Available online at www.ijpe-online.com

International Journal of Performability Engineering

Vol. 13, No. 6, October 2017, pp. 937-944
DOI: 10.23940/ijpe.17.06.p14.937944

AdaRate: A Rate-Adaptive Traffic Measurement Method in Software Defined Networks

Jixing Tang[a,b], Yue Zhang[a,b,*], Yan Li[a,b]

[a]*MoE Engineering Research Center for Software/Hardware Co-Design Technology and Application, East China Normal University, Shanghai 200062, China*
[b]*Shanghai Key Lab for Trustworthy Computing, East China Normal University, Shanghai 200062, China*

Abstract

Traffic measurement is the basis of analysis and prediction of network traffic. Its accuracy directly affects the reliability of upper applications. This paper proposes a rate-adaptive traffic measurement method called AdaRate. It adjusts the polling period according to fluctuations of flow rates which is calculated with a sliding window queue. We noticed that some fluctuations of flow rates may be dropped in AdaRate while the polling period is too long. Then we propose RAdaRate. It adjusts the polling period according to AdaRate or sets the polling period to minimum randomly. We compared the performance of AdaRate and RAdaRate with SWT in CeMon and Payless in Mininet when monitoring those VBR flows with frequent and large fluctuation. The results show that the accuracy of RAdaRate is the best while the overhead of AdaRate is the least. The overhead of AdaRate is 68.75% less than periodic polling.

Keywords: SDN; traffic measurement; adaptive

(Submitted on July 25, 2017; Revised on August 30, 2017; Accepted on September 15, 2017)

(This paper was presented at the Third International Symposium on System and Software Reliability.)

1. Introduction

Software-defined networks define a new generation of network paradigm and provide the possibility to solve the drawbacks of traditional network technology. In the research of software-defined networks, traffic measurement is the basis of traffic engineering, and the accuracy of traffic measurement has a direct impact on the performance of upper applications.

In the traditional network, traffic measurement technologies have been developing for many years. Cisco NetFlow [4,5] is a popular flow-based fine-grained traffic measurement technology. NetFlow uses a cache to sample or completely collect traffic statistics and then transmits the statistics to the central collector. NetFlow can provide rich and accurate traffic statistics. sFlow [9] collects traffic statistics by sampling. sFlow agents are inserted in the switches for collecting traffic statistics and then send the statistics to the sFlow collector. sFlow agents can be embedded into the ASIC chip. That reduces the overhead of the CPU on collecting traffic statistics and has no impact on the line rate of switches.

In software-defined networks, Jose et al. proposed a measurement framework in which packets are matched based on some rules, and only the highest-priority match can update the counter of flow entries. For tradeoffs of overhead and accuracy of identifying large traffic aggregates, they proposed a hierarchical heavy hitter algorithm [7]. Adrichem et al. proposed OpenNetMon [1], which can accurately monitor per-flow metrics. In OpenNetMon, controllers poll edge switches and adjust polling period according to the characteristics of flows for minimizing the number of queries. That makes controllers have tradeoffs of overhead and accuracy. Suh proposed OpenSample [13], which is a measurement platform based on sampling. Its target is to make the control loops of software-defined networks faster. In OpenSample, controllers use sFlow to measure

* Corresponding author.
E-mail address: yzhang@sei.ecnu.edu.cn .

network loads and individual flows. The measurement is close to real-time in OpenSample. Yu et al. proposed a traffic measurement method with OpenSketch [16]. OpenSketch process the packets with a three-stage pipeline in the data plane. The pipeline includes hashing, filtering, and counting. It reduces the overhead of memory and bandwidth. In the control plane of OpenSketch, the configurations of the pipeline and allocation of resources are automatic. The automation is implemented by a measurement library. Argyropoulos et al. proposed PaFloMon, a passive flow monitoring framework [2]. PaFloMon enables users to use tools, such as sFlow and NetFlow, and to deploy them widely in the legacy networking systems.

There are many types of research on traffic measurement in recent years. Much of this research focuses on the data plane or both of data plane and control plane, but we just focus on control plane for being compatible with more switches. In most research, controllers monitor traffics with periodic polling. Periodic polling is simple and accurate but also brings high overhead. For tradeoffs of overhead and accuracy, we decrease the overhead effectively by changing the polling period dynamically according to the behavior of flows. In this paper, our contributions are as follows:

- We propose AdaRate in which controllers adjust the polling period according to fluctuations of flow rates. The fluctuation of flow rates is calculated with a sliding window queue.
- We propose RAdaRate in which controllers adjust the polling period according to AdaRate or sets the polling period to minimum randomly.

The rest of this paper is as follows: Section II is the main introduction to related works. The design of traffic measurement methods is in section III. We evaluated the performance of adaptive measurement methods in section IV. The conclusions of the paper are in Section V.

2. Related Works

The traffic measurement methods discussed in this paper are mainly based on OpenFlow [10] protocol. OpenFlow is the most popular southbound interface in SDN currently [8]. In OpenFlow-enabled switches, the flow table can be programmed via OpenFlow. There are two ways for controllers to obtain statistics about flow's pass through switches: active polling and passive receiving.

In the existing passive measurement methods, such as Flowsense [15], controllers calculate flow rates with the statistics of *PacketIn* and *FlowRemoved* messages. These flow rates are the average rates within the alive duration of flows. We divide the measurement methods with active polling into two groups according to polling period: fixed polling period and dynamic polling period. The polling period of adaptive traffic measurement methods are usually dynamic. Payless [3] presents an adaptive measurement method in which controllers adjust polling period according to the size of flow rates. Su et al. proposed CeMon, a flow monitoring system in SDN. There three adaptive measurement methods presented: Proportional Tuning, EWMA tuning, and SWT [12]. The polling period is in proportion to the size of flow rate in Proportional Tuning. Proportional Tuning assumes that the future flow rates are based on current and previous data. In EWMA, controllers need to consider more historical data, and more emphasis is placed on recent data compared with Proportional Tuning. SWT is based on sliding window queue, in which controllers adjust the polling period according to the data in the sliding window. Tahaei et al. proposed an adaptive flow measurement method called Elastic [14] which is similar to SWT. Compared with SWT, controllers adjust polling period according to the ratio of the flow utilization on the link utilization in Elastic.

3. Design of Traffic Measurement Methods

In the software-defined networks, the southbound interface is located between control plane and data plane. In this paper, the method design and implementation are based on the OpenFlow protocol. Our traffic measurement method is mainly based on active polling. Controllers poll switches to obtain the statistics of flows with *FlowStatRequest* and *FlowStatReply* messages [6]. Controllers send the *FlowStatRequest* messages to switches, and then switches receive the messages, match flow table entries and encapsulate the statistics of the matched flow table entries to the *FlowStatReply* messages and send it to controllers.

3.1. Architecture

In OpenFlow-enabled networks, the basic mechanism of our traffic measurement is shown in Figure 1. Controllers collect the statistics of flows through the southbound interface, estimate the current utilization of flows and evaluate whether the current polling period applies to the current flow. If the current polling period is not applicable to the current flow, controllers will increase or decrease the polling period in our traffic measurement method.

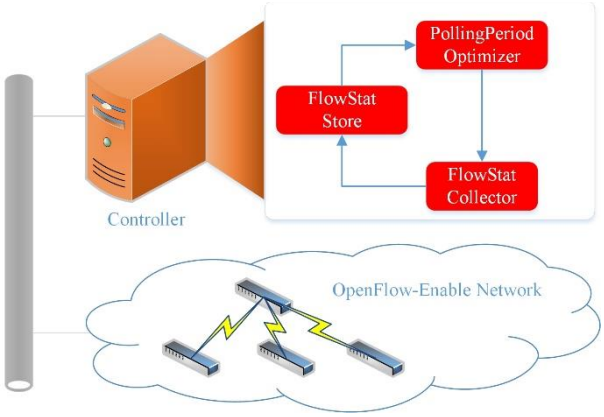


Figure 1. Adaptive Measurement Architecture

FlowStat Collector: It is an online statistical information collection module. The collector uses *FlowStatRequest* messages to poll flow entries on switches with a specified polling period. The switches send the statistics of flows to the collector with *FlowStatReply* messages. The collector analyzes the received messages and puts the statistics into FlowStat Store.

FlowStat Store: It calculates the current utilizations of flows with the statistics obtained from FlowStat Collector. The current utilizations of flows are stored in this module and are provided to Polling Period Optimizer and upper application.

PollingPeriod Optimizer: It determines how controllers adjust polling periods. The polling periods are adjusted according to the utilization information in FlowStat Store. PollingPeriod Optimizer provides the polling periods adjusted to FlowStat Collector.

3.2. Algorithm Design

3.2.1. Rate-Adaptive Measurement Algorithm

Rate-Adaptive Measurement Algorithm(AdaRate) is similar to SWT in CeMon. SWT is a traffic measurement method based on a sliding window queue. The queue stores recent data which are the differences between the values of two adjacent measurements. The value of two adjacent measurements represent the number of bytes transmitted within the interval of two measurements, as shown as equation (1). Controllers compare the value of current data with the mean and standard deviation of the values of the data in the queue in SWT. If the value of current data is larger than the sum of the mean and standard deviation, controllers decrease the queue size and polling periods. Otherwise, it increases the queue size and polling periods. This judgment is shown in detail in the equation (3):

$$var = bytecounter_n - bytecounter_{n-1} \tag{1}$$

$$\hat{w} = \frac{1}{N}\sum_{i=1}^{N} w_i \tag{2}$$

$$var > \hat{w} + 2\sqrt{\frac{\sum_{i=1}^{N}(\hat{w} - w_i)^2}{N-1}} \tag{3}$$

where w_i is the ith element in the sliding window queue, and $\hat{w}$ is the mean of all the element in the sliding window queue. $bytecounter_i$ is the value of byte counter obtained from nth measurement.

SWT has drawbacks. If the fluctuation of the values of the sliding window queue is frequent and large, the next value of var need to be enough large to trigger controllers for decreasing polling period. It means that the controllers are less sensitive to the fluctuation of flow rates for VBR flows with frequent and large fluctuation. We notice that it is necessary to consider

the fluctuation of flow rates, especially for VBR flows. As shown as Algorithm.1, we get the average rate within the interval between two adjacent measurements as ar :

$$var = \frac{bytecounter_n - bytecounter_{n-1}}{period_{lastpolling}} \tag{4}$$

where $bytecounter_i$ is the value of byte counter obtained from the current measurement. Controllers adjust polling periods according to the fluctuation of flow rates in AdaRate. If the standard deviation of the values of the data in the queue is enough large, controllers decrease the polling periods. Otherwise, controllers increase the polling periods. This judgment about fluctuation of flow rates is shown in equation (5):

$$\sqrt{\frac{\sum_{i=1}^{N}(\hat{w}-w_i)^2}{N-1}} > \frac{\hat{w}}{10} \tag{5}$$

3.2.2. AdaRate Based on Random Distribution

AdaRate adjusts polling periods according to the average rates within the interval between two adjacent measurements. There may be fluctuation of flow rates between two adjacent measurements, as show as Figure 2. We notice that the loss of fluctuation is proportional to the interval between two adjacent measurements, and could bring errors to the adjustment of the polling period. We propose RAdaRate in which controllers adjust the polling period according to AdaRate or set the polling rate to minimum randomly. The data obtained with minimum polling period usually provide controllers more fine-grained flow rates and are helpful to calibrate polling period, as shown as Algorithm.2. The random distribution is a uniform distribution:

$$\mathrm{P}\{X \geq x\} = \begin{cases} 1, x \leq 0 \\ 1-x, 0 < x \leq 1 \\ 0, x > 0 \end{cases} \tag{6}$$

Algorithm 1 AdaRate

Input: target flow f
Output: polling period τ

1. Initialization: $queue_{win} \leftarrow []$, $size_{win} \leftarrow 3$, $\tau_{min} \leftarrow 500ms$, $\tau_{max} \leftarrow 5000ms$;
2. Calculate var;
3. Push var into $queue_{win}$;
4. **if** $queue_{win}.stdev > 0.1 * queue_{win}.mean$
5. $\quad \tau_{now} \leftarrow max(\tau_{min}, \frac{\tau_{last}}{2})$;
6. $\quad size_{win} \leftarrow min(3, \frac{size_{win}}{2})$;
7. **else**
8. $\quad \tau_{now} \leftarrow min(\tau_{max}, \tau_{last} * 2)$;
9. $\quad size_{win} \leftarrow size_{win} + 1$;
10. **end if**
11. **if** $queue_{win}.lenth > size_{win}$
12. $\quad$ remove the front element from $\mathrm{queue_{win}}$
13. **end if**
14. **return** τ_{now};
15. **end**

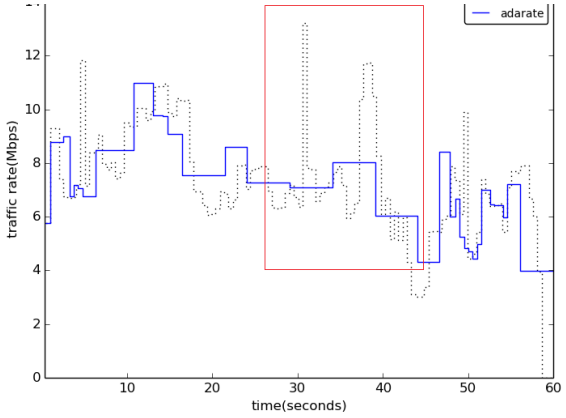


Figure 2. Periodic Polling and AdaRate For VBR Flow

Algorithm 2 RAdaRate

Input: target flow f
Output: polling period τ

1. Initialization: $queue_{win} \leftarrow []$, $size_{win} \leftarrow 3$, $\tau_{min} \leftarrow 500ms$, $\tau_{max} \leftarrow 5000ms$;
2. Calculate var;
3. Push var into $queue_{win}$;
4. **if** $random > 70\%$
5. $\tau_{now} \leftarrow \tau_{min}$;
6. **else**
7. $\tau_{now} \leftarrow AdaRate(queue_{min}, size_{win}, \tau_{min}, \tau_{max})$;
8. **end if**
9. **return** τ_{now};
10. **end**

4. Evaluation

In this section, we evaluate the performance of fine-grained adaptive traffic measurement methods using *Mininet*[11]. We implemented AdaRate, RAdaRate, Payless, SWT in CeMon and periodic polling on the Floodlight platform. We performed scenarios of VBR traffic to test traffic measurement methods using *vlc*. The details of the experimental scenarios will be presented in the SectionIV-A. The results will be presented in the SectionIV-B.

4.1. Experimental Setup

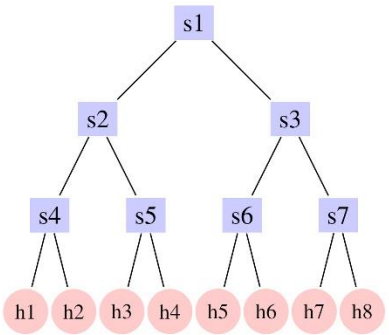


Figure 3. Topology

We emulate hosts and Open vSwitch using mininet on an Intel i5-3230M 2.60GHz and 16G RAM. The topology is shown in Figure 3, we use a 3-level tree topology for this evaluation. We use a video flows as VBR traffic. The video flow has a length of 60 seconds.

4.2. Results

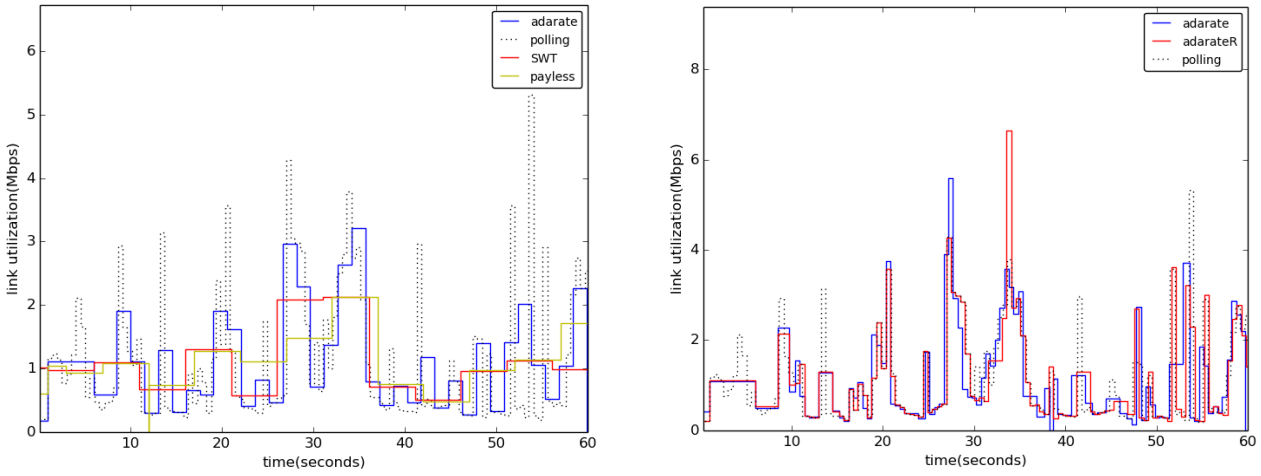
4.2.1. Accuracy

In this section, we evaluate AdaRate and RAdaRate in terms of accuracy of link utilization in 60 second times. All of SWT, AdaRate, and RAdaRate are fine-grained flow measurement method based on a sliding window. Their configuration parameters are functionally similar, so we evaluate their performance on the same parameter values($T_{min} = 500ms, T_{max} = 5000ms, winsize_{initial} = 3$). The polling period of periodic polling is the minimum polling period of controllers, and the link utilization measured with periodic polling is taken as the baseline scenarios[14]. We use two metrics for evaluating the accuracy of AdaRate: Fluctuation Loss Number which is the number of fluctuations loss of controllers in 60 second times; Error Rate represents the error between the measured data and baseline scenarios. Error Rate is defined as Equation (7). Where U_i is the link utilization measured at time i and B_i is from the baseline scenarios:

$$ErrorRate = \frac{1}{N}\sqrt{\sum_{i=1}^{N}\left|\frac{U_i - B_i}{B_i}\right|} \tag{7}$$

Figure 4(a) show the link utilization measured with SWT and AdaRate. Controllers adjusts polling period adaptively according to the changes of the pattern of flow rates in SWT. This measurement method is more suitable to burst traffic flows. It does not perform well for VBR flows with frequent and large fluctuation. Because if the fluctuation of the values of the sliding window queue is frequent and large, the next measured value need to be enough large to trigger controllers for decreasing polling period. It means that the controllers are less sensitive to the fluctuation of flow rates for VBR flows with frequent and large fluctuation. AdaRate adds the fluctuation of flow rates to the factors that affect the adjustment of polling period. As a result, AdaRate can capture the fluctuation of flow rates very well in VBR flows. As shown in Figure 4(a), Fluctuations Loss Number of AdaRate is obviously less than SWT, and the link utilization of AdaRate is more closely to the baseline scenarios than SWT.

Payless adjusts the polling period according to the size of flow rates. Therefore, when flow rates are less than the threshold set by Payless, if the flow is VBR, the Payless will lose a significant amount of fluctuations of flow rates, as shown as Figure 4(a).



(a) (b)

Figure 4. (a) The link utilization measured with AdaRate, Payless, SWT in CeMon and periodic polling and (b) The link utilization measured with AdaRate, RAdaRate and periodic polling

Table 1. The error rate of AdaRate, RAdaRate, Payless and SWT in CeMon

Traffic measurement method	Error Rate
Payless	4.6%
SWT	3.25%
AdaRate	3.1%
RAdaRate	2.45%

In RAdaRate, controllers adjust the polling period according to AdaRate or set the polling rate to minimum randomly. The values measured with minimum polling period will help the controllers to respond faster to the fluctuation of flow rates, or to reduce the loss of peak information at infrequent fluctuations. As shown as Figure 4(b), at 53 seconds, the controller in RAdaRate catches the fluctuation of flow rates lost by the controller in AdaRate. The controller in AdaRate drops this peak information because, at 49 to 51 seconds, a smoother flow rate causes the polling period to grow longer, causing the next sampling point to cross the fluctuation of flow rates of 53 seconds directly. The controller in RAdaRate captured the fluctuations of 53 seconds, so that the controller began to adjust the polling period in time, and improved the accuracy of the measurement method.

Table 1 shows the error rate of Payless, SWT, AdaRate, and RAdaRate.

4.2.2. Overhead

In this section, we evaluate AdaRate and RAdaRate in terms of overhead in 60s time. The overhead of the algorithm in the same scene increases with the improvement of its accuracy. The improvement in accuracy can be achieved by changing the configuration of the parameters of the algorithm. Therefore, we adjust the parameters of SWT, AdaRate, RAdaRate, and Payless to achieve the same level of accuracy, as shown as Figure 5.

We use polling times as metrics for evaluating the overhead of AdaRate and RAdaRate. Table 2 show the overhead of SWT, periodic polling, Payless, AdaRate, and RAdaRate. The overhead of periodic polling is highest. Compared with periodic polling, Payless reduces 50% of the overhead; SWT in CeMon reduces 64.1% of the overhead; RAdaRate reduces 66.4% of the overhead; AdaRate reduces 68.75% of the overhead.

Table 2. The overhead of AdaRate, RAdaRate, SWT in CeMon, Payless and periodic polling

Traffic measurement method	Polling times
Periodic polling	128
Payless	64
SWT in CeMon	46
AdaRate	40
RAdaRate	43

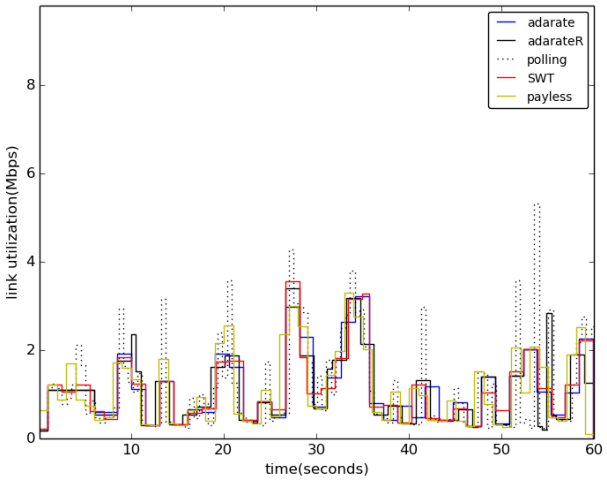


Figure 5. The link utilization measured by AdaRate, RAdaRate, SWT in CeMon, Payless and periodic polling

5. Conclusions

In this paper, we have proposed AdaRate and RAdaRate, which are rate-adaptive traffic measurement methods based on a sliding window. AdaRate adds the fluctuation of flow rate to the factors that affect the adjustment of the polling period. The experiment shows that AdaRate can capture more fluctuations of flow rates than SWT in VBR flows. RAdaRate is based on AdaRate and uses polling with minimum polling period to calibrate measurement algorithms randomly. The experiment shows that RAdaRate can catch the fluctuations of flow rates lost in AdaRate, but it brings a few additional overheads.

Acknowledgements

This paper is finished under the support of Key Program of the National Natural Science Foundation of China (61532019); ShanghaiCommittee of Science and Technology, China (No.15511104700,No.16DZ1100600); National Defense Basic Scientific Research program of China (JCKY2016212B004-2); open project fund of Shanghai Key Lab for Trustworthy Computing (No. 07dz22304201607). NSFC (Grant No.61472140).

References

1. N. L. M. V. Adrichem, C. Doerr, and F. A. Kuipers, "Opennetmon: Network Monitoring in OpenFlow Software-defined Networks," in *Network Operations and Management Symposium*, pp. 1–8, 2014.
2. C. Argyropoulos, D. Kalogeras, G. Androulidakis, and V. Maglaris, "Paflomona Slice Aware Passive Flow Monitoring Framework for OpenFlow Enabled Experimental Facilities," in *European Workshop on Software Defined NETWORKING*, pp. 97–102, 2012.
3. S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "Payless: A Low Cost Network Monitoring Framework for Software Defined Networks," in *Network Operations and Management Symposium*, pp. 1–9, 2014.
4. B. Claise, "Cisco Systems Netflow Services Export Version 9," *Rfc*, 2004.
5. C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a Better Netflow," in *ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 30 - September 3, 2004, Portland, Oregon, USA*, pp. 245–256, 2004.
6. B. Heller, "OpenFlow Switch Specification Version 1.3.5," 2009.
7. L. Jose, M. Yu, and J. Rexford, "Online Measurement of Large Traffic Aggregates on Commodity Switches," in *Usenix Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, pp. 13, 2011.
8. D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp.10–13, 2014.
9. G. Liu, W. Zuo, and Z. Ouyang, "Communication Network Traffic Anomaly Monitor Solution Based on sFlow," *Computer Engineering*, vol. 33, no. 6, pp. 245–247, 2007.
10. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM COMPUTER COMMUNICATION REVIEW*, vol. 38, no. 2, pp. 69–74, APR 2008.
11. R. L. S. D. Oliveira, A. A. Shinoda, C. M. Schweitzer, and L. R. Prete, "Using Mininet for Emulation and Prototyping Software-defined Networks," in *Communications and Computing*, pp. 1–6, 2014.
12. Z. Su, T. Wang, Y. Xia, and M. Hamdi, "CeMon: A Cost-effective Flow Monitoring System in Software Defined Networks," *COMPUTERNETWORKS*, vol. 92, no. 1, pp. 101–115, 2015.
13. J. Suh, T. T. Kwon, C. Dixon, W. Felter, and J. Carter, "Opensample: A Low-latency, Sampling-based Measurement Platform for Commodity SDN," in *IEEE International Conference on Distributed Computing Systems*, pp. 228–237, 2014.
14. H. Tahaei, R. Salleh, S. Khan, R. Izard, K.-K. R. Choo, and N. B. Anuar, "A Multi-objective Software Defined Network Traffic Measurement," *MEASUREMENT*, vol. 95, pp. 317–327, January 2017.
15. C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "Flowsense: Monitoring Network Utilization with Zero Measurement Cost," in *International Conference on Passive and Active Network Measurement*, pp. 31–41, 2013.
16. M. Yu, L. Jose, and R. Miao, "Software Defined Traffic Measurement with Opensketch," in *(to appear) NSDI*, 2013.