Adaptive Sampling for OpenFlow Network Measurement Methods

Guang Cheng
School of Computer Science and Engineering
Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education
Nanjing, P.R.China, 211198
gcheng@njnet.edu.cn

Jun Yu
School of Computer Science and Engineering
Key Laboratory of Computer Network and Information Integration, (Southeast University), Ministry of Education
Nanjing, P.R.China, 211198
jyu103.cn@gmail.com

ABSTRACT

OpenFlow provides a statistics collection scheme for per-flow and aggregate metrics at a user-specified frequency. However, periodic polling for flow statistics cannot well balance the tradeoff between measurement accuracy and limited control channel bandwidth. To further discuss the resource/accuracy tradeoff, this paper extends OpenFlow protocol to add sampling action for each monitoring flow entry, and systematically sample the matching packets to infer the flow-level statistics. Although traffic sampling can somehow be error-prone, it will provide near-real-time measurements of flow dynamics and determine its accurate polling frequency, which the polling-based approach cannot achieve. In this paper, we propose a per-flow sampling solution to instruct the controller to adaptively adjust polling frequency, then evaluate it in the context of link utilization monitoring.

KEYWORDS

Sampling, Software Defined Networking, OpenFlow Protocol, Adaptive Measurement

ACM Reference format:
Guang Cheng and Jun Yu. 2017. Adaptive Sampling for OpenFlow Network Measurement Methods. In *Proceedings of CFI'17, Fukuoka, Japan, June 14-16, 2017,* 7 pages.
https://doi.org/10.1145/3095786.3095790

1 INTRODUCTION

The emergence of Software-Defined Networking, Network Virtualization and Network Function Virtualization has led to the development of varieties of novel network applications. The unique attributes of these applications, such as dynamic optimization of network resources and construction of virtual network, require accurate and real-time statistics of massive network traffic at different time granularities. Network measurement needs to satisfy diverse monitoring requirements, provide more accurate and real-time measurement results, but also supports the optimal scheduling and dynamic loading of monitoring tasks. By separating the control plane and the data plane, SDN [5, 16] makes it possible for the measurement tasks to have a global view of the network to coordinate and control dynamic traffic monitoring in multi-switch. With software-defined measurement, the controller can schedule flow statistics collection tasks at different temporal and spatial scale, and optimize the deployment of network-wide measurement tasks. As the network traffic continues to grow, the key challenge for the network measurement is how to obtain accurate flow statistics in the high-speed network with very limited computational and storage resources [15]. For instance, each slot of new generation of data center architecture can support one or more ports of 40Gbit/s or 100Gbit/s, and the exchange capacity is at 1 10Tbit/s level. Therefore, the key challenge for modern network management is how to measure the high-speed network traffic more accurately and efficiently [4].

Flow programmable based network provides performance and reliability requirements for a wide range of network applications, such as rapid failure recovery, fast and reliable data transmission. In order to adapt to the changes of network traffic and architecture according to the pre-defined measurement goals, we need to continuously monitor the utilization of each link. SDN has emerged as the main technology for the next generation of network architecture, and supports scheduling for a variety of concurrent and dynamic instantiation measurement tasks. There are three ways [1] for the SDN controller to fetch flow entries statistics from OpenFlow switches:1) PacketIn Messages upon the arrival of the first packet of a new flow; 2) FlowRemoved Messages upon the expiry of flow entries; 3) Polling flow entries counters. Given limited resources and the need to support global optimization and dynamic scheduling of monitoring strategies, it is important to efficiently allocate flow entries for flow measurement tasks and coordinate the polling frequency for flow statistics. In this paper, we study the tradeoff between the limited control channel bandwidth and statistics querying accuracy to select the appropriate polling frequency.

The controller can fetch the flow statistics in the OpenFlow switches through the control channel at a pre-defined polling interval, but periodic polling cannot handle the flow dynamics and balance the monitoring accuracy and control channel bandwidth usage, making current polling scheme sub-optimal. Network applications require accurate and real time statistics on network resources at different aggregation levels, while the monitoring overhead for statistics collection should be minimal. Many network applications demand accurate and timely statistics, such as traffic engineering, accounting and load balancing. To strike a better balance between

CFI'17, June 14-16, 2017, Fukuoka, Japan
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5332-8/17/06…$15.00
https://doi.org/10.1145/3095786.3095790

statistics collection accuracy and monitoring overhead, we use sampling technique to improve the utilization efficiency of resources while satisfying measurement accuracy. Currently, adaptive sampling technique has been widely used in traditional IP networks, but not been studied in SDN environment.

In order to study the tradeoff between monitoring resources and accuracy, this paper implements a flexible sampling extension for OpenFlow, systematically sampling the matching packets to infer original flow length. Despite the fact that traffic sampling can be error-prone, it can perceive flow dynamics to instruct the adjustment of polling frequency. Thus, we propose per-flow passive sampling solution to instruct the controller to adaptively tuning the polling frequency of each flow.

The rest of the paper is organized as follows. In Section 2, we overview related work regarding variable rate adaptive data collection methods used in SDN network measurement, and point out some advantages and disadvantages of current polling approaches. Next, we propose an adaptive sampling measurement method based on SDN framework in Section 3. Section 4 presents an adaptive polling scheme based on adaptive sampling. In Section 5, we elaborates on the evaluation results of our approach by monitoring link utilization. Finally, Section 6 concludes this paper and point out some future directions of our work.

2 RELATED WORK

Flowsense [13] passively captures and analyses the control messages between switches and the controller to efficiently infer link utilization. It mainly utilizes the PacketIn and FlowRemoved messages to indicate the start and end time of each flow. The PacketIn message is sent by the switch when there is no matching entry for a new flow, containing the headers of the packet. The FlowRemoved message is triggered by the expiry of the flow entry, which is associated with two timeout values. The hard timeout is counted at the time of the flow entry installation, while the soft timeout is counted at the time of last matching packet. Since Flowsense must wait for all the active flows at the checkpoint to expire, it would have a large delay of up to 10s measuring the link utilization.

OpenTM [11] actively queries the number of packets and bytes for the active flows from an optimal set of switches, and the focus of this paper is about to solve the problem of traffic matrix estimation accuracy and switches querying cost. The logic of OpenTM is to get routing information from the controller, find the routing path of each flow and periodically queries for statistics from switches on the flow path. Due to the packet loss, the statistics for a given flow may be different from different switches on the path. OpenTM considers the last switch closer to the destination as the polling switch, which has most accurate TM compared to any other switches on the same path. However, it will incur significant overhead on the first/last switches, and the polling frequency is an important factor that affects the accuracy and overhead of TM estimation. OpenTM only considers the fixed querying frequency for all flow statistics on different switches, and the querying frequency can adjust itself for each source-destination IP pair based on the network performance (RTT, available bandwidth). However, OpenTM doesnâĂŹt consider the factor that the polling frequency should quickly adapt to the flow dynamics and traffic fluctuation.

PayLess [3] is a measurement framework built on top of an OpenFlow controller and provides a high-level RESTful API. This framework translates the high level primitives from the monitoring applications and conceals the details of statistics collection and storage management. In PayLess, the controller sends the FlowStatisticsRequest messages to switches to query specific flow information, and the switch send FlowStatistics-Reply messages containing the duration and byte count of that flow to the controller. This paper proposes a variable rate statistics collection method. It utilizes the collected data in the FlowStatisticsReply messages to determine whether the flow rate exceeds a pre-defined threshold. If the threshold is exceeded, the polling frequency for that flow will be increased. Otherwise, the polling frequency will be decreased accordingly. Meanwhile, the algorithm will batch the FlowStatisticsRequest messages together for flows with the same timeout. The experimental results show that PayLess is more accurate than FlowSense in detecting traffic spikes, and sends out 50% less control messages than that of periodic polling method.

CeMon [10] proposes an adaptive fine-grained polling scheme to optimize the polling scheme for flow entries statistics. OpenNetMon [12] adaptively adjusts the polling interval in terms of the flow behavior characteristics. Sahri N M [8] employs a higher polling frequency for abnormal flows. The above solutions dynamically adjust polling frequency based on the historical polling records. Since these approaches cannot get timely feedback through the traffic status on the switches, it is difficult to accurately fetch measurement results for each flow.

Exploring the tradeoff between resource and accuracy is a big challenge in SDN network measurement research. High-frequency statistics collection will relatively get more accurate data, which is particularly effective in high-speed network. However, this will induce significant monitoring overhead on the network when the traffic is relatively stable. For this reason, it is necessary to adjust the polling frequency in real time to strike a better balance between measurement accuracy and overhead. Although the controller can fetch the flow statistics through OpenFlow control messages, this method cannot timely perceive and adapt to the traffic dynamics. Adaptive sampling is used in this paper to estimate the flow statistics and better capture the network dynamics. If the sampled traffic is stable, then we can get per-flow statistics through sampling and increase the corresponding polling interval. If the sampled traffic is busy, the flow statistics estimated by sampling might be error-prone, thus we need to decrease polling interval to improve measurement accuracy.

3 SYSTEM DESCRIPTION

In this section, we presents an overview of the proposed method and how it uses the sampling technique to accurately measure the network without incurring too much overhead.

3.1 System Architecture

SDN controllers, like RYU, Floodlight, and NOX, provide a platform to develop customized network measurement applications that do not need to be aware of the implementation details and complexity of the underlying network. The SDN control plane performs direct control over the data-plane elements through a well-defined interface usually referred to as southbound API. OpenFlow is a dominant protocol of such example. The SDN control plane also takes care of collecting real-time flow statistics at different aggregation levels (e.g., packet, byte, port, etc.), integrates the raw data and sends the results to the measurement tasks. However, continuously polling for flow statistics will induce significant overhead on both the control plane and switches. Moreover, it will waste resources when the traffic is slow or stable, and cannot timely provides precise flow information as the traffic fluctuates. Hence, an adaptive OpenFlow monitoring system is required to provide timely and accurate per-flow or aggregate statistics to the network measurement application.

To this end, we propose a low cost accurate OpenFlow measurement method based on adaptive sampling. It utilizes the OpenFlow control messages and adaptive sampling to exercise direct control over the interval of sending FlowStatisticsRequest. Since the OpenFlow switches can enable flow-based management by installing the low-level rules in switches, the adaptive sampling is also flow-based. It can sample the specific flow or aggregate flow according to the requirements of measurement applications. By estimating the original flow size during each measurement period via sampling, we can infer the flow dynamics and adjust the polling accordingly. The measurement applications don't have to understand the implementation details of traffic sampling and statistics collections.

We describe the architecture of adaptive sampling network monitoring method in Figure 1. In general, it is mainly developed on the control plane and application plane. The control plane is the heart of the whole monitoring framework, and is responsible for making routing decisions, installing forwarding and sampling rules, and collecting flow statistics. Therefore, the main components of the control plane include Flow Status Tracker and Polling Manger. While in the application plane, we implement the Traffic Sampling to analyze the sampled traffic, and Polling Scheme Optimizer to generate the new polling interval for each active flow according to the sampling results. The measurement tasks can be link utilization, packet loss and anomaly detection monitoring applications.

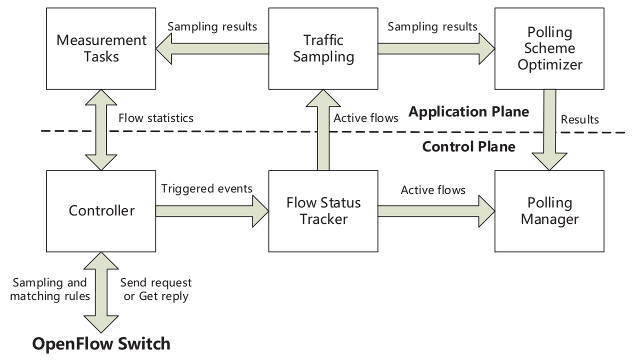

Figure 1: Adaptive Sampling Network Monitoring Architecture

Flow Status Tracker. This component is responsible for intercepting PacketIn and FlowRemoved messages to maintain an active flows list for each switch. When a switch has no matching entry for a new flow, it sends a PacketIn messages containing the flow identifier and datapath id of the switch. The system starts to assign an initial polling interval and a matching rule to that flow by sending a FlowMod message. Similarly, the switch will send a FlowRemoved message that notify the controller of flow entry expiry to terminate the polling action.

Traffic Sampling. The Traffic Sampling component installs and updates sampling rules in the switches to perform the sampling action. Since the OpenFlow enabled switches can execute per flow or per port action for these matching packets, it can also perform sampling action for these matching packets and decide which packet should be sampled. To minimize the network overhead, this component can tune the sampling ratio based on the sampled traffic volume of each flow.

Polling Scheme Optimizer. We have to adaptively adjust the polling interval to balance the tradeoff between measurement accuracy and overhead on the basis of sampled traffic. This component determines the polling frequency for each active flows in the network. Per flow sampling can timely capture its flow spikes and trend. Therefore, this component can increase the polling interval as the flow traffic is stable, and decrease the polling interval as the traffic is busy.

Polling Manager. This module is responsible for exercising statistics collection on the active flows. By receiving the results from Flow Status Tracker and Polling Scheme Optimizer, this module could update its polling timeout value for the active flow and remove flow records for expired flows to avoid wasting resources. When the timeout value of each flow is reached, the polling action is performed to collect the flow statistics.

3.2 Adaptive Sampling

In this paper, we propose an adaptive sampling approach to accurately and efficiently measure the OpenFlow network. In order to perform sampling action on the packets matching on the sampling flow entry, we extend the OpenFlow protocol to add two kinds of sampling: (i) Stochastic sampling: selecting each packet with probability p, $0 \leq p \leq 1$; (ii) Systematic sampling: select m packets from n consecutive packets. This sampling extension is very flexible, and has already been discussed in the literature of [9]. To customize the sampling action for each IP flow, a separate sampling table is additionally used for the switch to count and sample these matching packets while prevent monitoring from interfering in normal traffic forwarding. Meanwhile, we set higher priority to those sampling rules with longer IP prefixes to get fine-grained flow statistics.

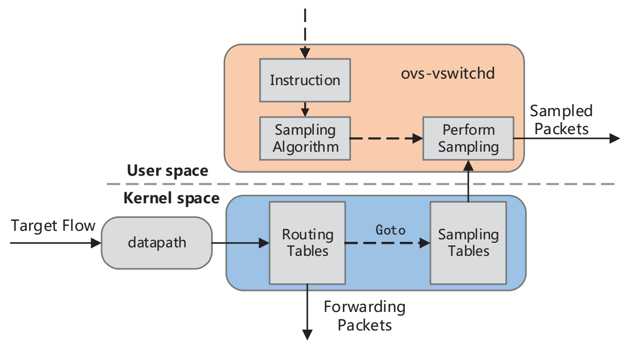

Figure 2: The workflow of traffic sampling at the data plane

In addition to extend the OpenFlow protocol, the sampling algorithm needs to be implemented at the data plane. Figure 2 shows the workflow of traffic sampling at the data plane. The corresponding sampling algorithms include stochastic sampling and systematic sampling, and the systematic sampling is used in this paper to infer the original flow length. To reduce memory usage, the systematic sampling will make use of the *rx_packets* field in each flow entry. For instance, to select the first m packets from each n packets, the sampling algorithm will decide whether or not to sample the matching packet based on the equation of $rx\_packets \bmod n < m$ [9]. If so, then the packet will be sampled and forwarded to a reserved port. Simultaneously, we use a single, centralized collector to gather samples from all the switches. From these sampled packets, the collector can infer a variety of information about the network including the heavy hitter which accounts for a majority of packets or bytes.

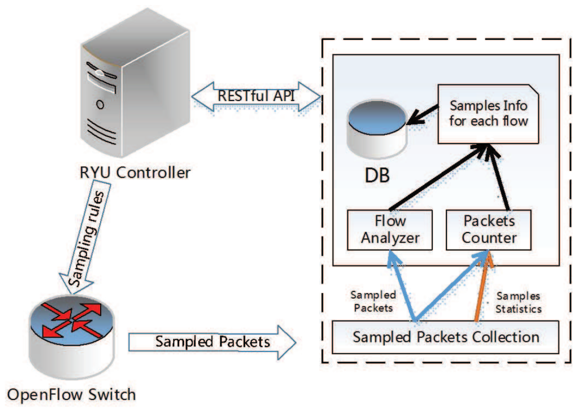

Figure 3: Collector Architecture

The architecture of the collector is illustrated in Figure 3. By gathering these sampled packets from switches, the collector can roughly estimate the number of per-flow or aggregate statistics. For example, it can infer the original number of packets and bytes in each flow by simply multiplying the number of sampled bytes and packets by sampling ratio, N [7]. Although this method might have some statistical errors in estimating the original flow length, it can timely capture the flow dynamics and be used to adaptively tune the polling interval.

To mitigate the overhead of sampled traffic on the collector, an adaptive sampling is imperative. We can exercise direct control over the sampling ratio of each flow by sending FlowMod messages to maintain the sampled traffic as stable as possible. The rationale behind this sampling ratio adjustment is that we sample less packets from flows with more traffic and more packets from flows with low flow rate. If the flow rate increases, the sampled traffic for that flow will also increase, thus we need to decrease the corresponding sampling ratio to further reduce the sampling overhead on the collector. To avoid excessively sending the FlowMod messages, the adjustment of sampling ratio should consider more historical data into account.

Suppose p_n is the sampling ratio of flow f_i at the n_{th} sampling interval, then we can derive it from p_{n-1} with the following equation:

$$p_n = p_{n-1} v \frac{t_n - t_{n-1}}{s(t_n) - s(t_{n-1})} \tag{1}$$

Where $s(t_n)$ and $s(t_{n-1})$ are the total number of bytes in the measured flow f_i at time t_{n-1} and t_n, respectively, v is a coefficient for the equation, also known as the expected sampling traffic volume for each flow. The sampling ratio will tune proportionally according to the traffic change rate. However, when the traffic changes severely, this scheme may frequently adjust the sampling ratio and result in continually sending FlowMod messages to the switches.

To this end, we adopt an exponential moving average, also known as an exponentially weighted moving average (EWMA) [6], to smooth the value of sampling ratio. EWMA can weight historical data or recent observations by choosing an appropriate smoothing coefficient. The following equation shows the improved version of sampling ratio adjustment:

$$p_n^{ewma} = \delta p_n + (1 - \delta) p_{n-1}^{ewma} \tag{2}$$

Where the observation p_n is derived from the equation (1), $\delta (0 < \delta < 1)$ is the smoothing coefficient. To avoid frequently modifying the sampling ratio, we should consider more historical data. Thus, we set the value of δ to 0.5 in this paper to avoid the fluctuations of sampling ratio adjustment. Existing sampling methods are prone to loss information about small traffic flows when massive traffic flows pass the switch, this adaptive sampling scheme enables adjusting sampling ratio for each flow independently, which can be used to detect the traffic anomalies like super-spreader which connects a large number of distinct destinations during a measurement interval [2].

4 ADAPTIVE POLING SCHEME

In this section, we present a low cost and fine-grained flow level polling scheme based on adaptive sampling to measure the network resources. The goal is to fetch timely and accurate statistics, while bounding the flow table and monitoring bandwidth usage. We utilize the passive sampled traffic to instruct the adjustment of active polling frequency. Given the limited CPU and control channel bandwidth resources, it is necessary to actively adjust the polling interval for different flows. There are two constraints: (i) As the number of flows becomes large, the number of polling from controller to switches also increases, which result in excessive usage of control channel bandwidth; (ii) Given the overall polling accuracy bound, it is necessary to tune the polling frequency of different flows, which means we should maintain a higher polling interval for stable flows and a lower polling interval for busy flows.

4.1 Polling Frequency Tuning

To explore the feasibility of our adaptive sampling approach, we design a polling frequency tuning scheme to adjust the interval of sending FlowStatisticsRequest Messages for each flow. Relying solely on the control messages cannot timely capture traffic dynamics of each flow, instead we get the flow statistics by simply multiplying the number of bytes by the sampling ratio of each flow, p. A pseudo-code of this sampling based polling scheme is depicted in Algorithm 1. Flow Status Tracker is used to intercept the PacketIn messages so that the adaptive polling scheme can assign a flow entry to the new active flow alone with an initial sampling ratio and statistics collection interval, t seconds. For the expired flow entries, it can get the statistics and duration of flows matched against it from FlowRemoved messages. To respond to the timeout event, we first calculate the difference between the previous and current estimated byte count against flows with the same timeout value. If the difference is not above a threshold, say $mean + 2 * std$, where $mean$ and std are the average and standard deviation of historical records respectively. The timeout value for satisfied flows will be proportionally multiplied by a small constant, say α. Otherwise, the timeout value for not satisfied flows will be divided by another constant β to quickly adapt the traffic changes. Additionally, the polling interval needs to be controlled within the range (t_{min}, t_{max}). The rationale behind this tuning algorithm is that we allocate more control channel bandwidth for flows that change significantly and reclaim bandwidth resources for flows that are stable in order to balance the tradeoff between resource and accuracy.

Algorithm 1 Adaptive Polling Scheme

```
Require: Event $e$, Historical records $records$; currently active
    flows $active\_flows$ ; polling interval associated with flows
    $polling\_table$;
Ensure: next polling scheme;
 1: if $e$ is a $packet\_in$ event then
 2:     $flow\_bytes \leftarrow 0$
 3:     $f \leftarrow < flow\_bytes, t, p >$
 4:     $active\_flows \leftarrow active\_flows \cup f$ //register a new flow
 5:     $polling\_table[t] \leftarrow polling\_table[t] \cup f$
 6: else if $e$ is timeout event for $t$ in $polling\_table$ then
 7:     for flow $f$ in $polling\_table[t]$ do
 8:         $cur\_bytes \leftarrow getSamples(f)/f.p$ //get sampled statistics
 9:         $last\_bytes \leftarrow f.flow_bytes$
10:         $diff\_rate \leftarrow |cur\_bytes - last\_bytes|$
11:         if $diff\_rate < records.mean + 2 \times records.std$ then
12:             $f.t \leftarrow min(t_{max}, f.t \times \alpha)$ // the traffic is stable
13:         else
14:             $f.t \leftarrow max(t_{min}, f.t/\beta)$// the traffic is busy
15:         end if
16:         $f.flow\_bytes \leftarrow cur\_bytes$
17:         move $f$ to $polling\_table[f.t]$
18:         move $cur\_bytes$ to $records$
19:         send a $flow\_stats\_request$ to $f.dpid$
20:     end for
21: end if
```

4.2 Dynamic Resource Allocation

Due to these facts that a large number of flows present in the network and flow-based counters are maintained in a power-hungry TCAM, the number of flow entries for monitoring is very limited. If we provide an exact IP flow with a counter, the TCAM resources in switches will be quickly exhausted. When the flow table is nearly full, the switch needs to age-out the non-active flows in flow table to reclaim space for new flows. Frequently changeover may lead to a large delay in processing the packets and influence the normal traffic routing. In order to configure TCAM counters more reasonably, we propose a flow table resources allocation method based on the adaptive sampling in this section. It will ultimately find the aggregate flows that account for large traffic volume in the network.

In this paper, we monitor traffic aggregates to tradeoff some measurement accuracy. Every measurement task starts by measuring an initial set of prefixes and continues to divide that prefix or merge sibling prefixes on the basis of historical data. We employ the linear prediction approach to anticipate next monitoring result. The future values of traffic aggregates are estimated as a linear function of historical records. If the actual measurement value falls into the range of the estimation, it suggests the requirement of merging. Otherwise, it requires a division for that aggregate flow entry to exercise finer-grained control over the flows [14].

To be specific, the traffic sampling component maintains a list of n records for each aggregate flow. Let v_i and p_i represent the sampled bytes and its corresponding sampling ratio at time t_i respectively. The predicted value of an aggregate flow can be derived with the Equation(3).

$$v_p = v_n \frac{1}{p_n} + \frac{t_n - t_{n-1}}{n-1} \sum_{i=1}^{n-1} \left(\frac{v_{i+1} p_i - v_i p_{i+1}}{(t_{i+1} - t_i) p_{i+1} p_i} \right) \tag{3}$$

Then the difference between the predicted value and actual value is calculated using the following equation:

$$Error_p(n) = \| \frac{v_{n+1}}{p_{n+1}} - v_p \| \tag{4}$$

If the difference is above a predefined threshold, it indicates a change in traffic volume and needs to divide that prefix to get more fine-grained flow statistics. Conversely, if two sibling nodes in the prefix trie are both below that threshold, it suggests that the traffic for these two neighboring nodes is stable and require merging them to free up space for other flows. Similarly, we set the threshold value to $mean + 3 * std$, where $mean$ and std are the average and standard deviation of historical records respectively.

5 EXPERIMENTS

In this section, we evaluate the performance of the adaptive sampling based OpenFlow network measurement method from different aspects such as measurement accuracy and monitoring overhead. We have also implemented a link utilization monitoring application using both periodic polling scheme and the proposed polling algorithm to demonstrate the effectiveness of our algorithm. The RYU controller is used as our controller platform for its well defined API and rich set of protocols. To simulate an OpenFlow network, we use the Mininet to set up a small testbed comprising seven OpenFlow switches and eight hosts shown in Figure 4. For the experiments of the adaptive polling algorithm, we have set the minimum and maximum polling interval to 0.5s and 5s, respectively. And the constant polling interval for periodic polling is set to 1s. We use the real packet traces of 60s to perform the simulation. For the newly added flow entries, we have set the soft timeout to 10s to evict the long idle flow entries and initial sampling ratio to the maximum value to avoid losing information about small traffic flows.

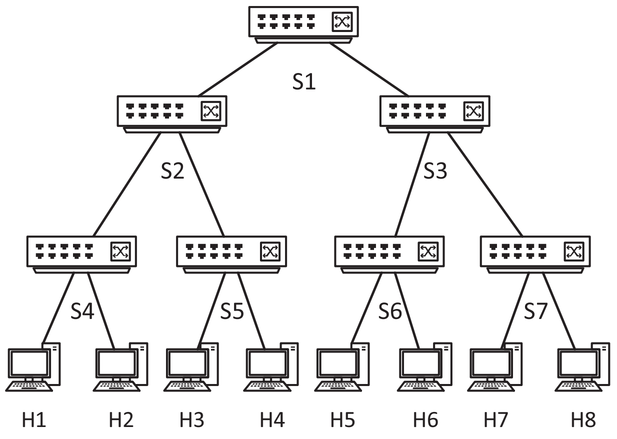

Figure 4: Network Topology for Experiment

5.1 Link Utilization

The link utilization can be obtained by summing up the latest polling results of each flow in the monitoring link. Therefore, the utilization of a particular link can be derived with Equation(5).

$$Link\_Utilization = \sum_{i=1}^{n} \left(\frac{bytes_{t_n}^{f_i} - bytes_{t_{n-1}}^{f_i}}{t_n - t_{n-1}} \right) \tag{5}$$

Where the $bytes_{t_n}^{f_i}$ represents the polling results of f_i at time t_n. As shown in Figure 4, we treat the link between the $S1$ and $S2$ as the monitoring link, and use the tcpreplay tool to replay the real packet traces to perform the simulation. We compare the utilization obtained by adaptive polling algorithm with that gathered from periodic polling scheme to evaluate the effectiveness of our proposed method. Figure 5 shows the utilization of monitoring link measured with two different approaches.

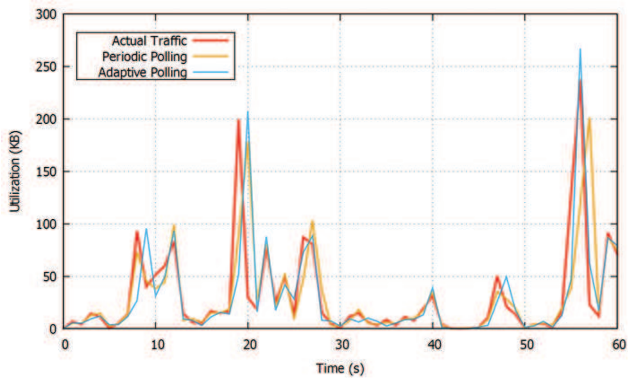

Figure 5: Link Utilization

Figure 5 shows the link utilization of two polling scheme. The link utilization measured by our approach follow closely to actual results. Although the adaptive polling approach cannot timely capture the traffic spikes, it quickly decrease the polling interval of the responsible flows to adapt to the traffic fluctuation. Figure 6 shows the sampled traffic records of a large flow. When the flow traffic changes significantly, the sampling ratio gradually adjusts itself until the minimum or maximum sampling ratio is reached to stabilize the sampled traffic. However, the sampling ratio adjustment method cannot respond to the sudden dynamics and reduce the number of sampled traffic spikes.

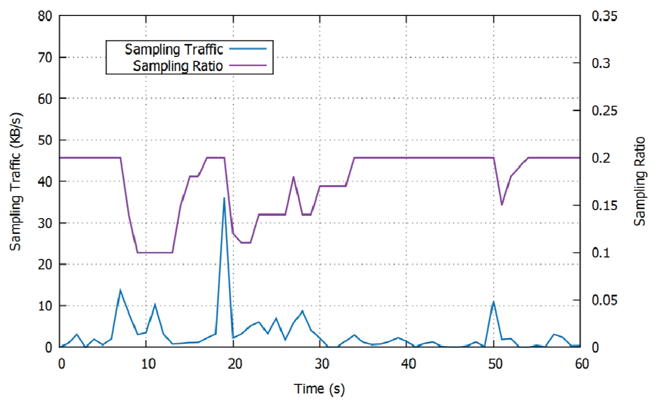

Figure 6: Sampling Traffic vs. Sampling Ratio

5.2 Monitoring Overhead

In this paper, we consider the number of FlowStatisticsRequest messages sent from the controller to be the network overhead metrics, and the overhead is recorded at the timeout expiration events. And we count only once for flows at the same timeout since the controller can batch the FlowStatisticsRequest messages together to mitigate the monitoring overhead.

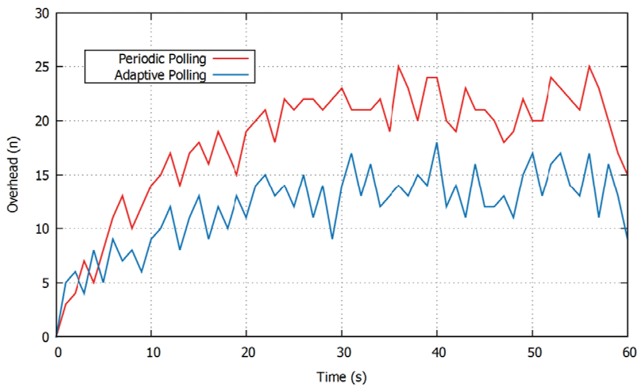

Figure 7: Sampling Traffic vs. Sampling Ratio

Figure 7 shows the monitoring overhead of the periodic polling scheme and our proposed algorithm. The periodic polling scheme polls all the active flows at fixed interval resulting in a large number of messages sent to the switches. While our proposed algorithm reduces the number of messages by increasing the polling interval for a majority of flows, thereby reducing the network overhead. However, the adaptive polling scheme may induce larger overhead on the network than periodic polling scheme. For instance, from 0 to 5s, the number of flows is small and the traffic changes dramatically. To adapt to changes, the adaptive polling scheme gradually decreases the polling interval to fetch accurate statistics. When the number of flows becomes large, only a few of large flow changes significantly, so the overall overhead of our proposed algorithm is much less than periodic polling scheme.

6 CONCLUSIONS

In this paper, an adaptive sampling approach for OpenFlow network measurement is proposed to dynamically adjust the polling interval for statistics collection. We study the tradeoff between the monitoring accuracy and the network overhead, and adjust the polling interval of sending FlowStatisticsRequest messages through the per-flow sampled traffic. We then compare our adaptive polling scheme with the periodic polling scheme, and the experimental results show that the proposed method could tremendously reduce the monitoring overhead with negligible loss of measurement accuracy. To further reduce the sampling traffic, an adaptive sampling method is proposed to tune the sampling ratio for each flow independently. We set a higher sampling ratio for large flows to minimize the sampling overhead and a lower sampling ratio for small flows to avoid losing flow information. In our ongoing work, we are going to leverage this adaptive sampling approach to detect traffic anomalies like super-spreaders and port scanning activities.

ACKNOWLEDGMENTS

This work was supported by the National High Technology Research and Development Program (863 Program) of China (2015AA 010201); six talent peaks of the high level Talents Project of Jiangsu province (2011-DZ024); and the CRENET IPv6 Innovation Project (Project No. NGII20150108).

REFERENCES

[1] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. 2014. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 87–95.
[2] Guang Cheng and Yongning Tang. 2013. Line speed accurate superspreader identification using dynamic error compensation. *Computer Communications* 36, 13 (2013), 1460–1470.
[3] Shihabur Rahman Chowdhury, Md Faizul Bari, Reaz Ahmed, and Raouf Boutaba. 2014. Payless: A low cost network monitoring framework for software defined networks. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 1–9.
[4] Nick Duffield, Carsten Lund, and Mikkel Thorup. 2005. Estimating flow distributions from sampled flow statistics. *IEEE/ACM Transactions on Networking (TON)* 13, 5 (2005), 933–946.
[5] Open Networking Fundation. 2012. Software-defined networking: The new norm for networks. *ONF White Paper* 2 (2012), 2–6.
[6] Cynthia A Lowry, William H Woodall, Charles W Champ, and Steven E Rigdon. 1992. A multivariate exponentially weighted moving average control chart. *Technometrics* 34, 1 (1992), 46–53.
[7] Peter Phaal. 2002. Packet sampling basics. *http://www. sflow. org/* (2002).
[8] Nor Masri Sahri and Koji Okamura. 2015. Adaptive Anomaly Detection for SDN. *Proceedings of the Asia-Pacific Advanced Network* 40 (2015), 57–63.
[9] Sajad Shirali-Shahreza and Yashar Ganjali. 2013. FleXam: flexible sampling extension for monitoring and security applications in openflow. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 167–168.
[10] Zhiyang Su, Ting Wang, Yu Xia, and Mounir Hamdi. 2015. CeMon: A cost-effective flow monitoring system in software defined networks. *Computer Networks* 92 (2015), 101–115.
[11] Amin Tootoonchian, Monia Ghobadi, and Yashar Ganjali. 2010. OpenTM: traffic matrix estimator for OpenFlow networks. In *International Conference on Passive and Active Network Measurement*. Springer, 201–210.
[12] Niels LM Van Adrichem, Christian Doerr, and Fernando A Kuipers. 2014. Opennetmon: Network monitoring in openflow software-defined networks. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 1–8.
[13] Curtis Yu, Cristian Lumezanu, Yueping Zhang, Vishal Singh, Guofei Jiang, and Harsha V Madhyastha. 2013. Flowsense: Monitoring network utilization with zero measurement cost. In *International Conference on Passive and Active Network Measurement*. Springer, 31–41.
[14] Ying Zhang. 2013. An adaptive flow counting method for anomaly detection in SDN. In *ACM Conference on Emerging NETWORKING Experiments and Technologies*. 25–30.
[15] AP Zhou, G Cheng, and XJ Guo. 2014. High-Speed network traffic measurement method. *Journal of Software* 25, 1 (2014), 135–153.
[16] Qing-Yun ZUO, Ming Chen, Guang-Song ZHAO, Chang-You XING, Guo-Min ZHANG, and PeiCheng JIANG. 2013. Research on OpenFlow-Based SDN Technologies [J]. *Journal of Software* 5 (2013), 015.