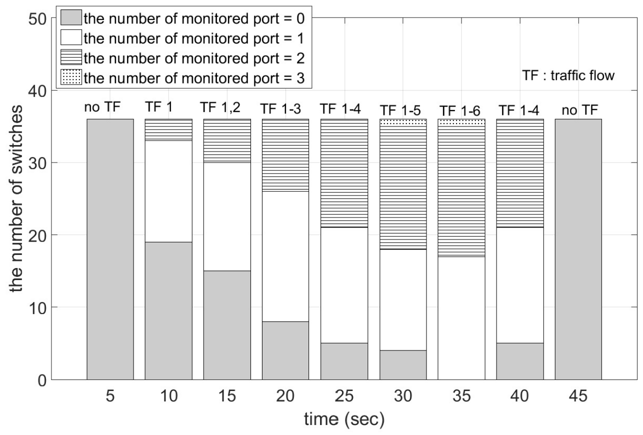


Fig. 9. The number of switches participating in monitoring according to traffic flows.

described. In this case, APAM recognizes the port traffic at 3.9 seconds and the monitored port throughput and the port utilization are 179 Mbps and 0.179, respectively. Because the port utilization does not exceed $thold_{S1}$ (0.2), the port is still monitored at S1. At the next monitoring operation (8.4 sec), the measured port throughput is 206 Mbps, which makes the monitoring active port moved to S2. After the change of the monitoring section, the monitoring loss time is 1.2 seconds. For the case that the port throughput is 200 Mbps, monitoring loss time events are marked by dotted circles. After the client stops to generate traffic after 20 seconds, APAM detects this traffic change at 23.2 seconds and the measured traffic is 12 Mbps, which makes the monitoring active port moved to S1.

C. Adaptability and Monitoring Overhead of APAM

This section evaluates the adaptability and monitoring overheads of APAM working on top of a more complex network topology. An emulated network is constructed by using Mininet [24] that is a framework for executing the real kernel and application code. The network topology consists of one SDN controller, 36 SDN switches with a total of 100 ports, and 12 terminals (TRMs) as shown in Fig. 8, which is based on the Internet2 Network Infrastructure Topology [25]. All network interfaces in the emulated network have 100 Mbps bandwidth and UDP traffic is used as network traffic. To utilize the network links, 6 traffic flows (TFs) are generated by using 12 terminals, which is indicated in Fig. 8 by lines with arrows, and the details of the traffic flows are shown in Table II.

Figure 9 shows the number of switches participating in monitoring according to traffic flows in the emulated network. Each bar graph depicts the number of switches corresponding to the number of monitored ports. In the initial state (no TF), none of the switches have any monitored port. From 10 seconds to 35 seconds, the traffic flows are added every 5 seconds. Following the traffic flows, the proportion of switches corresponding to the number of monitored ports is changed while the total number of switches is the same. After that, the traffic flows are deleted again. In order to quickly adapt changing flow rules, the dedicated update method is applied in APAM. The traffic flows (TFs) are presented in Table II. The results show that the number of monitored ports increases (decreases) whenever a traffic flow is added (deleted), which means active ports are detected and dynamically monitored depending on the traffic flows. Moreover, the monitoring overhead (monitored ports) is well dispersed to the switches: all switches except for one have one or two monitored ports in the measurement scenario. Although there are 13 switches in the network that have more than 3 ports, the results show that only one switch has three monitored ports in the cases that the 5 (TF1-5) and 6 (TF1-6) traffic flows are added. There are two reasons for the results. The first is that APAM reduces the number of monitored ports by monitoring a network link using one active port. The other is APAM tries to allocate a monitoring link to the switches that have less active ports than other switches as described in Section III.B.2. The switch status related to the number of monitored ports is shown in Table III.

Figure 10 shows the monitoring overhead according to the ratio of active ports in the emulated network. The ratio of active ports is changed by adding the traffic flows (TFs) in Table II. The monitoring overhead indicates the total required messages to collect monitoring information for the whole network. In the measurement, it is assumed that APAM and

TABLE II
TRAFFIC FLOWS

Traffic Flow (TF)	Route
TF1	TRM1-(s13-s15)-(s23-s28) -(s32-s36)-s32-s18-s9-s8-s1-TRM2
TF2	TRM3-(s1-s4)-s12-s13-TRM4
TF3	TRM5-s3-s7-s6-s11-s10-s18-s20 -s21-s25-TRM6
TF4	TRM7-s9-s10-s17-s16-s15-s22-s21 -TRM8
TF5	TRM9-s1-s6-s5-s4-TRM10
TF6	TRM11-s31-s30-s29-s28-TRM12

TABLE III
SWITCHES CORRESPONDING TO THE NUMBER OF MONITORED PORTS

	switches corresponding to the number of monitored ports			
TF	the number of monitored port = 0	the number of monitored port = 1	the number of monitored port = 2	the number of monitored port = 3
TF0	s1-s36	-	-	-
TF1	s2-s7, s10-s12, s16,-s17, s19,s20, s22, s23, s29-s31, s37	s1, s8, s13, s15, s18, s24-28, s32, s33, s35, s36	s9, s14, s34	-
TF1,2	s5-s7, s10, s11, s16, s17, s19, s20, s22, s23, s29-s31, s37	s3, s4, s8, s12, s15, s18, s24-s28, s32, s33, s35, s36	s1, s2, s9, s13, s14, s34	-
TF1-3	s5, s16, s17, s23, s29-s31, s37	s3, s6-s9, s11, s12, s15, s19, s20, s22, s24-s26, s28, s33, s35, s36	s1, s2, s4, s10, s13, s14, s18, s27, s32, s34	-
TF1-4	s5, s29-s31, s37	s3, s6, s7, s9, s12, s16, s17, s19, s20, s23, s25, s26, s28, s33, s35, s36	s1, s2, s4, s8, s10, s11, s13-s15, s18, s22, s24, s27, s32, s34	-
TF1-5	s29-s31, s37	s4, s7, s9, s16, s17, s19, s20, s23, s25, s26, s28, s33, s35, s36	s2, s3, s5, s6, s8, s10-s15, s18, s24, s27, s32, s34	s1
TF1-6	-	s4, s7, s9, s16, s17, s20, s23, s25, s26, s28, s31-s33, s35, s36, s37	s2, s3, s5, s6, s8, s10-s15, s18, s19, s22, s24, s27, s30, s34	s1

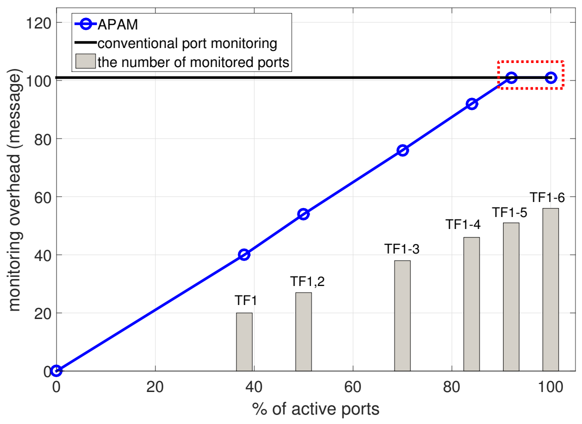


Fig. 10. Monitoring overhead according to the ratio of active ports.

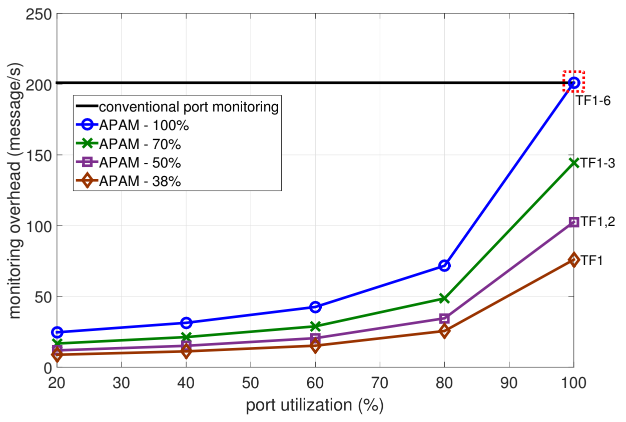


Fig. 11. Monitoring overhead according to the port utilization of active ports.

the conventional port monitoring have the same monitoring interval. The solid lines with circles indicate the monitoring overhead of APAM. For comparison with APAM, the solid lines show the monitoring overhead of the conventional port monitoring which monitors all switches ports. The bar graph depicts the number of monitored ports in the emulated network. In the case that the ratio of the active ports in the network is low, APAM can have less monitoring overhead than the conventional port monitoring, thanks to the reduction of the number of monitored ports: unlike the conventional port monitoring which monitors whole network ports, the reduction is achieved in APAM by monitoring only active ports as well as monitoring a network link using one active port. On the contrary, the monitoring overhead of APAM exceeds that of the conventional monitoring in the TF1-5 and the TF1-6 cases (marked by dotted square). It is due to the additional overhead of APAM in order to individually monitor active ports described in Section III.B.4. In these cases, APAM switches its operation to the conventional monitoring method, which makes the monitoring overhead of APAM the same as that of the conventional monitoring method.

Figure 11 shows the monitoring overhead according to the port utilization of active ports in the emulated network. For simplicity, all active ports are assigned the same port utilization. In these results, the monitoring overhead represents the required messages to collect monitoring information for the whole network per second (message/s). The APAM's results are represented by solid lines with marks and each mark represents the ratio of active ports in the whole network. The ratio of active ports is changed by the traffic flows (TFs) in Table II. For comparison with APAM, the solid lines show the monitoring overhead of the conventional port monitoring, and its monitoring period is 0.5 sec which is the shortest monitoring period in APAM's configuration. The results show that the monitoring overhead of APAM increases with the growth of port utilization. It is because APAM shortens the monitoring period of active ports to improve their monitoring accuracy according to the increase of traffic at active ports. When APAM requires more monitoring overhead than the conventional monitoring (marked by dotted square), APAM switches its operation to the conventional monitoring method. However, in the case that either the ratio of active ports or their port utilization is not a high proportion, APAM can have less monitoring overhead than the conventional monitoring owing to the reduction of the number of monitored ports or increasing monitoring period of active ports.

In Fig. 12, we investigated the effect of traffic dynamics on the monitoring overhead as well as the detection of congested links. In the measurement, the emulated network is used and all network ports are active ports by traffic flow rules (TF1-6). All network ports are underutilized at the initial state and then UDP traffic (90Mbps) is applied to the TF2 from 15 seconds to 45 seconds, which makes the 7 links related to the TF2 congested. The upper subgraph indicates the monitoring overhead according to the change of traffic flows and the lower subgraph represents the number of monitored ports in the monitoring sections. The solid lines with marks indicate the results corresponding to the monitoring sections and the solid lines show the total results of the monitoring sections. Before the UDP traffic is generated, all active ports stay in S1, which produces minimum monitoring overhead. After the UDP traffic is generated, APAM detects the traffic and assigns the related monitored ports to S5, allowing a

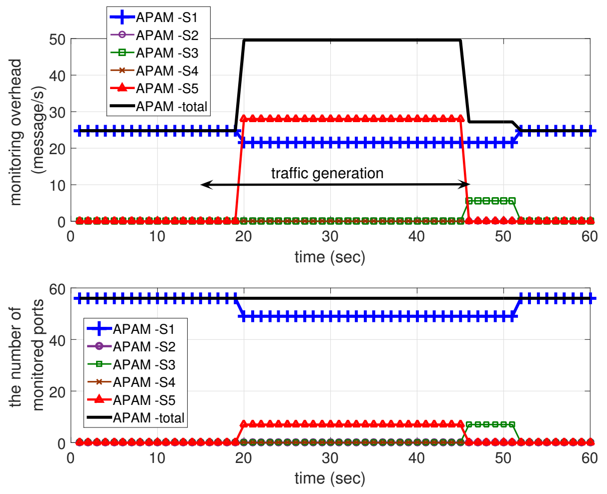


Fig. 12. Monitoring overhead and detection of congested links according to the traffic dynamics.

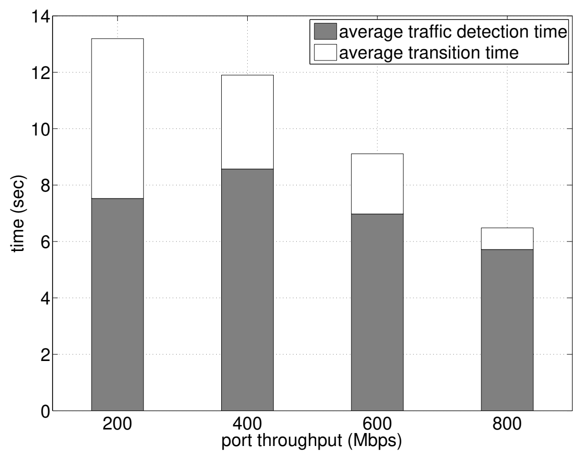


Fig. 13. Average required time to be stable after moving to a new monitoring section from the lowest monitoring section.

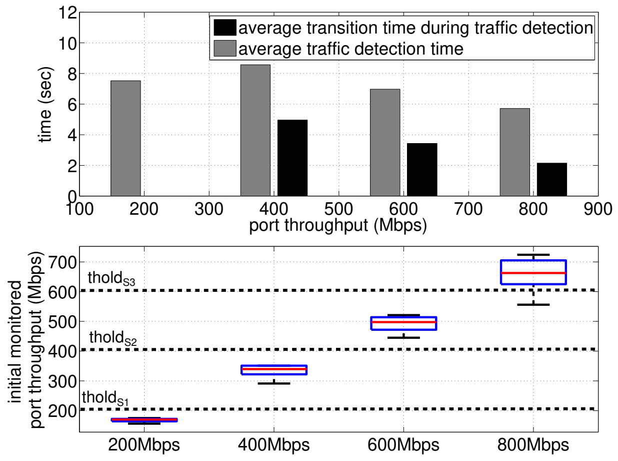


Fig. 14. Average traffic detection time and the initial monitored port throughput according to port throughput.

network operator to recognize the congested link. Furthermore, the upper subgraph shows that the proportion of monitoring overhead is different according to the traffic in the network links (ports). More than half of the total monitoring overhead (56%) is generated by the congested links (ports) during the traffic transmission although the ratio of congested links (port) in the network is only 12.5%. This means network and computing resources can be primarily used to monitor the congested ports. Lastly, the congested ports are moved to S1 after the traffic disappears, and the monitoring overhead is reduced to the minimum monitoring overhead.

D. Initialization Time after Moving to a New Monitoring Section in APAM

In this chapter, we investigate the initialization time of APAM when an active monitoring port is moved to a new monitoring section. The effect of amount of port traffic is firstly considered and the effect of background traffic is then examined.

In Fig. 13, we investigated the effect of port throughput on the time required to be stable after moving to a new monitoring section from the lowest monitoring section (S1). The measurement environment is the same as in Fig. 6. UDP traffic is generated by a client (host 1) and forwarded to a server (host 2) via switch 1 and switch 2. The monitoring period of the port 2 ($P_{ma}(2,1)$) on switch 1 is recorded as a monitoring active port during the traffic transmission. The generated traffic varies between 200 Mbps to 800 Mbps and measurements are repeated 10 times for a specific throughput. The gray bar graph depicts the average traffic detection time which is the time interval between the traffic generation time and the time when APAM accurately monitors a port throughput (utilization). In other words, this detection time represents the required time to move a monitoring active port to a proper monitoring section in the case that the port utilization is greater than $thold_{S1}$. The white bar graph indicates the average transition time which is the required time to re-estimate the port throughput (utilization) after the monitoring active port is moved to a proper monitoring section. This transition time equals to the sum of the monitoring loss time and the required time to calculate a port throughput and a port utilization at the monitoring section. The sum of both the traffic detection time and the transition time is the required time to be stable after moving to a new monitoring section. The results show that the average traffic detection time tends to decrease with increasing the port throughput. This is because, after the initial throughput detection, the port is generally moved to higher monitoring section as the port throughput increases. The detailed results of the traffic detection time are covered in Fig. 14. The average transition time also decreases with increasing the port throughput. This is a natural consequence of shortening the monitoring period; the monitoring loss time decreases in proportion to the monitoring period, and also the required time to calculate a port throughput and a port utilization is equal to the monitoring period.

Figure 14 shows the average detection time and the initial monitored port throughput according to port throughput. The measurement scenario is the same as in Fig. 13. The lower

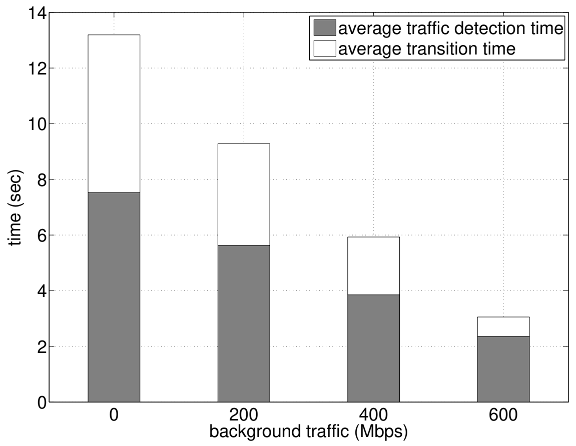


Fig. 15. Average required time to be stable after moving to a new monitoring section according to background traffic.

subgraph shows the initial monitored port throughput at S1 by using box-and-whisker plots. In the upper subgraph, the gray bar graph depicts the average traffic detection time which is the same results in Fig. 13. The black bar graph indicates the average transition time during the traffic detection time. This transition time is the required time that a monitoring active port is assigned to a proper monitoring section during the traffic detection time. It is generated in the case that a monitoring active port is moved and monitored at a wrong monitoring section before APAM accurately monitors a port throughput (utilization): the transition time occurs due to an inaccurate port throughput estimation which usually happens at the initial throughput estimation. The results show that there is no transition time during traffic detection period when the port traffic is 200 Mbps because the monitoring active port still stays in S1 based on the initial monitored port throughput as shown in the lower subgraph. In the cases that the port throughput is greater than 400 Mbps, the transition time is generated due to the inaccurate initial port throughput as shown in the lower subgraph. The inaccurate throughput estimation leads the monitoring active port to be placed in a wrong monitoring section, which causes resetting its byte information and performing an initial byte collection again when it moves to the right monitoring section. For this reason, the average traffic detection time when the port traffic is 400 Mbps is longer than the 200 Mbps case.

In Fig. 15, we investigated the effect of background traffic on the required time to be stable after moving to a new monitoring section. The measurement environment and the representation of the results are the same as in Fig. 13. The measurement scenario is as follows. There is background traffic which varies between 0 Mbps and 600 Mbps through the monitored and recorded port ($P_{ma}(2,1)$) and this background traffic is firstly generated. After that, the client traffic is generated and is fixed to 200 Mbps. The results show that both the average traffic detection time and the average transition time for the client traffic decrease with increasing the background traffic. This is because the client traffic is monitored with

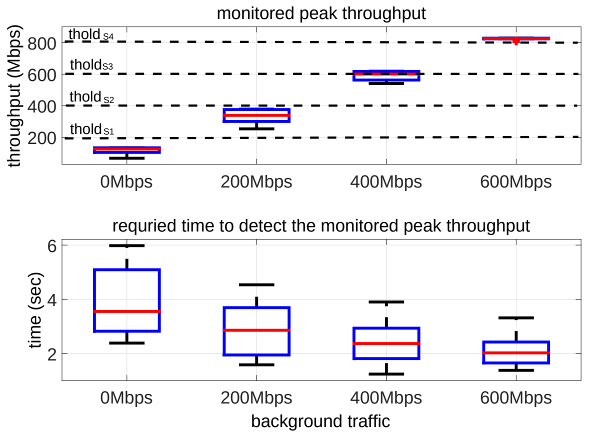


Fig. 16. Monitored peak throughput and required time to detect the measured peak throughput.

shorter monitoring period as the background traffic grows: larger background traffic makes a monitoring active port move to a higher monitoring section.

E. Robustness in Active-port Aware Monitoring

In this section, we consider the robustness of APAM for spike traffic according to background traffic. In this measurement, background traffic at the monitored and recorded port ($P_{ma}(2,1)$) varies between 0 Mbps and 600 Mbps. With the background traffic, spike traffic is generated for three seconds and its throughput is fixed to 200 Mbps. Measurements are repeated 10 times.

Figure 16 shows the monitored peak throughput by APAM and the required time to detect the measured peak throughput according to background traffic. The results are represented by box-and-whisker plots. The upper subgraph indicates the monitored peak throughput by APAM and these results show that the accuracy of the monitored peak throughput increases as increasing the background traffic. In the cases that the background traffic is equal to or less than 400 Mbps, APAM cannot detect the spike traffic accurately. This is because APAM can only monitor the partial spike traffic. When the background traffic is 400 Mbps, although APAM can accurately detect spike traffic in some cases, it is not guaranteed to monitor the spike traffic. When the background traffic is 600 Mbps, APAM detects the peaked throughput (800 Mbps) because the monitoring interval is sufficiently short (1.5 seconds) compared to the spike traffic generation time. The lower subgraph in Fig. 16 shows the time interval between the time to generate spike traffic and the time to detect the monitored peak throughput as shown in the upper subgraph, which is the required time to detect the monitored peak throughput. The results show that the required time decreases with increasing of the background traffic. Except for the case that the background traffic is 600 Mbps, the other cases have some results that the required time is less than the monitoring interval. These results occur when the throughput monitoring results which partially monitor the spike traffic become the peak throughput. In the case that the background traffic is

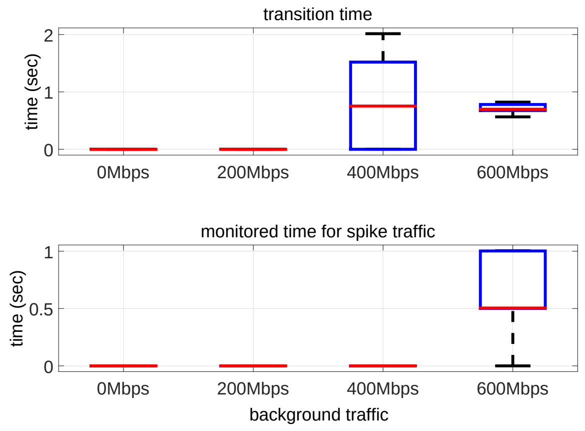


Fig. 17. Transition time and monitored time for spike traffic.

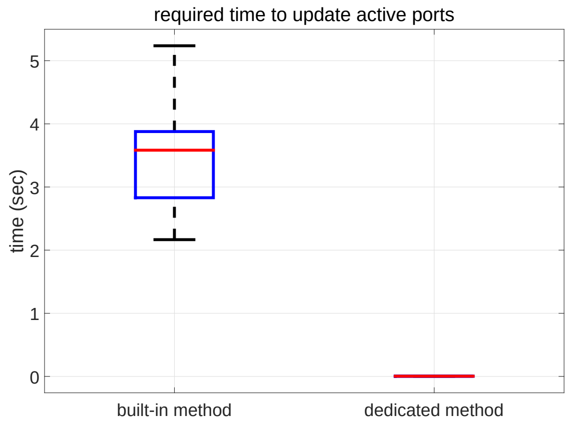


Fig. 18. Required time to update active ports.

600 Mbps, the required time is greater than the monitoring period and the peaked throughput is detected as shown in the upper subgraph. According to both these results, it is shown that APAM can monitor the spike traffic when the monitoring interval is enough small due to the increase of the background traffic.

In Fig. 17, the transition time which is the required time to re-estimate the port throughput (utilization) due to the spike traffic and the monitoring performance to measure the spike traffic are considered according to the background traffic. The results are also represented by box-and-whisker plots. The upper subgraph shows the transition time caused by the spike traffic. When the background traffic is less or equal to 200 Mbps, there is no transition time for the spike traffic because the monitored traffic is less than the monitoring threshold due to the inaccurate throughput measurement as shown in the upper subgraph in Fig. 16. In the case that the background traffic is 400 Mbps, the transition time occurs or not according to the monitored peak throughput results. This is because, APAM partially monitors the spike traffic according to the start time of its monitoring operation. When the background traffic is 600 Mbps, the transition time has quite a stable value (0.7) because APAM can detect the spike traffic as shown in the monitored peak throughput results shown in Fig. 16. The lower subgraph in Fig. 17 shows the monitored time for the spike traffic according to the background traffic. When the background traffic is less than 600 Mbps, there is no monitoring time. In the case that the background traffic is 400 Mbps, although APAM can detect the spike traffic in some results, it cannot monitor the spike traffic due to the transition time shown in the upper subgraph: after the spike traffic is detected and the transition process is performed at the new monitoring section, the spike traffic has already disappeared. When the background traffic is 600 Mbps, APAM monitors the spike traffic for 0.5 second on average. This is because APAM not only quickly detects the spike traffic but also has a relatively short transition process at the new monitoring section. Consequently, the spike traffic still remains after the transition time and this remained traffic is monitored by APAM.

In summary, the traffic sensitivity in APAM varies according to background traffic. APAM improves its monitoring performance according to the increase of background traffic, which is the intended operation to meticulously monitor a link status as the link become congested.

Figure 18 shows the required time to update active ports according to the flow rule update method. In this measurement, new flow rules are added by an SDN application during APAM operation, which generates a new active port. The results are represented by box-and-whisker plots. The built-in update method uses the flow rule information which is collected and updated by a built-in function in SDN controller. This function collects and updates the flow rules every 5.5 sec. The dedicated update method can instantly update the active ports when the flow rules are updated. In this method, the flow rule update is notified to APAM by an internal signal generated by the SDN controller. The results show that APAM with the built-in update method requires relatively large time (3.7 sec on average) to update active ports due to the update interval of the built-in function. Also, this required time fluctuates because it varies according to both the start time to change the flow rules and the time updated by the built-in function. However, the dedicated update method can update active ports quickly, which makes APAM instantly adjust the change of flow rules. This is because the change of flow rules is detected by the SDN controller and this change is instantly notified to APAM, which significantly improve the flow rule update speed.

V. Conclusion

In this paper, we present a novel monitoring mechanism for SDN to mitigate the trade-off of traffic monitoring as a macroscopic monitoring function. The key novelty is that APAM retrieves active ports that are the switch ports utilized by current flow rules and dynamically monitors active ports according to their port utilization, which is able to focus network and processing resources on primarily utilized ports. Through the measurement in our testbed, we determined that APAM can not only reduce the number of switch ports to monitor a whole network but also adjust monitoring overhead and performance according to port congestion level although it may require transition time for some off-the-shelf SDN switches when active monitoring ports are moved to new monitoring sections. In addition, the measurement showed that APAM is sufficiently robust for spike traffic and reconfigures a monitoring topology according to a change of flow rules in a network. Finally, the simplicity and lightweight of APAM allows APAM can be applied as a practical network monitoring solution.

ACKNOWLEDGMENT

The authors would like to acknowledge the support from the University of Surrey's 5G Innovation Centre (5GIC) (http://www.surrey.ac.uk/5gic) members and NG-CDI (https://www.ng-cdi.org/) project for this work.

REFERENCES

[1] N. Mckeown, *How SDN Will Shape Networking*, Open Netw. Summit, Los Angeles, CA, USA, Oct. 2011.

[2] S. Schenker, *The future of networking the past of protocols*, Open Netw. Summit, Los Angeles, CA, USA, Oct. 2011.

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

[4] P. Bosshart et al., "P4: programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun.*, Rev., vol. 44, no. 3, pp. 87–95, 2014.

[5] H. Xu, Z. Yu, C. Qian, X.-Y. Li, L. Huang, "Minimizing flow statistics collection cost using wildcard-based requests in SDNs", *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3587-3601, Dec. 2017.

[6] A. Tootoonchian, M. Ghobadi, Y. Ganjali, "OpenTM: traffic matrix estimator for openflow networks" *in Passive and Active Measurement, Springer*, pp. 201-210, 2010.

[7] S. Chowdhury, M. F. Bari, R. Ahmed, R. Boutaba, "PayLess: A low cost network monitoring framework for software defined networks", in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, pp. 1-9, May. 2014.

[8] N. L. M. van Adrichem, C. Doerr, F. A. Kuipers, "Opennetmon: Network monitoring in openflow software-defined networks", in *Proc. IEEE Netw. Oper. Manage. Symp.*, pp. 1-8, 2014.

[9] Z. Su, T. Wang, Y. Xia, M. Hamdi, "FlowCover: Low-cost flow monitoring scheme in software defined networks",in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, pp. 1956-1961, Dec. 2014.

[10] Y. Zhang, "An adaptive flow counting method for anomaly detection in SDN", in *Proc. of the Association of Computing Machinery Intern. Conf. on Emerging Networking Experiments and Technologies (ACM CoNEXT)*, pp. 25-30, 2013.

[11] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, H. V. Madhyastha, "Flowsense: monitoring network utilization with zero measurement cost", in *Proc. 14th Intern. Conf. Passive and Active Meas.*, pp. 31-14, 2013.

[12] H. Xu, Z. Yu, C. Qian, X.-Y. Li, L. Huang, "Minimizing flow statistics collection cost using wildcard-based requests in SDNs", *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3587-3601, Dec. 2017.

[13] H. Tahaei, R. B. Salleh, M. F. Ab Razak, K. Ko and N. B. Anuar, "Cost Effective Network Flow Measurement for Software Defined Networks: A Distributed Controller Scenario", *IEEE Access*, vol. 6, pp. 5182-5198, 2018.

[14] G. Tangari, D. Tuncer, M. Charalambides, Y. Qi and G. Pavlou, "Self-Adaptive Decentralized Monitoring in Software-Defined Networks", *IEEE Trans. Netw. Service Manag.*, vol, 15, no. 4, pp. 1277-1291, Dec. 2018.

[15] J. Rasley et al., "Planck: Millisecond-scale monitoring and control forcommodity networks", in *Proc. ACM Conf. SIGCOMM*, pp. 407-418, 2014.

[16] J. Suh et al., "OpenSample: A Low-Latency Sampling-Based Measurement Platform for Commodity SDN", in *Proc. IEEE Int'l. Conf. Distrib. Comp. Sys.*, pp. 228-37, Jul. 2014.

[17] S. Yoon, T. Ha, S. Kim, and H. Lim, "Scalable Traffic Sampling Using Centrality Measure on Software-Defined Networks," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 43-49, Jul. 2017.

[18] L. Castanheira, R. Parizotto and A. E. Schaeffer-Filho, "FlowStalker: Comprehensive traffic flow monitoring on the data plane using P4", *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1-6, May 2019.

[19] R. Hark, D. Bhat, M. Zink, R. Steinmetz and A. Rizk, "Preprocessing Monitoring Information on the SDN Data-Plane using P4," *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Dallas, TX, USA, pp. 1-6, 2019.

[20] B. Guan and S. Shen, "FlowSpy: An Efficient Network Monitoring Framework Using P4 in Software-Defined Networks," *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, Honolulu, HI, USA, pp. 1-5, 2019.

[21] A. Yassine, H. Rahimi and S. Shirmohammadi, "Software defined network traffic measurement: Current trends and challenges," *IEEE Instrumentation & Measurement Magazine*, vol. 18, no. 2, pp. 42-50, Apr. 2015.

[22] ONF TR-521, "SDN Architecture", issue 1.1, Feb. 2016.

[23] *RYU SDN Framework Community.* Accessed: Nov. 15, 2018. [Online]. Available: http://osrg.github.io/ryu/

[24] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," *in Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw.*, 2010, Art. ID 19.

[25] *Internet2.* Accessed: Oct. 25, 2020. [Online]. Available: https://internet2.edu/