Scientific Research Publishing

Int. J. Communications, Network and System Sciences, 2017, 10, 54-66
http://www.scirp.org/journal/ijcns
ISSN Online: 1913-3723
ISSN Print: 1913-3715

A Low-Cost Measurement Framework in Software Defined Networks

Qiang He, Shengbao Wang

School of Computer Science and Engineering, Northeastern University, Shenyang, China
Email: heqiangcai@gmail.com, cfwsbtx@gmail.com

How to cite this paper: He, Q. and Wang, S.H. (2017) A Low-Cost Measurement Framework in Software Defined Networks. *Int. J. Communications, Network and System Sciences*, **10**, 54-66.
https://doi.org/10.4236/ijcns.2017.105B006

Received: March 6, 2017
Accepted: May 23, 2017
Published: May 26, 2017

Abstract

Software Defined Network (SDN) makes network management more flexible by separating control plane and data plane, centralized control and being programmable. Although, network measurement still remains in primary stage in SDN, it has become an essential research field in SDN management. In this context, this paper presents a low-cost high-accuracy measurement framework to support various network measurement tasks, such as throughput, delay and packet loss rate. In this framework, we only measure per-flow edge switches (the first and the last switches). In addition, a new adaptive sampling algorithm is proposed to significantly improve measurement accuracy and decrease network overhead. Meanwhile, we consider a low-cost topology discovery approach into our framework instead of topology discovery currently implemented by SDN controller frameworks. In order to improve the accuracy of delay, we also join a time threshold value to adjust the time delay. Furthermore, we consider and analyze the balance between measurement overhead and accuracy in several aspects. Last, we utilize POX controller to implement the proposed measurement framework. The effectiveness of our solution is demonstrated through simulations in Mininet and Matlab.

Keywords

Software Defined Network, Adaptive Sampling, POX Controller, OpenLL

1. Introduction

Software Defined Network, as the mainstream technique of future network architectures has attracted more and more attention from academia and industry [1] [2]. Control and measurement are two important parts of SDN management, so far, most research has been focused on SDN control and little work has gone into SDN measurement. Besides, centralized control of the controller makes control process more complicated, which consumes too much network re-

sources. Therefore, it's essential to propose a low-cost high-accuracy software-defined measurement architecture.

There are two ways to measure the network performance: active or passive methods [3]. Active measurement needs to inject probe packets into the network for monitoring their behaviours, such as popular ping and iperf. While passive measurement consumes little overhead, such as Simple Network Monitoring Protocol [4], NetFlow [5] and sFlow [6], which are widely used in networks. Active and passive measurements have their own advantages. However, active measurement may violate the network behaviour and produce a negative influence on the measurement accuracy. Meanwhile, passive measurement requires a number of network devices.

Flow-based measurements such as NetFlow and sFlow provide general support for various measurement tasks. On one hand, NetFlow was introduced by Cisco, which provided a method to collect statistical features about individual IP flows and collect IP network traffic. NetFlow produces per-flow statistics collection without requiring rules to be installed. A typical flow monitored by NetFlow consists of three main components: flow exporter, flow collector and analysis application. The popular sFlow is an industry standard technique for monitoring high speed networks. It provides fine grained network measurement without requiring per-flow state at switches. The advantages of sFlow include scalable technique, a low cost solution and providing a network-wide view of usage. However, NetFlow and sFlow consume two resources (CPU and bandwidth) [7] [8], which can increase network overhead. Therefore, we need to propose a low-cost measurement method, which supports various measurement tasks.

The rest of this paper is organized as follows. Section 2 introduces the background and related work. In Section 3, we present the architecture of the proposed measurement system, named OpenFlow-based Low-cost and Low-error measurement architecture (OpenLL), and then we also present the calculating delay, loss, throughput, a presentation of our proposed adaptive sampling algorithm and a low-cost topology discovery method. In Section 4, we evaluate and compare the performance with that of OpenNetMon and polling through simulations using Mininet and Matlab. In Section 5, we make a summary of this paper.

2. Background and Related Work

In this section, we briefly discuss the most related proposals, namely those about OpenFlow protocol and measurement methods in SDN. Some other recent works that utilize SDN for measurement can be found in [12] [13] [14] [15] [16].

A typical SDN architecture includes application layer, control plane and data plane. In SDN architecture, OpenFlow is the most popular and widely accepted open southbound interface standard. As an open source standard, OpenFlow protocol [2] ensures communication compatibility and scalability both switches and controllers, such as NOX [9], POX [10] and Floodlight [11]. Although, OpenFlow is not the only available southbound interface for SDN, it is regarded

as great popularity by the academia and industry. In OpenFlow protocol [2], packet-in messages are sent by forwarding devices to the controller when a new incoming flow or because there is an explicit "send to controller" action in the matched entry of the Flow Table. As a response, the controller will use Flow Table Modification messages (FlowMod) to process the entry of the Flow Table and resend the packet to the switch using a Packet-out message. FlowMod message also specifies whether the switch should send a Flow Removed message (FlowRemoved) to the controller when the flow expires.

Recently, OpenFlow methods to measure traffic have been proposed. OpenNetMon [12] presents an active approach and open-source software implementation to monitor per-flow metrics, especially throughput, delay and packet loss. Payless [13] provides a flexible RESTful API to monitor link utilization, which utilizes an adaptive statistics collection algorithm. FlowSense [14] proposes a passive monitoring method, which utilizes FlowRemoved and PacketIn messages to evaluate per-flow link utilization. Although, communication overhead for FlowSense is fairly low, its accuracy is comparatively low.

In other respects, OpenSample [15] proposes a sampling based SDN measurement platform. It utilizes sFlow packet sampling to offer near real-time measurement, which is implemented in the Floodlight OpenFlow controller. OpenSketch [16] proposes an active software defined traffic measurement architecture. It makes sketches more flexible in supporting various measurement tasks through high accuracy with low latency. However, OpenSketch is required to upgrade or deploy the network where a new protocol standardization has been applied to subsititute the popular OpenFlow protocol.

In summary, our target is to design a measurement framework supported by OpenFlow protocol in SDN, which can significantly reduce network overhead while guaranteeing measurement accuracy.

3. System Design and Problem

3.1. OpenLL Measurement Architecture

In this section, we first give an overview of OpenLL. The calculation formulas of throughput, delay, packet loss rate and a new adaptive sampling algorithm are presented thereafter. Besides, a low-cost topology discovery mechanism is introduced. OpenLL is an improved measurement framework based on OpenNetMon where we can measure per-flow throughput, delay and packet loss rate. OpenLL also polls edge switches (the first and the last switches). Compared to OpenNetMon, we present a new adaptive sampling algorithm, which can significantly improve the measurement accuracy and decrease the network overhead. We obtain throughput by the first and the last switches instead of the last switch, contributing to the measurement accuracy. Meanwhile, OpenLL adds a delay threshold value, which adjusts delay and improves the precision. In addition, we introduce a low-cost topology discovery mechanism to decrease the controller overhead. Last, we use multipath to consider network topology instead of simple single path.

Figure 1 shows OpenLL measurement architecture, which is implemented in POX controller. It includes four aspects: monitoring, forwarding, data storage and performance. Monitoring module is used to monitor and calculate data, such as the byte counters, packet counters, duration, throughput and delay. Forwarding module is used to process the forwarding of flow and find the shortest path. Monitoring and forwarding modules require OpenFlow discovery module, which achieves the network topology. Monitoring and forwarding modules are also the core of our proposed measurement architecture. Data storage is used to store the monitored data, mainly including throughput, delay, byte counters, packet counters and duration. Performance analysis includes throughput, delay and packet loss rate, which are most meaningful elements for network performance and scalability in SDN.

3.2. Delay, Packet Loss Rate and Throughput

Per-flow delay is defined as transmission interval from the origin node to the destination node. However, it's very difficult to measure path delay. What's more, the existing measurement techniques are mainly aimed at traditional networks, which may encounter all sorts of problems in SDN. For example, the popular ping sends ICMP packets to measure connected end-to-end path and round-trip time of the path. Although it can measure real-time path delay, the accuracy is quite inferior.

In this paper, we use iperf to send TCP packets from the client to the server through active measurement to generate traffic. The controller sends packet from the first switch, which probe packet can traverse the same path, and the probe packet returns to the controller through the last switch. The leaving time from the controller is T_A. The arriving time from the last switch to the controller is T_B. The round-trip time (RTT) from the controller to the first switch is RRT_1. RTT from the controller to the last switch is RRT_2. In addition, standard value of link delay is set to $\bar{t}$. This approach can relatively and accurately test link delay

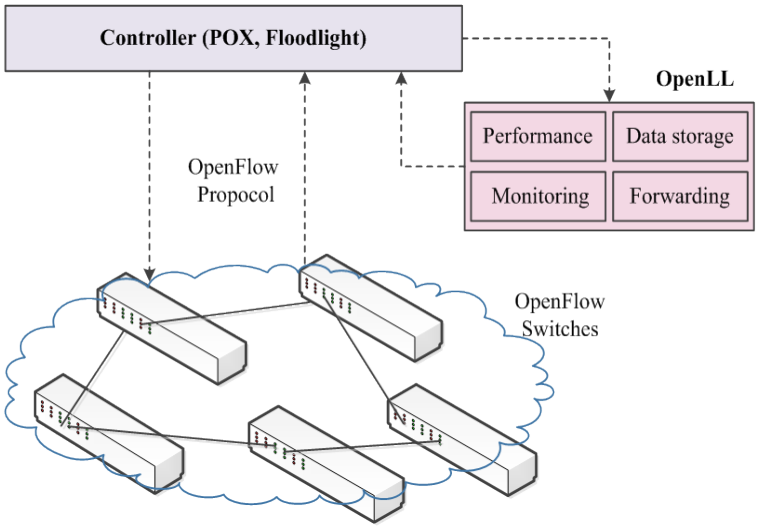

Figure 1. OpenLL measurement architecture.

of the experiment. However, it ignores the processed packet time on the switch. Our proposed method introduces a time threshold value ξ_2 to adjust time delay and improve the accuracy. We define per-flow delay as:

$$T_{delay} = T_B - T_A - \xi_1(RRT_1 + RRT_2) + \xi_2, \quad (1)$$

where ξ_1 is the coefficient of proportionality varying from 0 to 1 and ξ_2 ranges from $-\bar{t}$ to $\bar{t}$.

Packet loss is the number of lost packets and transmission ratio of the total number of packets within a certain interval. It is mainly caused by network congestion and transport link error. We calculate packet loss rate using packet counters of the first switch to minus packet counters of the last switch, and then to divide by packet counters of the first switch. Our proposed adaptive sampling algorithm can be contributed to obtaining a more accurate per-flow measurement of the packet loss. The formula of packet loss rate is defined as:

$$P_{loss} = (Packet_f - Packet_l) / Packet_f, \quad (2)$$

where $Packet_f$ is packet counters of the first switch measured by our proposed measurement architecture. $Packet_l$ is packet counters of the last switch measured by our presented measurement architecture.

Throughput is the number of successful data transmission within unit time, such as bits, bytes and grouping. It mainly depends on link rate, equipment port rate and the degree of network load, etc. To determine per-flow throughput, we adopt a similar approach with OpenNetMon. We poll edge switches (the first and the last switch), which can effectively reduce network cost. Compared to OpenNetMon, we calculate per-flow throughput through byte counters of the first and the last switch instead of byte counters of the last switch. On the one hand, compared to polling algorithm, it can promote network efficiency and decrease measurement overhead, especially in larger networks. On the other hand, compared to OpenNetMon, our proposed method may exert an active effect on some singular value, which can improve the measurement accuracy. Per-flow throughput is defined as:

$$B_t = \frac{1}{2}(B_f + B_l), \quad (3)$$

where B_f is measured byte counters of the first switch by OpenLL measurement architecture. B_l is measured byte counters of the last switch by OpenLL measurement architecture.

3.2. Adaptive Sampling Scheduling Algorithm

During flow transmission process, it is difficult for the existing sampling measurement methods to process the fluctuation of the flow, which can decrease the measurement accuracy and cause a waste of measurement resource. Therefore, our measurement architecture presents an adaptive sampling scheduling algorithm, which reduces network overhead while optimizing measurement accuracy. When flows change greatly, we will decrease the sampling interval. When flows keep a stable state, we will increase sampling interval. When fluctuation

among flows is the third situation, we will increase the small sampling interval. The algorithm details are shown in Algorithm 1, where T_{max} and T_{min} are interval upper and lower limit, respectively. We first consider the active flow in lines 1-3. After that, we utilize adaptive sampling method to adjust the interval in lines 6-14.

Algorithm 1: Adaptive Sampling Scheduling algorithm

```
1:   for flow in active flow:
2:      send a FlowStatsRequest to flow.switch;
3:   end for
4:   for dpid in FlowStatsReply event:
5:    diff_throughput = stat_throughput − prev_throughput
6:       if  diff_throughput < α * prev_throughput
7:           interval = min( interval * γ1 , Tmax )
8:           increace1.time =ture
9:       else if  diff_throughput > β * prev_throughput
10:          interval = max( interval/γ2 , Tmin )
11:          decreace.time =ture
12:       else
13:          interval = min( interval * γ3 , Tmax )
14:          increace2.time =ture
15:       end if
16:  end for
```

3.4. Topology Discovery

In SDN measurement, we need to update network state in real time, in particular its topology and link. Therefore, topology discovery is a critical component in SDN architecture. OpenLL measurement framework requires the topology discovery during complete measurement process. In this paper, we introduce a low-cost topology discovery approach [17] widely used POX controller platform in SDN. The remarkable advantage of this method is to guarantee the accuracy of the link discovery while decreasing the controller overhead.

4. Experimental Results and Discussion

In this section, OpenLL measurement framework is deployed on POX controller. We implement OpenLL compared it to OpenNetMon and periodic polling. We utilize Mininet to simulate a network consisting of hosts and OpenFlow switches and use Matlab to process the simulation data. Experiments are conducted on a computer with Intel(R) Core(TM) 2 Duo CPU processor and 4G RAM.

4.1. Experiment Setup

Table 1 describes the used software, function and version in all simulation experiments. We consider network topology in our experiments in **Figure 2**. Although it is not an excellent solution for POX controller to solve broadcast storm problem, OpneLL measurement architecture increases processing rules to ensure the normal operation of loop communication. TCP flows between hosts

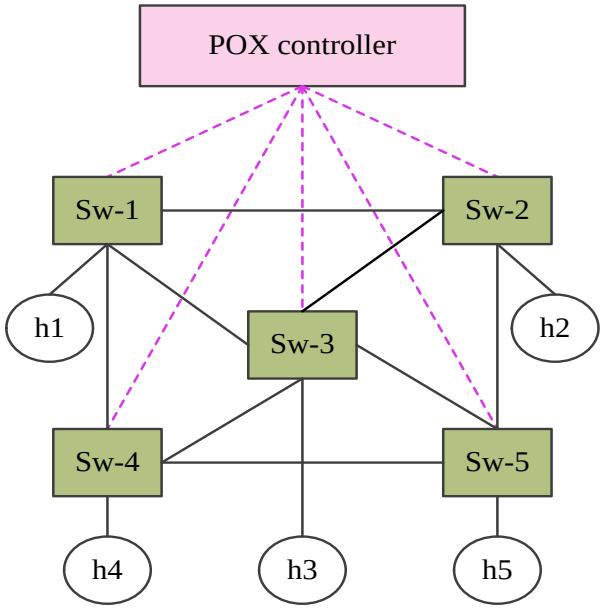

Figure 2. Network topology for experiment.

Table 1. Software used in experiments.

Software	Function	Version
Mininet	SDN simulation	2.21
OpenvSwitch	Switch	1.4.3
POX	SDN controller	Betta branch
VM	Virtual software	10.0.4
Ubuntu	Operating system	14.04
Python	Programming language	2.7
MATLAB	Programming language	2012a

are generated using iperf. The delay between switches is 5ms to emulate a WAN connection where all switches form a private WAN. Furthermore, the packet loss rate between all switches equals 1%. The bandwidth between switches is limited to 10Mbps. The minimum and maximum interval for our sampling scheduling algorithm is set to 1s and 30s. The parameter ξ_1 is set to 1/2. The parameters γ_1, γ_2 and γ_3 are set to 2, 1.5 and 1.125, respectively. The parameters α and β are set to 0.05 and 0.2, respectively.

4.2. Evaluation

Accuracy: Measurement accuracy is evaluated by root-mean-square error (RMSE) of throughput in our simulation experiments. It reflects the deviation between observed value and true value. What's more, the smaller the RMSE is, the higher the accuracy becomes. We define RMSE as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}\ , \tag{4}$$

where x_i is the throughput calculated by our measurement architecture. The parameter n is the polling times by our measurement architecture.

Overhead: As the core of SDN, the controller can encounter the performance bottleneck, which can produce a negative impact on the overall network performance and scalability. Therefore, it's essential to decrease the controller overhead. In this paper, network overhead is calculated by the number of FlowStatsRequest message sent from the controller.

4.3. Simulation Results

We use iperf to send TCP packets to model traffic from the client to the server. It is well-known that TCP is a connection-oriented, reliable transport layer communication protocol based on byte. Therefore, TCP packets drastically increase estimation accuracy for any given sampling ratio [18]. **Figure 3** shows the throughput from the client host *h1* to the server host *h5* by our proposed method and iperf. The throughput by OpenLL can reflect the similar variation trend with iperf, which is widely utilized in network performance analysis. By the new adaptive sampling algorithm, OpenLL measurement architecture can quickly adjust itself to successfully capture the traffic.

Figure 4 shows the delay comparison on the path from the first switch to the last switch. Because the delay of link equals 5ms and standard delay is set to 10ms from switch *s1* to switch *s5*. As is shown in **Figure 4**, compared with OpenNetMon, OpenLL is closer to the true value and has a lower delay. However, the delay measurement by OpenNetMon is much higher, which results in the inaccuracy. As we all known, delay is an essential part in network performance analysis. Thus, more accurate delay measurement can produce a significant effect on network management. On one hand, our measurement method introduces a time threshold value to adjust the time delay and improve the accuracy.

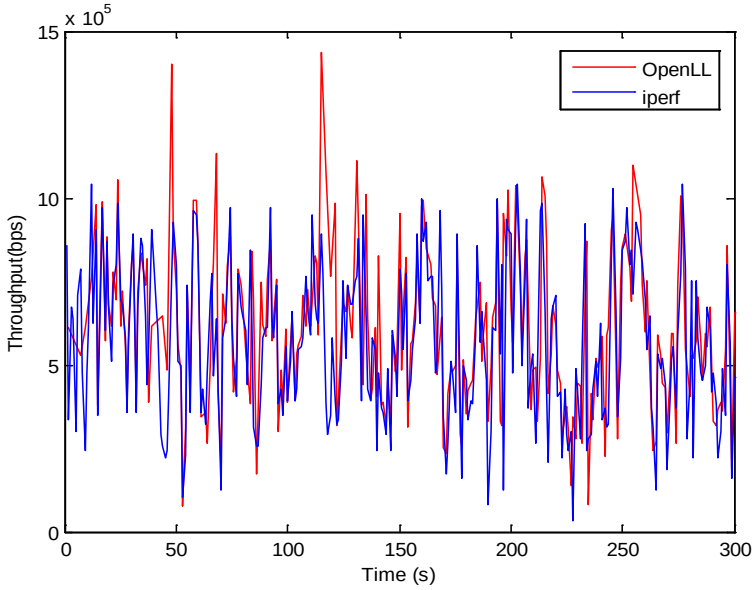

Figure 3. The comparison of throughput on the path.

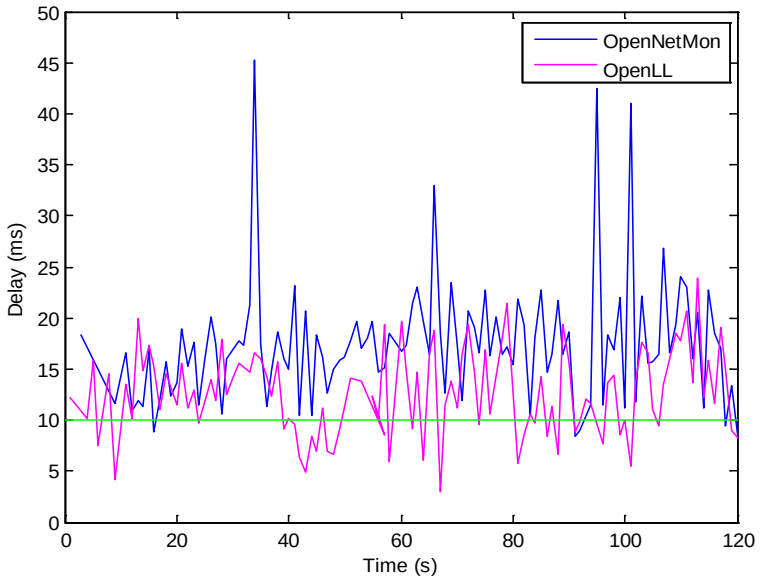

Figure 4. The comparison of delay on the path.

The parameter ξ_2 is set to −0.002 in simulation experiment. On the other hand, we can adjust ξ_2 when the delay error is large. To keep the parameter up to date, we also recalculate ξ_2 periodically.

The packet loss rate is calculated by using packet counters of the first switch to subtract packet counters of the last switch, and then dividing by packet counters of the first switch. The packet loss rate between all switches is set to 0.01. **Figure 5** shows the packet loss rate measured by OpenLL and OpenNetMon. The averages of packet loss rate measured from *s1* to *s5* using OpenLL and OpenNetMon are 0.02 and 0.017, respectively. RMSE of packet loss rate measured using OpenLL and OpenNetMon techniques are 0.0027 and 0.0034, respectively. Because of the better average and RMSE of the packet loss rate measured by our proposed method, packet loss rate of our proposed algorithm is closer to the true value. In conclusion, our proposed measurement framework can effectively improve the measurement accuracy by adaptive sampling scheduling algorithm.

Figure 6 shows the network overhead evaluated by the number of the Flow Statistics Request (StatsRequest) messages. The controller sends message to the switch with periodically queried StatsRequest messages. In return, the switch sends Flow Statistics Reply (StatsReply) messages to the controller. OpenLL monitoring module measures the byte counters, packet counters and duration of each flow, which can be used to calculate throughput, delay and packet loss rate. Compared to peridoic polling, OpenNetMon may have a lower network overhead. But, the network overhead proposed by our method is the lowest. In some cases, our algorithm sends out 6.14% less messages than that of OpenNetMon method and 39.03% less messages than that of periodic polling method. Our measurement framework plays a significant role in decreasing network overhead, especially in large scale network.

The balance between measurement overhead and accuracy is shown in **Figure 7** and **Figure 8**. **Figure 7** shows the effect of $T_{\min}$ on measurement over

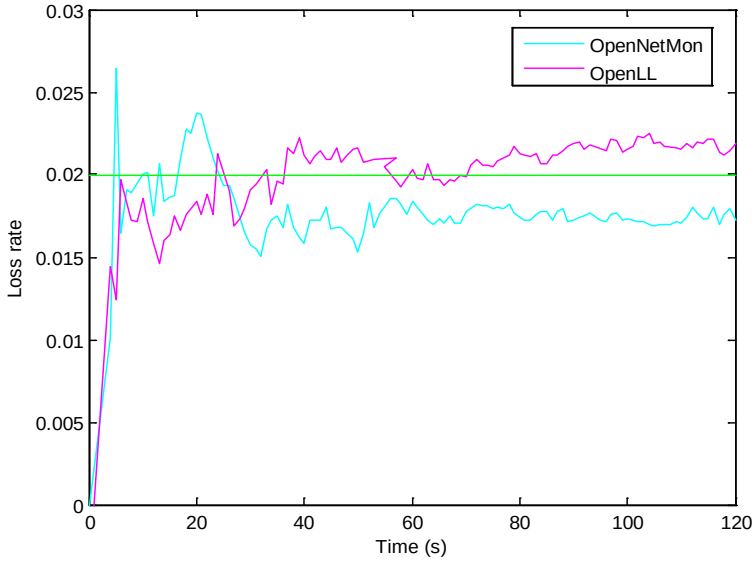

Figure 5. The comparison of packet loss rate on the path

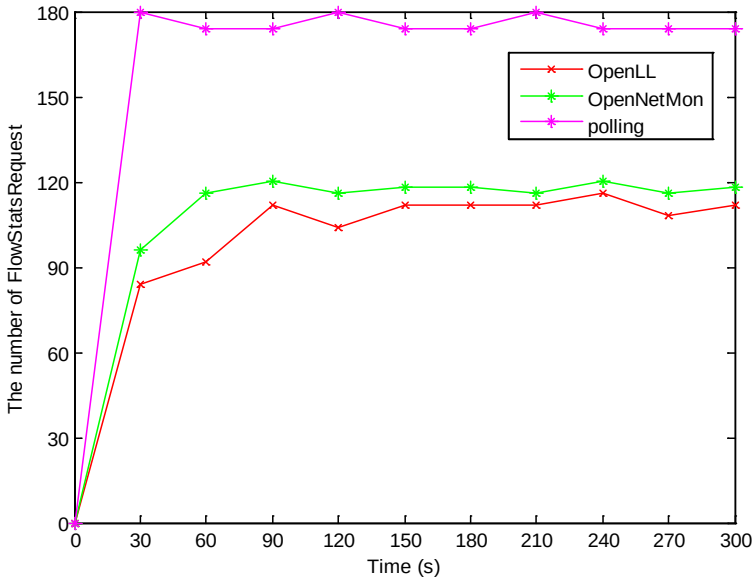

Figure 6. The comparison of network overhead.

head and accuracy. Compared to 2s and 3s in **Figure 7**, when $T_{\min}$ equals 1s, the network overhead is the maximum and RMSE is the minimum. Meanwhile, the smaller RMSE is, the much higher the accuracy becomes. This suggests that the improvement of accuracy is usually at the cost of network overhead. When $T_{\min}$ becomes lower, our adaptive sampling algorithm will decrease the sampling time (increasing the sampling rate), which increases the network overhead but improves the measurement accuracy.

Figure 8 shows the influence of time on measurement overhead and accuracy. With the increasing of time, the network overhead and RMSE increase slowly and eventually keep stable. At the beginning, the accuracy increases slowly. When time is more than 120 seconds, the network overhead and RMSE keep

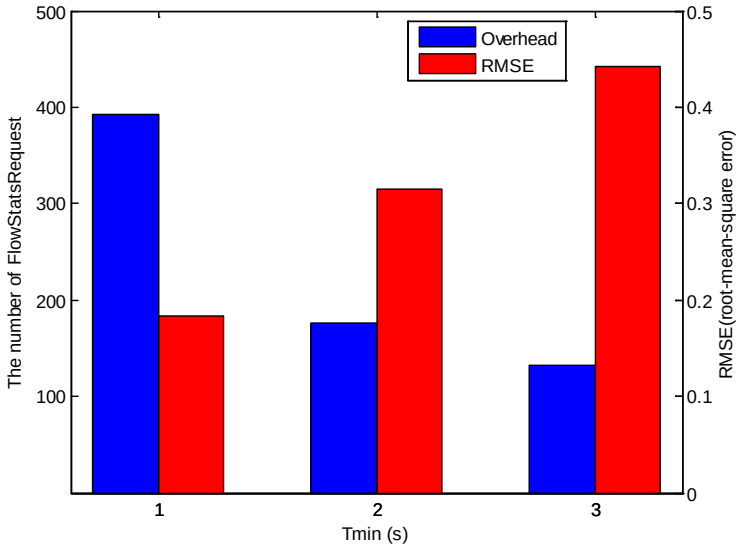

Figure 7. Overhead and measurement error.

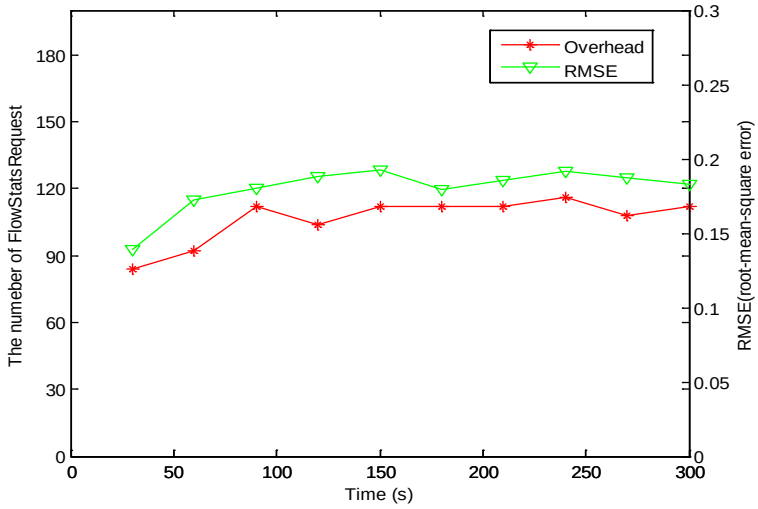

Figure 8. Overhead and measurement error.

stably. Results indicate that a large amount of time contributes to the improvement of measurement accuracy.

5. Conclusion

In this paper, we have presented OpenLL, an improved measurement framework based on OpenNetMon. It can calculate per-flow throughput, delay and packet loss rate. First, we only measure the edge switches (the first and the last switches), which can effectively lower down network overhead. In addition, we propose a new adaptive sampling algorithm, which contributes to the measurement accuracy. We also use a time threshold value ξ_2 to adjust the delay, which can produce a significant effect on precision of path delay. Furthermore, we consider a low-cost topology discovery algorithm and the balance between the network overhead and measurement accuracy in several aspects. Last, we utilize POX

controller to implement the proposed measurement framework. The effectiveness of our solution is demonstrated through simulations in Mininet and Matlab. Compared to periodic polling algorithm and OpenNetMon, experiment results show that OpenLL reduces roughly 39% and 6% of network overhead, respectively.

References

[1] Feamster, N., Rexford, J. and Zegura, E. (2013) The road to SDN. *Queue*, **11**, 20:20-20:40. https://doi.org/10.1145/2559899.2560327

[2] Open Networking Foundation (ONF), 2014. https://www.opennetworking.org/

[3] Mohan, V., Reddy, Y.R.J. and Kaplan, K. (2011) Active and Passive Network Measurements: A Survey. *Intern. Jour. Computer Sci. and Information Technologies*, **2**, 1371-1385.

[4] Presuhn, R. (2002) Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP). Internet Engineering Task Force, RFC 3416 (Internet Standard), Dec. http://www.ietf.org/rfc/rfc3416.txt

[5] Sampled NetFlow [Cisco IOS Software Releases 12.0 S]. http://www.cisco.com/en/US/docs/ios/120s/feature/guide/12s sanf.html

[6] sFlow.org Forum, 2012. http://www.sflow.org/

[7] Duffield, N., Lund, C. and Thorup, M. (2003) Estimating Flow Distributions from Sampled Flow Statistics. In *ACM SIGCOMM*.

[8] Estan, C., Keys, K., Moore, D. and Varghese, G. (2004) Building a Better Netflow. *ACM SIGCOMM*. https://doi.org/10.1145/1015467.1015495

[9] Gude, N., *et al.* (2008) NOX: Towards an Operating System for Networks. *Comput. Commun. Rev.*, **38**, 105-110. https://doi.org/10.1145/1384609.1384625

[10] McCauley, M. (2012) POX. http://www.noxrepo.org/

[11] Big Switch Networks (2013) Project Floodlight. http://www.projectfloodlight.org/

[12] van Adrichem, N.L.M., Doerr, C. and Kuipers, F.A. (2014) OpenNetMon: Network Monitoring in Openflow Software Defined Networks. In: *Proc. IEEE Network Operations and Management Symposium* (*NOMS*). https://doi.org/10.1109/noms.2014.6838228

[13] Chowdhury, S.R., Bari, M.F., Ahmed, R. and Boutaba, R. (2014) Payless: A Low Cost Network Monitoring Framework for Software Defined Networks. *Proc.* 14*th IEEE/IFIP Network Operations and Management Symposium*, *NOMS*'14, May 2014. https://doi.org/10.1109/noms.2014.6838227

[14] Yu, C., Lumezanu, C., Zhang, Y., Singh, V., Jiang, G. and Madhyastha, H.V. (2013) FlowSense: Monitoring Network Utilization with Zero Measurement Cost. In: *Passive and Active Measurement*, Springer, 31-44. https://doi.org/10.1007/978-3-642-36516-4_4

[15] Suh, J., Kwon, T., Dixon, C., Felter, W. and Carter, J. (2014) OpenSample: A Low-Latency Sampling-Based Measurement Platform for SDN. IBM Research Report, Jan. 2014. https://doi.org/10.1109/icdcs.2014.31

[16] Yu, M., Jose, L. and Miao, R. (2013) Software Defined Traffic Measurement with Opensketch. *Proceedings* 10*th USENIX Symposium on Networked Systems Design and Implementation*, *NSDI*, Vol. 13.

[17] Pakzad, F., *et al.* (2014) Efficient Topology Discovery in Software Defined Net-

works. 8*th Signal Processing and Communication Systems* (*ICSPCS*), Dec. 2014. https://doi.org/10.1109/icspcs.2014.7021050

[18] Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M. and Sherwood, R. (2012) On Controller Performance in Software-Defined Networks. *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services* (*Hot-ICE*), Vol. 54.