KSÜ Mühendislik Bilimleri Dergisi, 22, Özel Sayı, 2019 KSU J Eng Sci, 22, Special Issue, 2019
Uluslararası İleri Mühendislik Teknolojileri Sempozyumu (ISADET) Özel Sayısı
International Symposium on Advanced Engineering Technologies (ISADET) Symposium Special Issue
Araştırma Makalesi Research Article

Kahramanmaras Sutcu Imam University
Journal of Engineering Sciences

Geliş Tarihi : 22.07.2019 Received Date : 22.07.2019
Kabul Tarihi : 14.10.2019 Accepted Date : 14.10.2019

YAZILIM TANIMLI AĞLARDA İZLEME

MONITORING IN SOFTWARE DEFİNED NETWORKS

Hasan ÖZER [*1], *İbrahim Taner OKUMUŞ* [2]

[1] Kahramanmaraş Sütçü İmam Üniversitesi, Biyomühendislik ve Bilimleri Bölümü, Kahramanmaraş, Türkiye
[2] Kahramanmaraş Sütçü İmam Üniversitesi, Bilgisayar Mühendisliği Bölümü, Kahramanmaraş, Türkiye

*Sorumlu Yazar / Corresponding Author: Hasan ÖZER, hozer77@gmail.com

ÖZET

Ağ izleme ve ölçme, ağ yönetiminde çok önemli bir yere sahiptir. Yazılım Tanımlı Ağ (SDN), veri düzlemini ve kontrol düzlemini ayırarak ağ yönetimini kolaylaştırır. OpenFlow protokolü, veri ve kontrol düzlemi arasında güvenli bir kanalla iletişim kuran bir arayüz olarak kabul edilebilir. Aynı zamanda, OpenFlow, veri düzleminden farklı toplama seviyelerinde (örneğin akış, liman ve paket vb.) istatistik toplamamızı sağlar. Bu makalede, port bazında trafik ölçüm istatistiklerinin periyodik olarak ağ cihazlarından nasıl alınacağını ve SDN ortamında bu istatistiklerin ağa getirdiği iş yükünü nasıl hesaplanacağını gösterdik. Test ortamımızda, gerçek zamanlı trafik, periyodik trafik ve SDN'de tahmini trafik arasındaki ilişkiyi karşılaştırmak için farklı zaman aralıklarında toplanan istatistikler kullandık. Farklı zaman aralıklarında gerçekleştirilen ağ sorgulamalarında doğruluk ile ek yük arasında bir ilişki olduğunu gözlemledik. Ayrıca, istatistiklerin daha doğru yorumlanabilmesi için örnek tahmin tasarımı (SPD) ortaya koyduk. İzleme yapmak için Floodlight denetleyicisini kullandık. Sonuçlarımızın geçerliliği, Mininet'teki emülasyonlar yoluyla gösterilmektedir.

Anahtar Kelimeler: Floodlight Denetleyicisi, Mininet, Periodik Port İstatistik Toplama, SDN/OpenFlow, SPD

ABSTRACT

Network monitoring and measurement have a very important place in network management. Software Defined Network (SDN) makes network management easier by separating the data plane and the control plane. The OpenFlow protocol can be considered as an interface that communicates with a secure channel between the data and control plane. At the same time, OpenFlow allows us to collect statistics at different aggregation levels (e.g., flow, port and packet etc.) from the data plane. In this paper, we demonstrate how to get traffic measurement statistics at the port level periodically from network devices and also have calculated the overhead of this traffic measurement statistics in the SDN environment. In our test environment, collected statistics in different time intervals are used to compare relationship between real-time traffic, periodic traffic and predicted traffic in SDN. We have observed that there is a trade-off between accuracy and overhead in the network monitoring performed at different time intervals. A sample predict design (SPD) is introduced to be interpreted more accurately of the statistics. We utilize the Floodlight controller for monitoring. The validity of our results is shown through the emulations in Mininet.

Keywords: Floodlight controller, Mininet and Periodic Port Statistics Collection, SDN/OpenFlow, SPD

INTRODUCTION

Traffic engineering (TE) helps in using network resources effectively by dynamically analyzing, estimating and editing the behavior of the transmitted data (I.F. Akyildiz ve ark.,2014). Traffic Analysis/Characterization is a significant part of mechanisms for Traffic Engineering. Network monitoring and measurement mechanisms are particularly very important for determining traffic behaviors, detecting network errors, and timely intervention of network congestion.

SDN (I.F. Akyildiz ve ark., 2014; Open Networking Foundation, 2012) simplifies control of the network, provides central visibility including global network information and improves network programmability by separating the control plane from the data plane. The control plane generally arranges network states in a centralized or distributed manner through specific network policies. In addition, SDN applications take place in the application panel of SDN architecture. Applications communicate with control plane through Noth-bound APIs. The monitoring istatistics can be used quality of service (QoS), energy usage, resource utilization, security, anomaly detection and many types of network management and measurement. In addition, the data forwarding plane can operate programmable OpenFlow switches through SDN controller, and the switches can communicate with SDN the controller over South-bound Open APIs (e.g., OpenFlow protocol) (Open Networking Foundation, 2014). As a result, thanks to the communication between these planes, the SDN paradigm provides a global and holistic view of complex networks and thus it will allow network resources to be used more easily and efficiently.

OpenFlow (OF) protocol was created in 2008 at Stanford University and was developed under the Open Network Foundation (ONF) in 2008. SDN controller can program the control functionalities of switches on the network by using the OpenFlow protocol. These OpenFlow switches are managed by the policies to generate by the SDN / OF controlers. There are different versions of OpenFlow (OpenFlow Versions, 2016).

The goal of this study is to collect statistics at the port level from the OF switches in SDN environment and to generate a network status information by used this statistics. It is very important that the status information is obtained correctly and on time. Thanks to this information, network resources will be used more efficiently and flexibly. In addition, it was aimed to show the overhead brought to the system via the equations in the SDN environment and calculate them numerically. It is to show whether there is a trade-off between accuracy and overhead. It would helped to develop new network monitoring techniques according to this trade-off. A sample predict design (SPD) is proposed to be interpreted more accurately of the statistics.

RELATED WORK

Generally, network monitoring is both a neglected issue and being very important for network management. There are different approaches in network monitoring. For example, such as host-based latency measures Internet Control Message Protocol (ICMP) or network node-based queries through Simple Network Management Protocol (SNMP) (Marc Hartunga ark., 2017). In most SDN architectures are used existing streaming-based network monitoring tools that are available in traditional IP networks. For example, The most well-known is NetFlow (NetFlow, 2017) from Cisco, which collects sampled traffic statistics and sends them to a central aggregator. It uses probe methods that are installed at switches as special module. Sflow (Sflow, 2017), which uses a time-based sampling to query traffic information, is a different flow sampling method. However, these methods result in a substantial overhead when collecting statistics from an entire network of centralized controls. Therefore, it may not be an efficient solution to implement in SDN architectures.

OpenTM (A. Tootoonchian ve ark., 2010) is a query-based monitoring method that predicts the traffic matrix (TM) by querying a switch for OF networks. It periodically queries the switch on each active stream to collect flow level statistics. This design, despite its high accuracy, brings with it a high overhead. OpenNetMon periodically queries packet counters from the source and destination switches, which is appropriate for end-to-end measurement (N. L. Van Adrichem ve ark., 2104).

FlowSense (C. Yu ve ark., 2013) is a passive push-based monitoring method that uses control messages between controls and switches. It uses control messages for monitoring and calculates network usage without any additional overhead. It uses the PacketIn and FlowRemoved messages to estimate the utilization of the stream on each link. However, FlowSense cannot capture for sudden network fluctuations. This is insufficient in terms of the accuracy and timing of received statistics.

MONITORING IN SDN

To properly provision resources in SDN networks, it is important to monitor resources especially link utilizations to quickly adapt routing rules to changes in workload. There are two different measurement methods: active and passive (V. Mohan ve ark., 2011). In active measurement, agents create probe instructions to perform a network characteristic evaluation. Measurement values can provide instant network behavior information, but if this happens frequently, there is an increase in system load (Overhead). For example, the popular application ping uses ICMP packets to reliably determine end-to-end connection status and compute a paths round-trip time (D. Erickson, 2013).

Passive measurement methods does not generate any traffic to the network. Instead, it provides traffic statistics from network devices. Because the passive approach measures actual network traffic, it does not generate additional test traffic on the network. However, all this information is stored on these network devices, both causing security problems and leads to the installation of in-network traffic monitors that are not suitable for all networks and require large investments.

By using statistics collection mechanisms provided by OpenFlow specification, over a specific network topology, statistics from switches can be obtained. There are two basic types of statistics collection messages provided by OpenFlow:

STATISTICS REQUEST MESSAGE: Message for requesting statistical data for ports, flows, etc. from switches.

STATISTICS REPLY MESSAGE: Message for replying a statistics request by providing requested statistical data for ports, flows, etc.

The monitoring work in SDN is done by connecting the controller to all switches with a secure channel. The controller and switches use this secure channel through a TCP connection. The controller obtains real-time flow statistics from the available switches and combines the raw data by providing interfaces for the applications in the top layer.

An important issue here is how to make monitoring on the SDN environment. Firstly, statistics request messages are sent to the OF switches at fixed intervals, then the statistic reply messages which containing the amount of data and duration on at that time in network are send from OF switches to controller. The amount of data and duration content in statistic reply messages is called as sample. According to these samples from the switches, controller can generate new network status policies for network management. In addition, controller can determine new polling intervals by evaluating samples from switches.

In this study we used these two messages to get port level traffic measurement statistics from switches in an SDN network. In the next section we provide the details of the test environment and measurement setup.

TEST ENVIRONMENT AND EVALUATION RESULTS

Network topology in Fig.2 is used in our study. We used Mininet (The mininet platform, 2106) to setup the network topology with virtual hosts and switches. It is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Its hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking.

OpenFlow controllers (e.g., Floodlight (Floodlight, 2017), NOX (N. Gude ve ark., 2008), etc.) introduce a platform to write special network applications. Floodlight is one of the popular controller currently used in the SDN environment. In this sample test topology, we used Floodlight as a controller. Floodlight is a Java-based OF controller that supports physical and virtual OF switches. It is based on the Beacon application. Beacon (D. Erickson, 2013) was created by David Erickson at Stanford University.

OpenFlow is the basic SouthBound API of SDN. Nowadays, it is widely used both in the industry and in the academy. OpenFlow is an interface that enables communication between the control plane and the data plane.

Overhead is the additional load that the OF messages used in the SDN environment bring to the network system by queries made at certain time intervals. Overhead is calculated by using topology in Fig. 2. and timing diagram in Fig. 3. Making this calculation will have an important knowledge in developing new inquiry techniques.

To obtain all flow statistics, it is necessary to obtain statistics from the switches for each flow along the path and combine the results. We study is an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, v_2, ..., v_k\}$ is the set of switches and E denotes the set of links. Hence $k = |V|$ is the number of switches in the network. In order to calculate the overhead of statistics collection we need to analyze the sizes of statistics request and reply messages. Equation (1) is the size of the reply messages for k flow entry and Equation (2) provides total message sizes for both request and reply for a single switch. Here, $L_{Request}$ represents the size of the flow statistics request message, L_{Rh} represents the length of flow statistics reply message header, L_{Sfe} represents the size of reply message body of a single flow entry, p represent the port numbers of each switch. For a flow statistics reply message with p entries, the whole reply message length $L_{Reply(p)}$ is a linear function:

$$L_{Reply(p)} = L_{Rh} + p * L_{Sfe} \tag{1}$$

$$W_n = k * (L_{Request} + L_{Reply(p)}) \tag{2}$$

According to the OpenFlow specification, size of these messages are as follows: $L_{Request}$ = 122 byte, L_{Rh}= 78 bytes, L_{Sfe} = 96 bytes and n denotes the polling frequency.

Sample Predict Design (SPD):

In network management, accurate measures of network status are needed to aid planning, troubleshooting, and monitoring. For example, it may be necessary to monitor the bandwidth consumption of several hundred links in a distributed system to pinpoint bottlenecks (Edwin A ve ark., 2000).

Nowadays, there are many measurement sampling methods. Sampling techniques are used to examine the behavior of a member population according to a representative subset. In fact, samples are periodically taken at a constant interval. However, as shown in Figure 3, 4 and 5 under some heavy traffic loads, static periodic sampling may not be suitable for the monitoring task. For example, a long sampling interval provides sufficient accuracy when traffic is not dense or there is a low network load. However, in high traffic fluctuations, shorter sampling intervals are required to accurately measure network situations. In this case, it also imposes an excessive overhead on the network. Therefore, there is a need for adaptive inquiry technique that dynamically adjust the interrogation interval based on network fluctuations. Although adaptive inquiry techniques are not studied in this paper, prediction methods that we studied in this paper will provide insights to design efficient adaptive techniques in future studies.

In this section, we present a Sample Predict Design (SPD) to predict the traffic rate of the next sample. The SPD is calculation that is used to select the most representative examples of the population by polling and following the significant fluctuations in the measured variables.

Equations (3) provide the details of sample prediction. Here $S_{Samples}$ denotes past N samples where N is the latest sample received and $S_{Predict}$ denotes the estimated value of the next sample to be received.

$$S_{Samples} = S_1, S_2, S_3, ..., S_N$$

$$S_{Total} = \sum_{i=1}^{N-1} S_i$$

$$S_{Avg} = \frac{S_{Total}}{N-1}$$

$$S_{Predict} = (1-\lambda) * S_N + \lambda * S_{Avg} \tag{3}$$

Where $0 < \lambda < 1$ (e.g., $\lambda = 0.125$) when calculating $S_{Predict}$, by adjusting the value of λ the weight of the most recent sample and average of the previous samples can be changed. Lower λ will increase the weight S_N and decrease the weight of S_{Avg}.

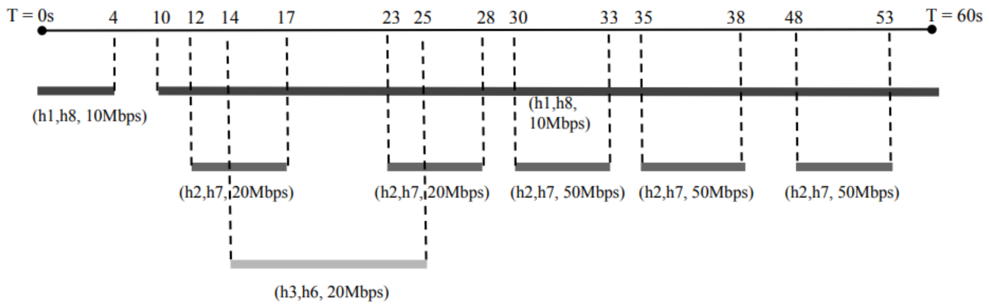

Figure 1. Timing Diagram of Experiment Traffic

To get the analysis results we have created a 3-level tree topology via Mininet and timing diagram shown respectively in Fig. 1 and Fig. 2. which are the same as Payless (S. R. Chowdhury ve ark., 2014) in this analysis. UDP flows for a total duration of 100s between hosts were generated using Iperf (Iperf: TCP/UDP Bandwidth Measurement Tool). Figure 1 shows the timing diagram; start, throughput and end time for each stream. In addition, we have paused traffic at different times between traffic streams to try different scenarios.

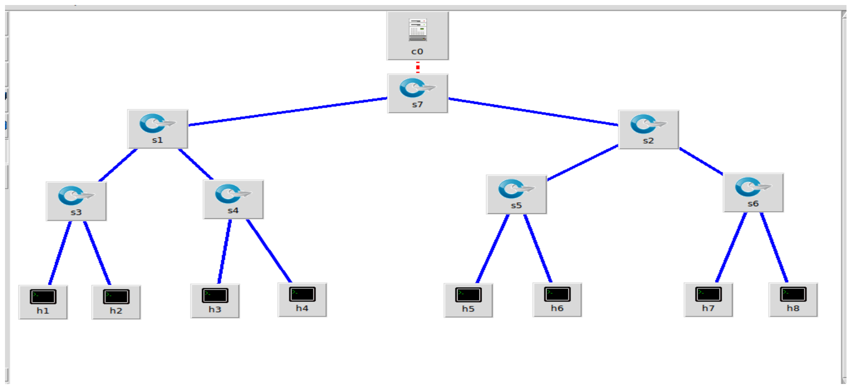

Figure 2. Generated Topology for Experiment

Overhead:

An important issue that needs to be explained here is how inquiries are made and the resulting network overhead (message overhead). OpenFlow defines a flow using fields from different layer (such as, layer 2, 3, 4) header of a package. When a switch receives a stream that does not comply with the rules in the routing table, it sends a PacketIn message to the controller. The control sends a FlowMod message to generate the necessary routing rules for the switches. The controller can specify idle timeout for a routing rule. This is referred to as the prologue of inactivity, after which a routing rule is removed from the switch. When this stream is removed, the switch sends a FlowRemoved message to the controller. This message contains the flow time and the number of bytes that match this flow input in the switch. In addition to these messages, the controller can send a FlowStatisticsRequest message to the switch for information about a particular flow. The switch sends the number of times and bytes for this stream to the controller in a FlowStatisticsReply message.

As shown in Table 1. we have considered the FlowStatisticsRequest message and the FlowStatisticsReply message as overhead. We calculated the overhead of the queries made at different frequencies by using topology created in Figure 2. We used the Equation (1) and (2) when calculating the overhead. The table shows us that the burden on the system is observed to be very high in the monitoring performed at the high interrogation intervals, while it is observed that the load coming to the system is less in the monitoring performed at lesser interrogation intervals.

$$L_{Reply(p)} = L_{Rh} + p * L_{Sfe} \quad (1)$$
$$L_{Reply(p)} = 78 + (96 * 20) = 1998byte$$
$$W_n = k * (L_{Request} + L_{Reply(p)}) \quad (2)$$
$$W_n = 7 * (122 + 1998) = 14840byte$$

According to Figure 2, $p = 20, k = 7$.

Table 1. Overhead at Different Time Intervals According to (1) and (2) Equations for Figure 1.

Polling frequency	**Overhead (byte) according to total duration of 100 sec**
1 sec	$W_1 = 14840 * 100 = 1484000byte$
5 sec	$W_5 = 14840 * 20 = 296800byte$
10 sec	$W_{10} = 14840 * 10 = 148400byte$

Figure 1. In the time diagram shown, the UDP stream of 10 mbps between host1 (H1) and host8 (h8), the UDP stream of 20 and 50 Mbps respectively between the host2 (H2) and the host7 (h7), the host3 (H3) and the main host6 (h6) 20 Mbps UDP stream traffic is generated. To see fluctuations in traffic, flows are paused at specific time intervals. The traffic generated according to the timing diagram in Figure 1. is monitored with 1 second, 5 second and 10 second monitoring frequencies. The results are shown in Figure 3, 4 and 5. Different flow times are used to test accuracy and performance.

Figure 3, 4 and 5 show that the overhead in the system is higher in the polling which are done frequently and the effect on accuracy increases. On the other hand, when we conduct polling interval over long time intervals, the overhead it brings to the system is less, but the effect on accuracy is low. This is a situation we don't want. The relationship between the value of the periodic queries and the real-time monitoring value at different time intervals was determined graphically in Figure. 3, 4 and 5. These graphs show us that a high-frequency inquiries in the SDN environment, a close real-time monitoring is seen. On the other hand, the overhead that it brings to the system is observed to increase. As a result, we have observed that there is a trade-off between accuracy and overhead in the network monitoring performed at different time intervals.

According to these analyzes, we need a design that will act according to our traffic density. In this case, if the traffic density is increasing we will need to use frequent query intervals, if not, we should use less query interval. Thus, we can stabilize both the system overhead and the system accuracy. We also think that using the $S_{Predict}$ equation in the SPD can help with these adaptive monitoring designs. As shown in Figures 3, 4 and 5, we have taken close to polling periodically.

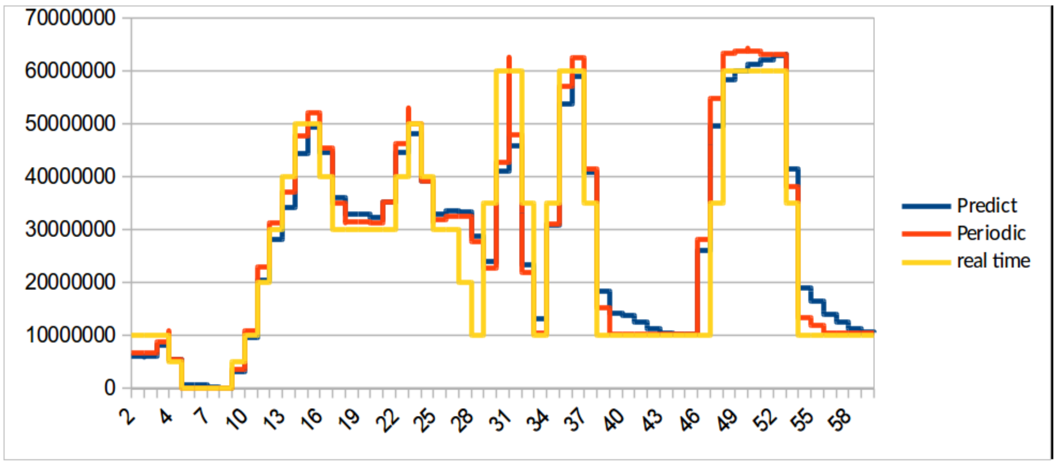

Figure 3. 1 sec Monitoring Frequency

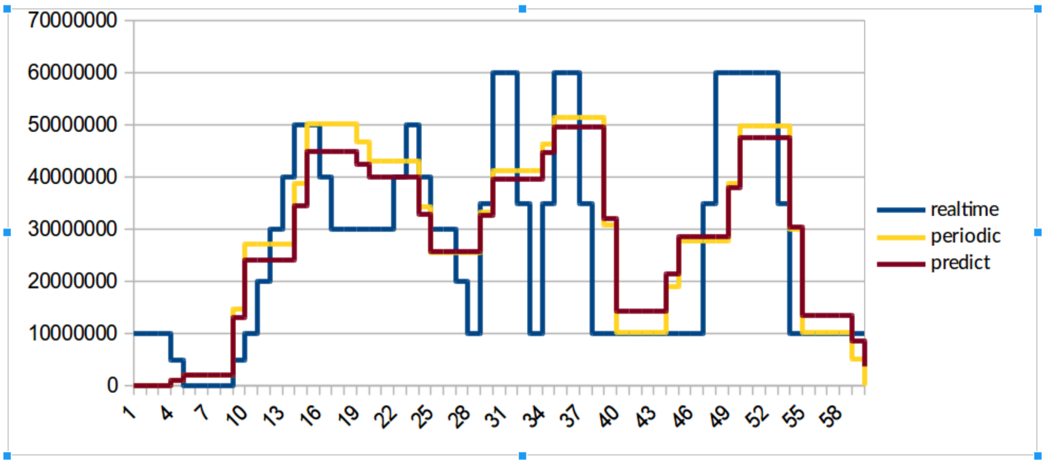

Figure 4. 5 sec Monitoring Frequency

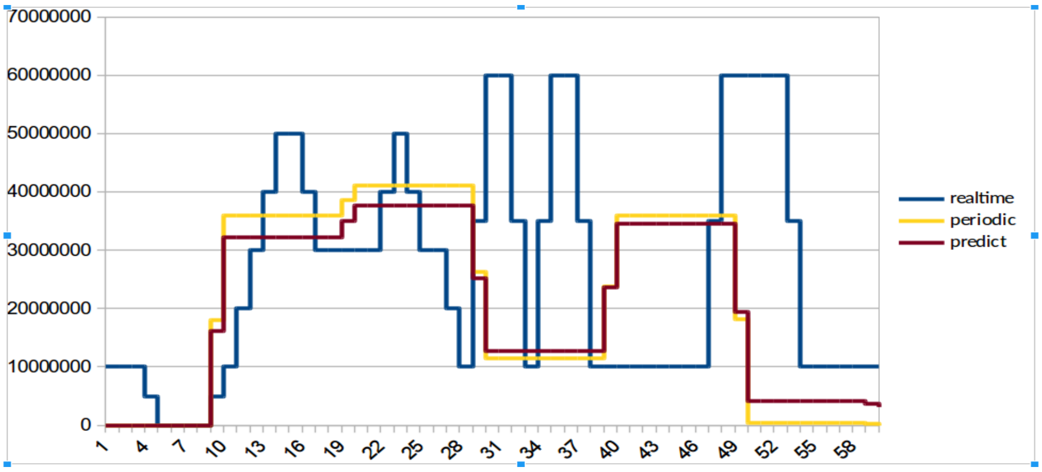

Figure 5. 10 sec Monitoring Frequency

CONCLUSIONS AND FUTURE WORK

In this article, we get periodically traffic measurement statistics at the port level from network devices in an SDN environment. The calculation of the overhead of these monitoring is shown by equations and calculated numerically. As a result of the analysis, we have shown that there is a trade-off between the overhead and the accuracy. At the same time, we present our SPD design, which could help new adaptive monitoring methods.

Our future study is to develop an adaptive inquiry techniques that can be adapted to the intensity of traffic using our SPD design.

REFERENCES

I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, “A roadmap for traffic engineering in SDN-OpenFlow networks,” Computer Networks, vol. 71, pp. 1–30, 2014.

Open Networking Foundation: OpenFlow Switch Specification version 1.5.0 (2014). https://www.opennetworking.org/software-defined-standards/specifications

OpenFlow Versions, updated Apr 26, 2016.
https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343541/Releases+and+Roadmap

Marc Hartunga, Marc K¨ornerb "SOFTmon - Traffic Monitoring for SDN" Procedia Computer Science 110 (2017) 516–523.

Open Networking Foundation: Software Defined Networking: The new norm for networks, White paper (2012).

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

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

A. Tootoonchian, M. Ghobadi, Y. Ganjali, Opentm: traffic matrixestimator for openflow networks, in: Proceedings of the 11thInternational Conference on Passive and Active Measurement,PAM'10, April 2010, pp. 201–210.

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

C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, H.V. Madhyastha,Flowsense: monitoring network utilization with zero measurementcost, in: Proceedings of the 14th International Conference onPassive and Active Measurement, PAM'13, March 2013, pp. 31–41.

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

D. Erickson, The Beacon OpenFlow Controller. In Proceedings of the ACM Workshop on Hot Topics in Software Defined Networks (HotSDN), Hong Kong, China, 12–16 August 2013; pp. 13–18.

The mininet platform, Retrieved March 20, 2016. http://mininet.org/ .

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

N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an operating system for networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 3, pp. 105–110, 2008.

Edwin A. Hernandez, Matthew C. Chidester, and Alan D. George, "Adaptive Sampling for Network Management,“ journal of Network and Systems Management, Vol.9, No. 4, December 2000.

S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "PayLess: A Low Cost Network Monitoring Framework for Software Defined Networks," in NOMS, 2014

Iperf: TCP/UDP Bandwidth Measurement Tool. http://iperf.fr/

ORCID

Hasan ÖZER http://orcid.org/0000-0002-0729-676X
İbrahim Taner OKUMUŞ http://orcid.org/0000-0001-9495-3133