MAKÜFEBED

ISSN Online: 1309-2243
http://dergipark.org.tr/makufebed
https://doi.org/10.29048/makufebed.1003502

Mehmet Akif Ersoy Üniversitesi Fen Bilimleri Enstitüsü Dergisi 13(1): 9-26 (2022)
The Journal of Graduate School of Natural and Applied Sciences of Mehmet Akif Ersoy University 13(1): 9-26 (2022)

Araştırma Makalesi / *Research Paper*

A Scalable and Efficient Port-Based Adaptive Resource Monitoring Approach in Software Defined Networks

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

[1]Kahramanmaras Sutcu Imam University, Department of Bioengineering and Sciences, Kahramanmaras, Turkey
[2]Kahramanmaras Sutcu Imam University, Department of Computer Engineering, Kahramanmaras, Turkey

Geliş Tarihi (Received): 07.10.2021, Kabul Tarihi (Accepted): 01.02.2022
Sorumlu Yazar (Corresponding author): hozer77@gmail.com*
+90 344 3001700 *+90 344 3001602*

ABSTRACT

The need for communication tools is increasing from the past to the present, therefore the tools and methods in network technologies are evolving. Depending on this development, network scale and complexity increase and the limitations of traditional network technology are surfaced. Software-defined Network (SDN) provides a variety of opportunities for the management and optimization of these challenges. Network resource monitoring is very important for providing information to network applications. This study introduces an adaptive port-level bandwidth monitoring method designed for SDN networks. The proposed approach uses a polling-based monitoring paradigm. There is a trade-off between measurement accuracy and monitoring overhead. With this adaptive method, it is aimed to decrease the overhead while maintaining an acceptable level of accuracy of the measurements and also to use network resources more efficiently. The proposed adaptive monitoring approach has 46% less overhead than the periodic polling method and 6.7% less overhead than the PayLess approach. At the same time, this approach is 5.4% more accurate than the periodic polling approach.

Keywords: Adaptive monitoring, floodlight, SDN, SPD

Yazılım Tanımlı Ağlarda Ölçeklenebilir ve Verimli Bir Port Tabanlı Adaptif Kaynak İzleme Yaklaşımı

ÖZ

Geçmişten günümüze iletişim araçlarına olan ihtiyaç artmakta, bu nedenle ağ teknolojilerindeki araç ve yöntemler gelişmektedir. Bu gelişmeye bağlı olarak ağ ölçeği ve karmaşıklığı artmakta olup geleneksel ağ teknolojisinin problemleri ortaya çıkmaktadır. Yazılım Tanımlı Ağ (YTA), bu zorlukların yönetimi ve optimizasyonu için çeşitli fırsatlar sunar. Ağ kaynaklarının izlenmesi, ağ uygulamalarına bilgi sağlamak için çok önemlidir. Bu çalışma, SDN ağları için tasarlanmış uyarlanabilir bir bağlantı noktası düzeyi bant genişliği izleme yöntemini amaçlamaktadır. Önerilen yaklaşım, yoklama tabanlı izleme paradigmasını kullanır. Ölçüm doğruluğu ve izleme yükü arasında bir denge bulunmaktadır. Bu uyarlamalı yöntemle, ölçümlerin kabul edilebilir doğruluk seviyesini korurken ek yükün azaltılması ve ayrıca ağ kaynaklarının daha verimli kullanılması amaçlanmaktadır. Önerilen uyarlamalı izleme yaklaşımı, periyodik yoklama yöntemine göre %46, PayLess yaklaşımına göre ise %6,7 daha az ek yük elde edilmiştir. Aynı zamanda bu yaklaşım, periyodik yoklama yaklaşımına göre %5,4 daha doğru bir ölçüm sağlamıştır.

Anahtar Kelimeler: Uyarlamalı izleme, floodlight, YTA, SPD

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

INTRODUCTION

Network management has a very important role in network systems. Network monitoring is vital in obtaining the network resource information required for network management. This information will enable more efficient and effective use of network resources. The statistics at different collection levels (i.e., port, flow) must be presented to network management applications in an accurate and timely manner. For example, network status statistics must be provided accurately and timely for QoS requirements, anomaly detection, and topology updates.

Traditional network monitoring methods can be classified as passive and active sampling-based methods (Huang et al., 2016; Terzi et al., 2017). Active sampling-based methods bring network load (overhead) to the network but are successful in achieving high accuracy. However, passive sampling-based methods do not bring an overhead into the network, but they do not succeed in obtaining accuracy (Mohan et al., 2011).

Monitoring methods should collect, process and transmit the desired statistics at the specified collection level and frequency, without bringing unnecessary monitoring overhead to the network. At the same time, these methods should be able to capture network traffic statistics in a timely manner. Therefore, there is a need for monitoring methods that reduce the overhead while maintaining accuracy level of network state.

Software Define Networking (SDN) paradigm is a recently emerged architecture that has been widely studied by researchers and used by vendors. By separating the control plane and the data plane, SDN proposes a global view on network systems and flexibility in network management. The data plane consists of OpenFlow (OF) switches (Open Networking Foundation, 2015). A controller works as the control plane.

The OF protocol can be considered as an interface between the data plane and the control plane in SDN. It allows communication with a secure channel between a switch and a controller. OF can specify a flow using the fields from the layer 2, layer 3 and layer 4 headers of a packet. A controller can query OF switches to collect statistics of active streams on the data plane through this protocol. OF protocol allows us to collect statistics at different aggregation levels (i.e., flow, port) from the data plane.

In SDN design, packet forwarding mechanisms are discussed in (Yang et al., 2021). On closer inspection, routing table entries in SDN nodes are identified and classified considering wildcard rules, their priority, validity, placement in multiple tables and integration of traffic statistics.

In this article, we present an adaptive monitoring approach designed for SDN. First component of the proposed system is prediction component which estimates the next measurement sample and let the developed adaptive method, adjust polling period proactively. Second, to maintain accuracy level and reduce the overhead we adaptively adjust polling period considering multiple parameters apart from the estimated measurement. Basically, if the traffic level is increasing or decreasing at a certain level that specific line needs to be monitored more closely which results in decreasing the polling period. This will increase the number of polling messages. If the traffic level is stable within certain boundaries, then monitoring period can be increased which will decrease number of polling messages. We implemented the proposed adaptive monitoring approach at the port statistics level in an SDN environment. According to the test results, it is seen that the adaptive monitoring approach reduces overhead by more than 56.7% compared to the periodic monitoring approaches.

Overheads are recognized as a major concern for SDN controllers; therefore, many studies aim to reduce monitoring costs. Typically, these studies use a trade-off between statistical accuracy and tracking overhead, so accuracy is compromised to reduce monitoring overhead. For example, transmission sampling (Li et al., 2019), statistical hash (Yu M. et al., 2013), and statistics estimation (Liu et al., 2016) pose problems with accuracy, bypassing the collection of some statistics samples.

Network monitoring is an important research area. As monitoring requirements change, different monitoring designs and methods are needed. Traditional network monitoring designs basically have four different functions: Collecting, aggregating, analyzing and storing statistics. Gathering statistics is the most important step. Because the analysis will be done according to the collected statistics. Monitoring approaches can be grouped into two types: distributed frameworks (Phan et al., 2017) and centralized frameworks (Shah et al., 2016).

NetFlow and sFlow (Huang et al., 2016; Terzi et al., 2017) use traditional network monitoring techniques in IP networks. NetFlow is widely used in network

management. It offers very low overhead since it uses a passive measurement method. However, because it provides full access to network devices, it causes privacy and security concerns (Su et al., 2015). Netflow requires significant pre-license and installation costs to integrate into a network. Another monitoring method is sFlow, which is offered by InMon as an open standard. It uses the time-based sampling technique to obtain network status information. In sFlow, the distribution is more specialized, but not adopted by vendors.

Today, various network monitoring studies are available on SDN networks. OpenSketch (Yu M et al., 2013) includes three-stage pipeline architecture. This architecture store data by integrating hash functions, classification and a counting table. OpenSkech uses static random-access memory (SRAM) instead of TCAM on the switches to store all counters. Thus, it is cheaper and more energy efficient. At the same time, offers a good trade-off between accuracy and usage of memory resources.

OpenTM (Tootoonchian et al., 2010) is a query-based monitoring method that predicts the traffic matrix (TM) by querying switches on OF networks. It periodically queries a switch on each active stream to collect flow level statistics. This design, despite its high accuracy, results in a high overhead.

OpenNetMon periodically queries packet counters from the source and destination switches, which is appropriate for end-to-end measurement (Van et al., 2014). The solutions offered by OpenNetMon are not universal and are specific only to certain applications.

Sirali-Shahreza and Ganjali proposes a method that offers packet-level information to the controller for flow monitoring by sending part of the packets to the controller based on a sampling method (Shirali-Shahreza and Ganjali, 2013).

FlowSense (Yu et al., 2013) is a passive push-based monitoring method that uses control messages between controller and switches. Push-based mechanisms consist of collecting measurements asynchronously. When a flow ends, the switch sends an OFPT FLOW REMOVED message containing flow statistics to the controller. This message also contains flags that indicate if the expiration was caused by either the idle or the hard timeout (José and Pere, 2014). FlowSense uses control messages for monitoring and calculates network usage without any additional overhead. It uses the FlowRemoved messages to estimate the utilization of the stream on each link. However, FlowSense cannot capture sudden network fluctuations. This is insufficient in terms of the accuracy and timing of received statistics.

FlowCover (Su et al., 2014) uses a polling schema optimizer through polling decisions, which reduces the cost of communication between the controller and the switches. By utilizing the full network visibility and central control features of SDN, FlowCover performs monitoring based on the number of active flows by selecting target switches instead of monitoring each stream. Although FlowCover basically reduces communication between target switches and controllers, it is insufficient in statistical aggregation methods. More precise and accurate monitoring designs are needed.

PayLess (Chowdhuryand et al., 2014) is a monitoring framework that is designed for SDN environment. This framework proposes an adaptive scheduling algorithm. However, PayLess can adapt unnecessarily when there are traffic bursts on the network. In this case, it can reach unwanted overhead level in the network system. Doing so may make the network inoperable. Another drawback of PayLess is that link capacity is not taken into account in the adaptive monitoring approach. There is no point in frequent monitoring unless the utilization of link capacity is below a certain rate. Therefore, there is a need for a new and effective adaptive monitoring method to solve the aforementioned problems.

IPro (Castillo et al., 2020), a traffic monitoring architecture using RL, which focuses on the problem of control plane overheads and extra additional CPU usage of the SDN controller. IPro uses Reinforcement Learning to determine the probing interval.

TSNu (Balasubramanian et al., 2021) framework was suggested in to deal with the admission control, routing, and scheduling at the network level in an SDN environment. It focuses on system reconfiguration owing to traffic type changes and reduces the network reconfiguration problem to a network utility maximization problem as the rate stability constraints meanwhile affect the network reconfiguration and utility.

As a result, these studies increase accuracy at the expense of an increase in network resources and costs, or vice versa, reducing overhead. IPro (Castillo et al., 2020) is focused on control plane overhead using RL (Reinforcement Learning). At the same time, Payless (Chowdhuryand et al., 2014) is an adaptive design that has been worked on overhead. However,

in our design, different parameters were used for more efficient use of network resources, unlike these designs. By using these parameters, a more balanced system has been presented in terms of overhead and accuracy.

The rest of this article is structured as follows. Related work is provided in section 2, proposed adaptive monitoring approach is presented in section 3, analysis and test results are provided in section 4 and section 5 is the discussion and concluding remarks.

MATERIAL AND METHOD

Material

In traditional network architectures, the fact that the data plane and the control plane have an integrated structure poses a problem in both resource utilization and network management. In contrast, in SDN architecture, network policies and management are centralized by a controller. Thus, it provides a global view of the network and provides more flexibility, easier manageability and effective resource usage compared to traditional networks.

By using statistics collection mechanisms provided by OF switches (Open Networking Foundation, 2015), over a specific network topology, statistics from switches can be obtained. There are two basic types of statistics collection messages provided by OF:

> STATISTICS REQUEST MESSAGE: Message for requesting statistical data for ports, flows, etc. from switches.
>
> STATISTICS REPLY MESSAGE: Message for replying to a statistics request by providing requested statistical data for ports, flows, etc.

An OF switch maintains counters for each port, flow table/entry, queue, group, group bucket, meter and meter table. It also features the ability to combine multiple streams into a group and monitor aggregate statistics to track statistics for multiple streams. Table 1 shows the Per Port counters for the OF protocol (Open Networking Foundation, 2015).

Table 1. Counters

Per Port		
Counter	**Bits**	
Received Packets	64	Required
Transmitted Packets	64	Required
Received Bytes	64	Optional
Transmitted Bytes	64	Optional
Receive Drops	64	Optional
Transmit Drops	64	Optional
Receive Errors	64	Optional
Transmit Errors	64	Optional
Receive Frame Alignment Errors	64	Optional
Receive Overrun Errors	64	Optional
Receive CRC Errors	64	Optional
Collisions	64	Optional
Duration (seconds)	32	Required
Duration (nanoseconds)	32	Optional

OF messages allow us to continuously query statistics using *port_stat_request* messages from each switch for each switch port. Switches return port statistics using *port_stats_reply* message. This message contains all the implemented counter values shown in Table 1. In order to follow link capacity usage, we follow Received Bytes and Transmitted Bytes portions. Information provided with Received Bytes is the number of bytes received by that port from the epoch time, which is the time when that port starts functioning. By using two consecutive statistics information and the message interval, it is possible to calculate number of bytes received and transmitted within that time interval. This information naturally does not provide us with instant link utilization at every instant of time within that interval but the average utilization. Thus, by comparing the values of multiple different samples taken at a given frequency, it is possible to follow the link usage of the link connected to that port.

Accurate measurement of network resources has an important place in network management. For example, as network traffic flows through hundreds of links and network devices, a congestion or collapse in them may cause the network to become inoperable and crash completely if it is not properly and timely intervened. Using network monitoring techniques, the network

resources must be allocated in a balanced manner according to the bandwidth consumption on each link. There are two types of active monitoring techniques (Hernandez et al., 2001). One is conventional sampling and the other is adaptive sampling technique. Conventional sampling is a traditional sampling technique that is used to monitor and collect samples with a fixed frequency. In such a sampling technique, if the sampling frequency is decreased, it cannot accurately capture the fluctuations in the network thus accuracy reduces, and if it is increased, it results in too much overhead because high frequency will cause too many probe messages and replies to appear on the network with the benefit of increased accuracy. Therefore, this sampling technique is not very efficient and scalable in a dynamic and quickly changing environment. In the adaptive sampling technique, sampling frequency is dynamically adjusted according to the sampling data monitored.

In adaptive monitoring, monitoring frequency is based on the status of the current network load. More specifically, when traffic load is low on the network there is no need to monitor with high sampling frequency. In this case, since the link utilization is low, high accuracy is not vital, low frequency will decrease the number of polling and reply messages which means reduced overhead. If the link utilization is high, accurate measurement becomes important. In this case, the sampling frequency will be increased to obtain more sampling data for accuracy and close monitoring of changes in the traffic. Developed algorithms determine how and when to change the sampling frequencies to monitor the network resources.

Many monitoring techniques have been introduced, but many have not demonstrated the effect of link capacity on sampling techniques and have not considered the efficiency and scalability of the developed approaches. For example, if the ratio of existing network traffic to link capacity is low, we do not need to monitor closely. However, if the link capacity reaches a critical threshold, we should be able to monitor resource usage more closely to quickly react to the changes in the network and plan resource allocations more effectively.

The adaptive approach takes a decision to correctly set the polling interval regarding overhead. Figure 1 shows the working principle of our adaptive approach. If we explain in more detail. 1- Control Plane receive statistical information from the data plane in a specific polling interval. Since this collected information closely affects the network status information, a different situation of the network occurs. 2- Management plane extracts these statistics to determine a new network condition by analyzing the current overhead. 3- Management plane sends decision plane to decide this new network status. 4- Decision plane takes such a state to calculate the reward. Based on this incoming reward, the decision plane sets a new polling interval to minimize overhead. 5- Decision plane transmits this new polling interval to the Control plane. The Control panel applies this polling interval, which affects the new network status. Figure 2 introduces and details the Adaptive Approach architecture.

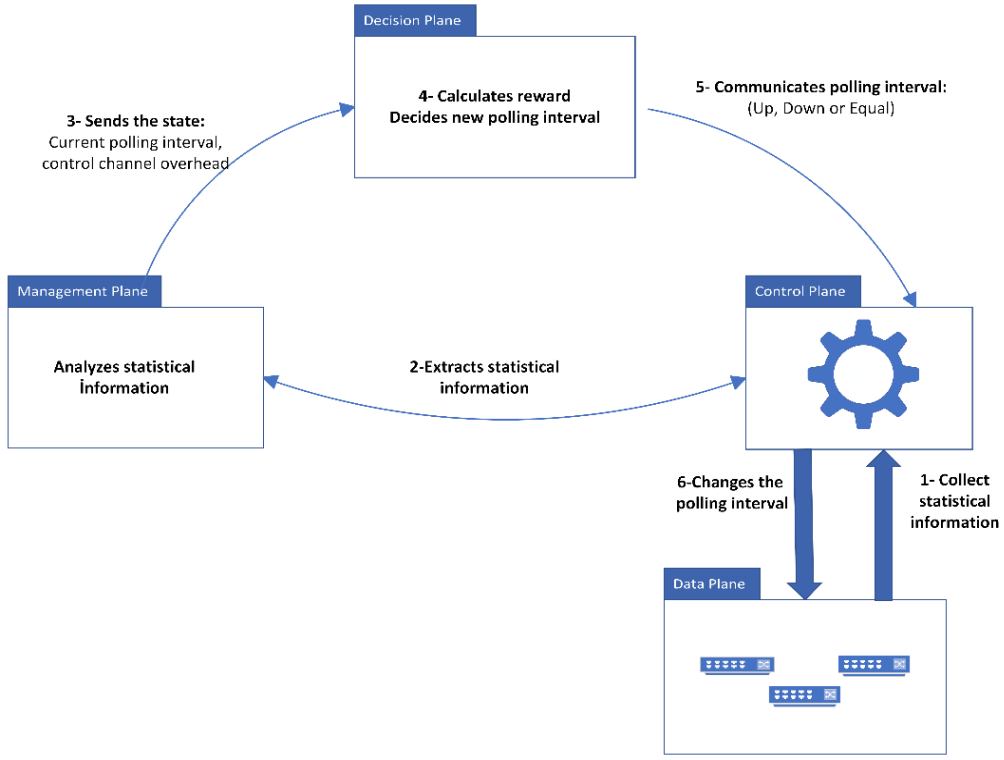

Figure 1. Adaptive approach high-level operation

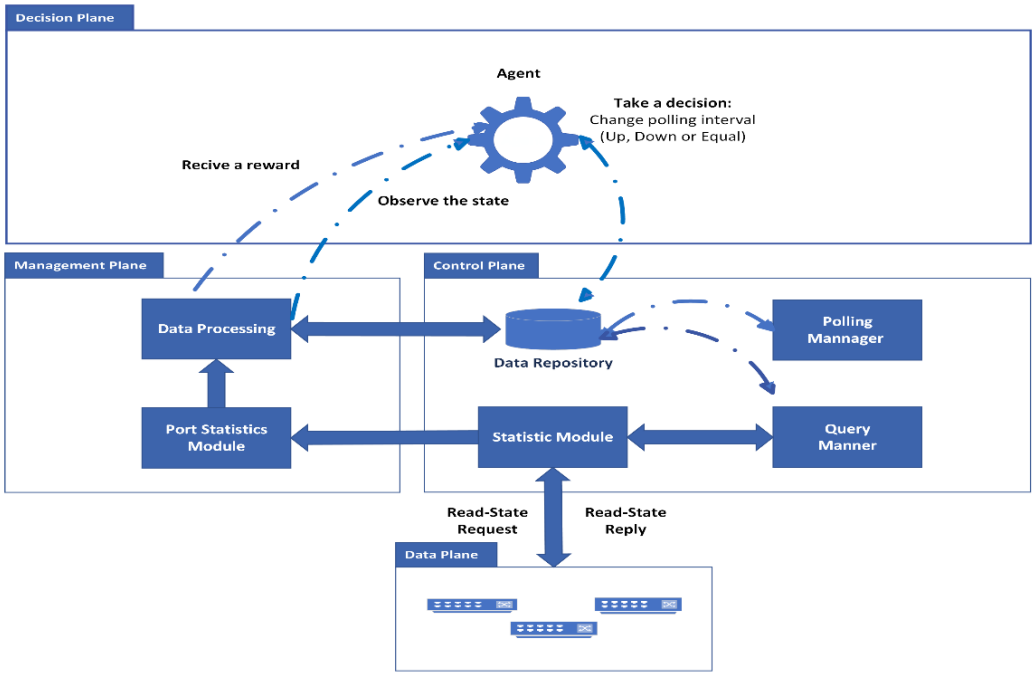

Figure 2. Adaptive approach architecture

Method

We present a scalable and efficient adaptive Port-level network resource monitoring approach for SDN networks. In our model switch set is denoted as vertices, with k=|V|. The switches form the forwarding plane of the SDN network. Topology is modeled as a $G = (V, E)$, where E is a set of links connecting switches.

In this approach, we follow active statistics collection paradigm and collect statistics periodically by sending requests to switches and getting replies with statistics data. We modeled our system to work with port level statistics.

In adaptive monitoring method, the first step is to estimate the value of the next measurement sample that will be received during the next polling sequence. Based on this information polling frequency will be determined. Studies showed that in active monitoring there is a trade-off between accuracy and overhead (Özer and Okumuş, 2019). Prediction based adaptivity allows the system to proactively adjust the polling frequency which will let the system compensate sudden burst while decreasing the overhead. For this purpose, we present a Sample Predict Design (SPD). Equation. 1 provides the calculations of sample prediction. There are n active samples in the network $S_{Samples} = S_1, S_2, S_3, \dots, S_n$. Here S_n 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. In this calculation, we used a moving average filter to calculate S_{Avg}. In this filter, considering that n last samples are present, 1 to n-1 of these samples are used. Prediction is made based on S_{Avg} and the latest sample obtained which is S_n. α-filter is used for $S_{Predict}$ calculation.

$$S_{Avg} = \frac{1}{n-1}\sum_{i=1}^{n-1} S_i \qquad (1)$$
$$S_{Predict} = (1-\lambda) \times S_n + \lambda \times S_{Avg})$$

In Equation 1, where λ is the weight factor $0 < \lambda < 1$ (e.g., λ = 0.125). 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 of S_n and decrease the weight of S_{Avg}.

The second step is to incorporate the rate of change of traffic into the adaptive method. Because this rate will enable us to monitor the instant movement of network traffic. This rate will help us decide how fast we need to monitor the traffic so that we can respond to sudden changes in the network conditions more quickly. In Equation.2, we calculated the ratio of traffic change (*F)*. The *F* shows us whether there is a significant traffic fluctuation on the link or not. Here L_c denotes the link capacity. The important point here is that we consider $S_{Predict}$ and S_{Avg} when calculating the rate of traffic change with respect to the link capacity because it has a more representative effect. Thus, unnecessary polling will be prevented as a result of short time bursts.

$$F = |\frac{S_{Predict} - S_{Avg}}{L_c}|) \qquad (2)$$

We could determine our sampling period adaptively according to the traffic change ratio F, but one of our important purposes is to add the effect of the link capacity to our adaptive sampling method. If we explain in more detail, the traffic change ratio F gives us information about the current traffic trend, whether the traffic is stable or changing (increasing or decreasing). However, this ratio is of no importance if the utilization of the link is low. For example, for a link that has a capacity of 1 Gbps, 100 Mbps traffic utilization does not require close monitoring of that link. However, if we do not consider link utilization in adaptive monitoring, traffic fluctuations on a low utilized link will cause an increase in polling frequency which in turn leads to increased overhead and reduced efficiency. Therefore, if we establish a weighted relationship between the link utilization and the ratio of traffic change, we will adjust polling interval more realistically, which is necessary for our adaptive design's scalability and efficiency. As a third step, this relation is provided in Equation 3. L_c denotes the link capacity and L_{ratio} denotes the predicted link utilization for the next polling period. Again, by using an α-filter we incorporate link utilization and traffic fluctuation in the calculation of F' which is the main indicator to adjust the polling frequency in presented adaptive approach.

$$L_{ratio} = (\frac{S_{Predict}}{L_c}) \qquad (3)$$
$$F' = \beta \times F + (1-\beta) \times L_{ratio})$$

Where β is the weight factor $0 < \beta < 1$ (e.g., β= 0.5). By adjusting the value of β, the weight of the ratio of traffic change and current link utilization ratio can be changed.

OF port statistics request is sent to a switch and switch returns the statistics for each active port on it. Depending on these values, polling frequency will be determined. However, since the statistics for each port will be different there will be different polling frequencies for each port on a single switch. Since it is not feasible to send statistical request messages separately for each port in terms of overhead and scalability., we need to employ a mechanism to reduce the polling frequency to a single one that will be used to poll the statistics from that switch for all the ports. Since we need to monitor the most highly utilized and highly fluctuating ports closely, we chose to select the port that has the maximum utilization and maximum fluctuation to calculate the polling frequency and that frequency will be used to get statistics from that particular switch. F' is calculated separately for each port. Since F' is the main parameter to determine next polling frequency, and higher F' indicates high utilization and high fluctuation, we select the highest F' among the ports and use that as F'_{sw} to calculate switch polling frequency (Equation.4)

$$F'_{sw} = MAX(F') = (F'_i : F'_i \geq F'_j, i \neq j \forall i, j \in n) \quad (4)$$

Where F'_i is the value for *i*th port and is the F'_j value for *j*th port and F'_{sw} is the maximum of those port F' values selected to represent switch.

To decide the value of the polling interval we follow the change in F'_{sw}. This value is the difference between previously calculated F'_{sw} which will be denoted as F'_{swold} and current F'_{sw}:

$$\Delta F'_{sw} = |F'_{swold} - F'_{sw}| \quad (5)$$

Equation.5 shows us the trend of the traffic, whether it is increasing, decreasing or stable. If the traffic trend is increasing or decreasing, we need to monitor traffic more closely. However, the main idea of the adaptivity is to decrease the monitoring frequency, if the traffic is stable. However, in order to prevent unnecessary fluctuations in polling frequency with small increase and decrease of the value of $\Delta F'_{sw}$ we use a buffer ($\Delta F'_{swTh}$) so that if the change in the traffic is below a threshold value, we can treat the traffic as stable.

Table 2. shows the parameters and conditions to adjust the sampling period. There is a limit on the minimum and maximum value of the polling period. T_{min} denotes the minimum value of polling period and T_{max} denotes the maximum value of polling period. $T_{current}$ denotes present polling period and T_{next} is the polling period that will be used for the next polling sequence. If the link utilization is below $L_{threshold}$ we don't need to monitor the link closely and polling period is set to be the maximum value T_{max}. If link utilization is above $L_{threshold}$, then we need to look at the trend of F'_{sw} which is denoted as $\Delta F'_{sw}$. If $\Delta F'_{sw}$ is below $\Delta F'_{swTh}$ then traffic on the link is stable and we can loosen the monitoring frequency which means getting fewer samples. To achieve this, we multiply $T_{current}$ with ω, where $0 < \omega < 10$, to gradually increase polling period until it reaches T_{max}. If $\Delta F'_{sw}$ is above $\Delta F'_{swTh}$, that means traffic is either increasing or decreasing rapidly which means that we need to monitor link utilization more closely so that we can more accurately follow the actual traffic on the link. To achieve this, we divide $T_{current}$ with φ, where $0 < \varphi < 10$, to gradually decrease polling period until it reaches T_{min}. Table 2 summarizes the conditions and actions to be taken and adaptive monitoring algorithm is given in Adaptive Monitoring Algorithm.

Table 2. Sampling period condition

Link Ratio	Traffic Fluctuation	Adaptive Calculation
$L_{ratio} < L_{threshold}$	-	$T_{next} = T_{max}$
$L_{ratio} > L_{threshold}$	$\Delta F'_{sw} \geqslant \Delta F'_{swTh}$	$T_{next} = max(T_{min}, 1/\varphi \times T_{current})$
	$\Delta F'_{sw} < \Delta F'_{swTh}$	$T_{next} = min(T_{max}, \omega \times T_{current})$

Adaptive Monitoring Algorithm:

$T_{current} = T_{max}$
while true
    port_stat_request to v_i
    port_stat_reply from v_i // Message contains statistical information;
    Store S_i
    count_samples++
    if count_samples < n **continue** //wait to get n samples before starting calculations
    else //Samples are ready, start adaptive calculations
        $S_{Avg} = (S_1 + S_2 + S_3 + \ldots + S_n)/(n-1)$
        $S_{Predict} = (1-\lambda) \times S_n + \lambda \times S_{Avg}$
        $F = (S_{Predict} - S_{Avg})/L_c$
        $L_{ratio} = S_{Predict}/L_c$
        $F' = \beta \times F + (1-\beta) \times L_{ratio}$
        if $L_{ratio} < L_{threshold}$ **then** // Link usage is below threshold
            $T_{next} = T_{max}$
        end if
        else //Link usage is above threshold
            if $\Delta F_{sw} < \Delta F'_{swTh}$ **then** //Traffic change is below threshold
                $T_{next} = \omega \times T_{last}$
                if $T_{next} \geqslant T_{max}$ **then**
                $T_{next} = T_{max}$
                end if
            end if
            else $T_{next} = 1/\varphi \times T_{last}$ //Traffic change is above threshold
                if $T_{next} < T_{min}$ **then**
                $T_{next} = T_{min}$
                end if
            end else
        end else
        end else
        $T_{current} = T_{next}$ // Polling period is set
        Wait for $T_{current}$ seconds
end while

FINDINGS

Experiment Setup

For the experiments, we used Mininet (The Mininet Platform) to set up a network topology with virtual hosts and switches. Mininet is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Hosts in Mininet run standard Linux network software, and switches support OF. There are various open-source controllers available (e.g., Floodlight (Floodlight, 2017), NOX (NOX, 2008), POX (POX, 2017), etc.). Floodlight is one of the popular controllers currently used in the SDN environment which is a Java-based OF controller that supports physical and virtual OF switches. In the test setup, we used Floodlight as a controller.

We used two different topologies and traffic patterns for two different scenarios. In the first scenario, the topology and traffic pattern used is the same the one used in PayLess (Chowdhuryand et al., 2014) to be able to compare the results of two studies. Iperf is used for traffic generation. UDP traffic is used for the tests. UDP traffic maintains a steady traffic rate. Thus, since we know exact traffic rate at certain instance, we will have a chance to compare the measurement value with the rate of the actual traffic. TCP traffic behavior is not predictable and steady. Traffic rate changes according to the network conditions and it will not be

possible to know exact traffic rate at a certain instance. In the tests we set the initial polling interval to 5s. T_{min} and T_{max} polling intervals are set to 1s and 5s respectively. We tested fixed period polling and adaptive polling approaches. For the fixed period polling case polling interval of 1s is used. We used n, number of past samples to be used in prediction as 6 and set ω and φ to 2 and 6 respectively. The reason to use a larger fixed value for φ is because it adapts better to the sudden changes in traffic.

Evaluation metrics used at the test cases are accuracy and overhead. Accuracy is the similarity between the actual link usage in the Network and the link utilization measured in the SDN environment with different measurement methods. Overhead is calculated from the number of FlowStatisticsRequest messages sent from the controller to switches.

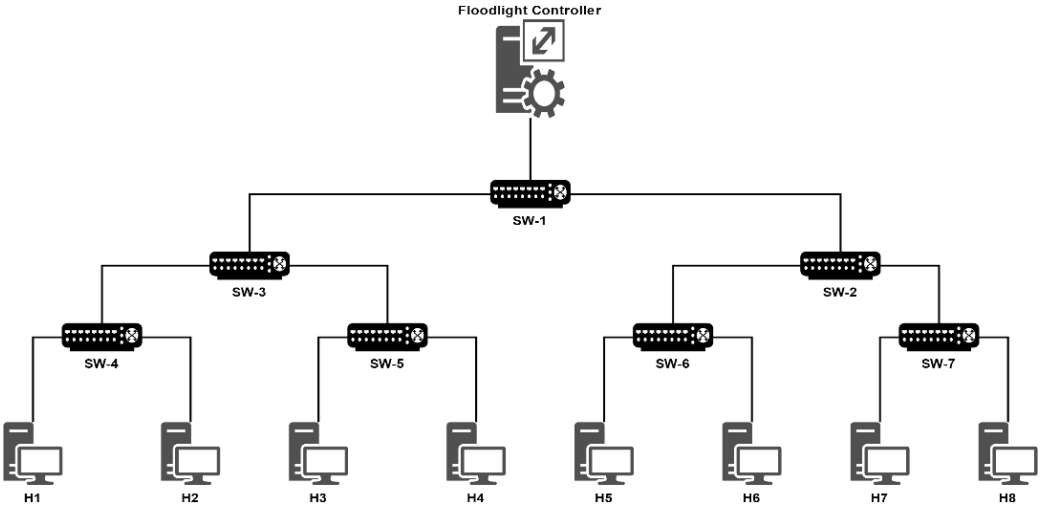

Figure 3. Topology 1

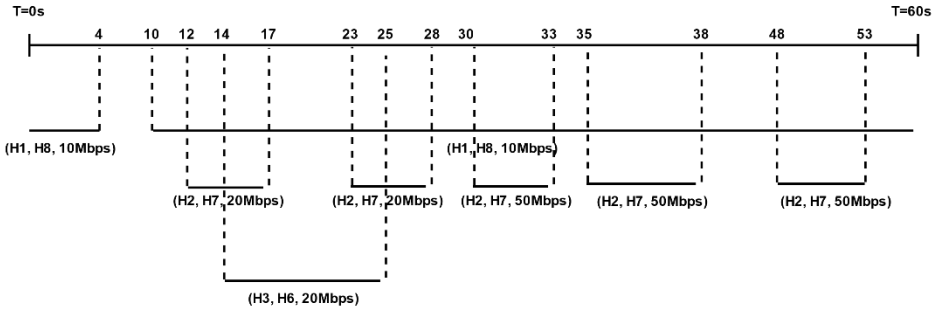

Figure 4. Traffic timing diagram1

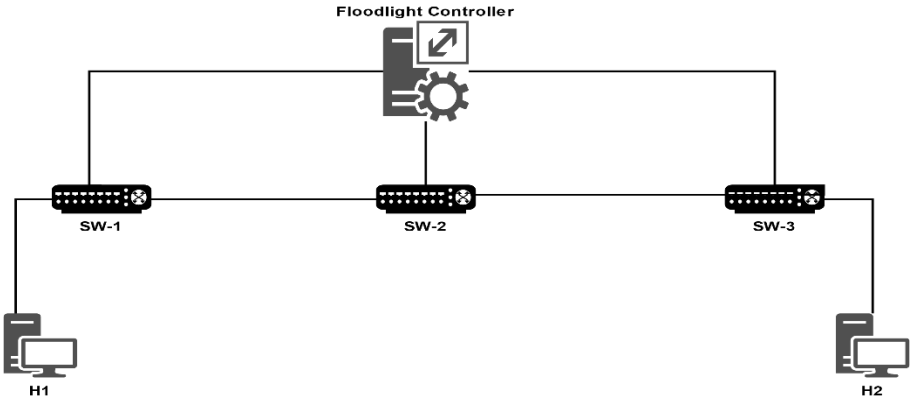

Figure 5. Topology 2

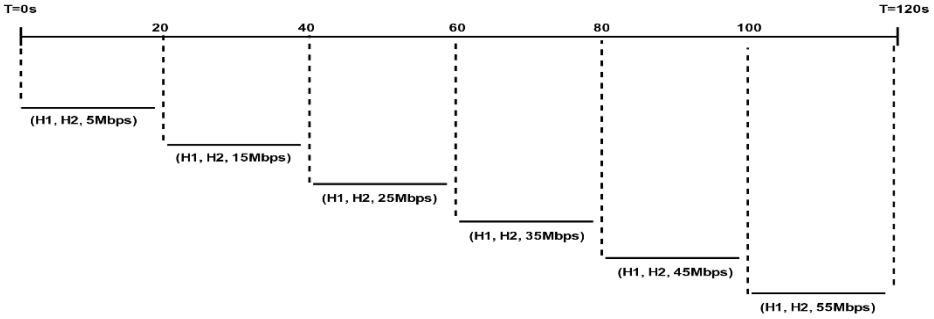

Figure 6. Traffic timing diagram 2

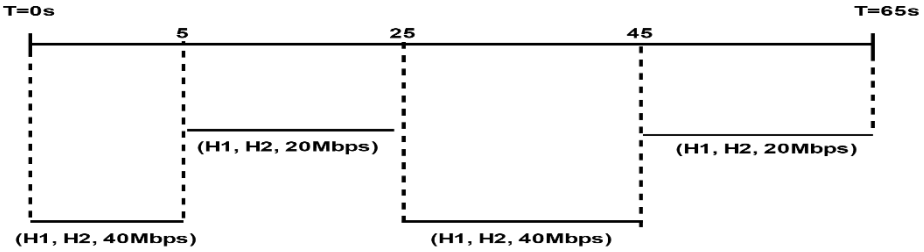

Figure 7. Traffic timing diagram 3

Findings of Survey

Prediction Phase

Since the algorithm starts with predicting the next sample value, we start with analyzing the effect of λ (Eq1) on the accuracy of prediction ($S_{Predict}$). Table 4 and Figure 8 shows the average error and overhead rate of $S_{Predict}$ for different λ values. Results show that smaller λ results in more accurate estimation however it results in more overhead which shows the trade-off between accuracy and overhead. λ value can be used to adjust the amount of trade-off between accuracy and overhead. In order to see the effects of other parameters on more accurate measurement, the λ value is used as 0.125 in all scenarios.

Table 3. Effect of λ on $S_{Predict}$ error rate

Values of λ	Error rate (%)	Overhead
0.125	12	39
0.325	32	29
0.525	62	25
0.925	82	10

In order to fully demonstrate the effect of λ value on $S_{Predict}$, sudden fluctuations in network traffic have been addressed. In network traffic that is already stable, $S_{Predict}$ closely follows the actual traffic. If the traffic fluctuation is high, it takes time to adapt to the actual traffic which causes prediction error.

Figure 9 shows the actual measured values from the switches and the prediction values calculated. These values show us how close the estimated values are with the actual values. Thus, it will enable us to achieve significant accuracy for our adaptive design.

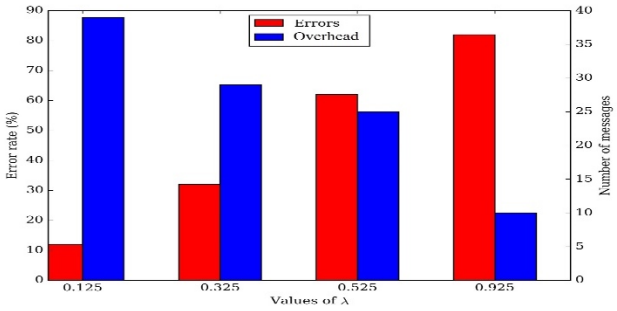

Figure 8. Effect of λ on $S_{Predict}$ error rate (%) and overhead

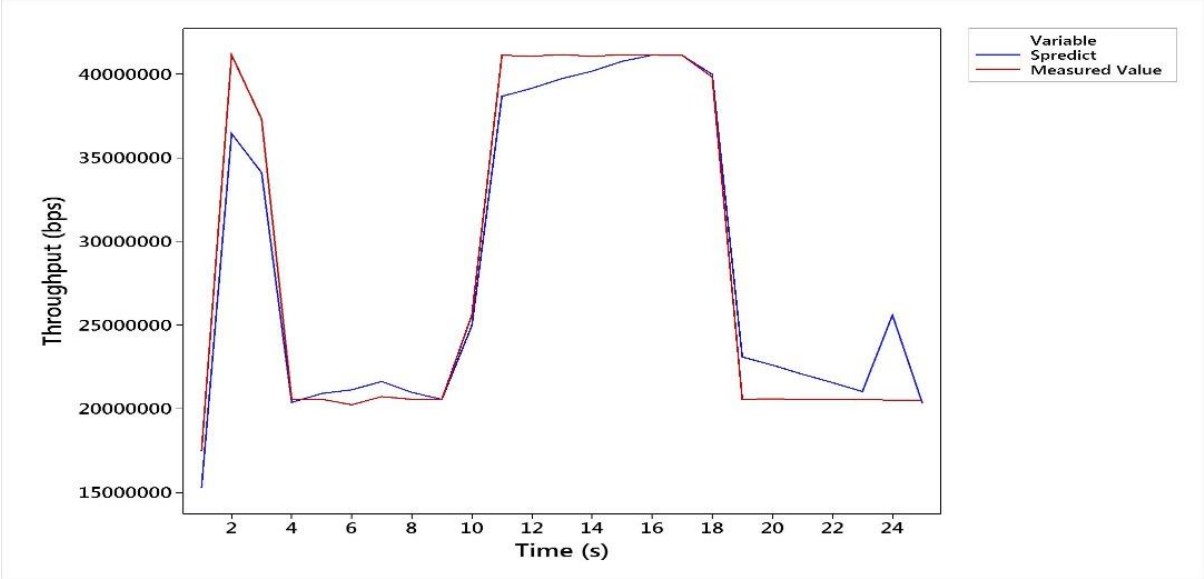

Figure 9. Utilization Measurements

Scenario 1

Our aim in this scenario is to observe the error and overhead effect of $\Delta F'_{swTh}$ value. To get the results we have created a liner-topology as shown in Figure 5 and the traffic timing diagram as shown in Figure 7 UDP flows for a total duration of 65s between hosts were generated.

Table 5 and Figure 10 shows the error and overhead effect of the $\Delta F'_{swTh}$ value. As it can be seen from Figure 11, with each traffic fluctuation, higher $\Delta F'_{swTh}$ value causes higher error. In contrast, also from Table 5 and Figure 10, number of polling messages decreases with higher $\Delta F'_{swTh}$ value. It is clear from these results that there is a trade-off between accuracy and overhead depending on the value of $\Delta F'_{swTh}$

Table 5. The effect of $\Delta F'_{swTh}$ on overhead and error

Values of $\Delta F'_{swTh}$	Error rate (%)	Overhead
0.05	15.2	30
0.1	16	20
0.15	20.3	16
0.2	19.9	14

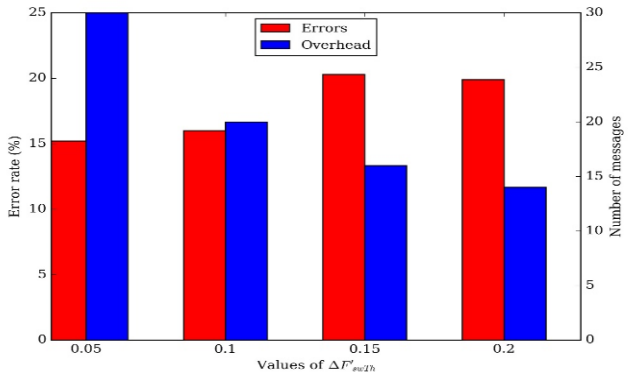

Figure 10. Overhead and Error (%) effect of $\Delta F'_{swTh}$

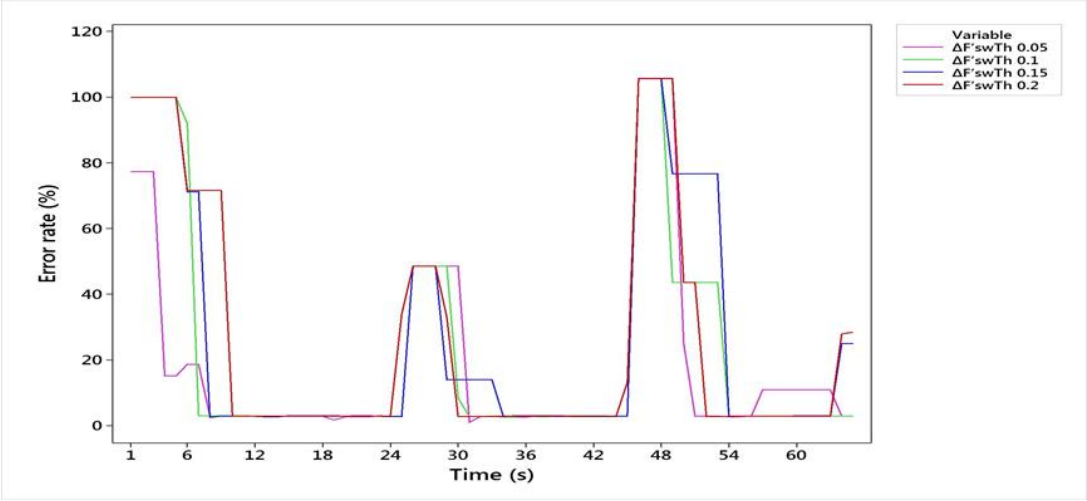

Figure 11. Error effect of $\Delta F'_{swTh}$

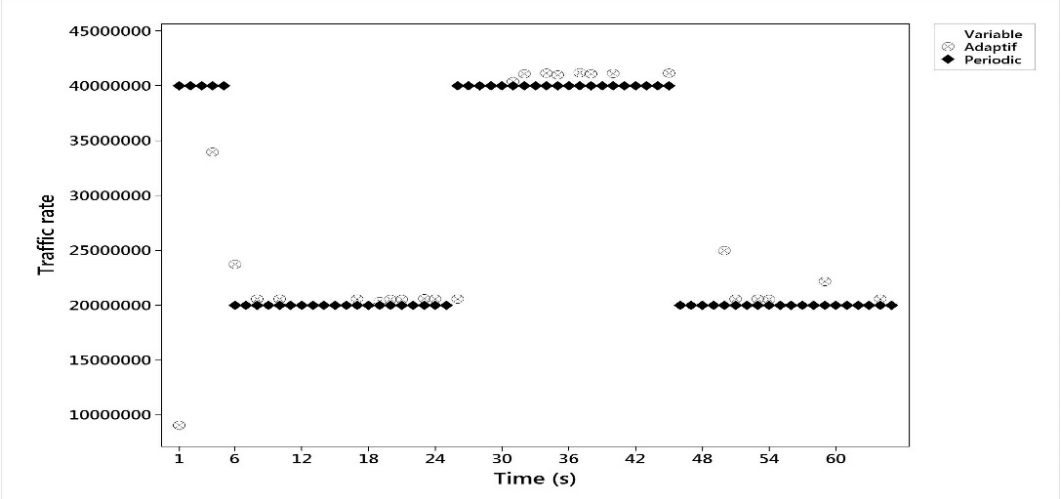

Figure 12. Messaging timing diagram

Figure 12 shows the timing diagram of messages sent by periodic polling approach and adaptive approach. Each dot in the figure represents a sent message. The figure shows that number of messages sent by adaptive approach changes over time and is less than the number of messages sent by periodic polling approach.

Scenario 2

In scenario 2, we aim to show the effect of β from Eq. 3 on the error and overhead. To get the results we created a linear-topology via Mininet in Figure 5 and the timing diagram of the traffic is shown in Figure 6 UDP flows for a total duration of 120s between hosts were generated using Iperf. Based on the results obtained in Scenario 1, $L_{Threshold}$ and $\Delta F'_{swTh}$ was set at 0.01, 0.05 respectively. Value of β is started from 0.125 and increased by 0.1 on each run.

Figure 13 shows the error and overhead effect of the β value of the adaptive approach. Table 6 summarizes the error value (%) for different β values. As it can be seen from Table 6 as the value β increases error increases. The least overhead is observed at value 0.625 and least error is observed at 0.125. Figs. 14-15 shows the change of error over time for different β values. At the beginning, error value is high. That region can be considered as warm up region for adaptive approach to adapt to the traffic. In every 20 sec traffic increases. Each traffic fluctuation causes increased error and adaptive approach quickly catches up with the actual traffic.

Table 5. Effect of β on Error

Value of β	**Error (%)**	**Overhead**
0.125	5.9	47
0.225	7.8	45
0.325	12.2	44
0.425	12.3	42
0.525	7.4	43
0.625	9.4	37
0.725	9.8	46
0.825	15.3	47

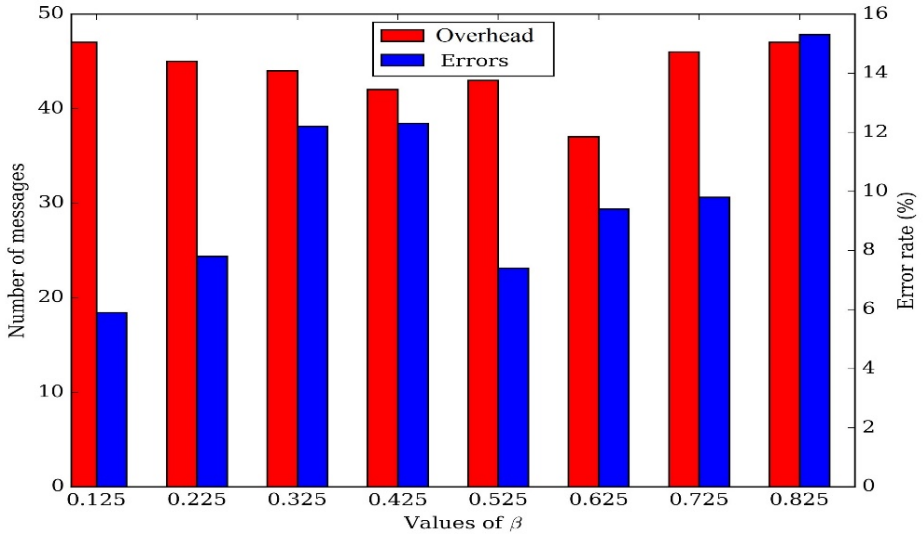

Figure 13. Overhead and Error (%) Effect of β

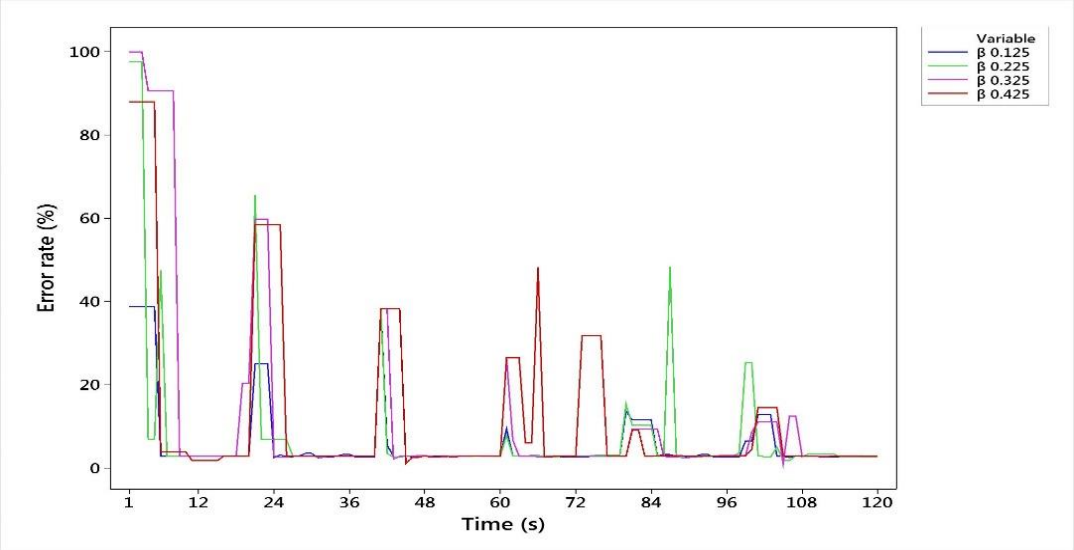

Figure 14. The change of Error over time for different β values

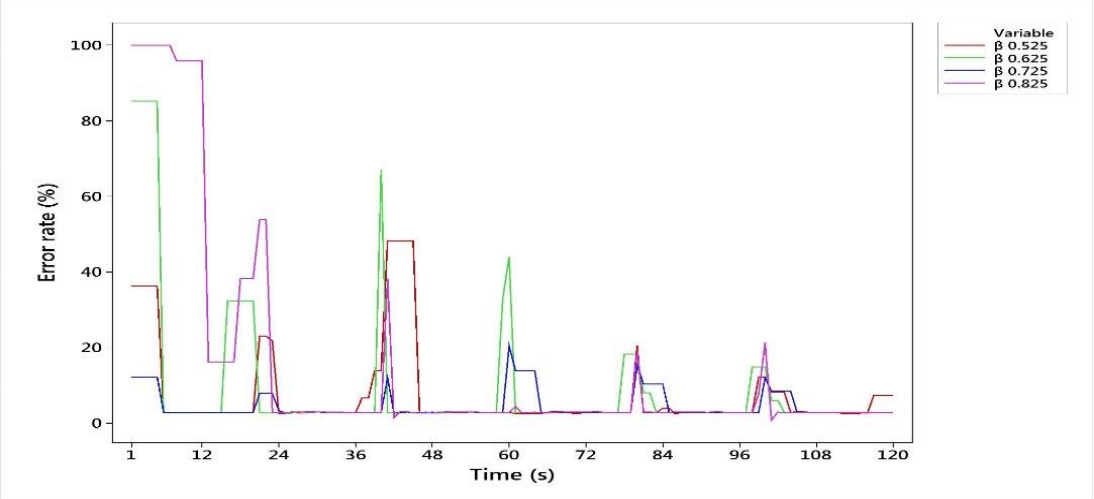

Figure 15. The change of Error over time for different β values

We also analyzed the effect of L_{ratio} on polling message overhead. Figure 16 presents the number of messages sent for different $L_{threshold}$ values. β and $\Delta F'_{swTh}$ was set to 0.125, 0.05 respectively. $L_{threshold}$ started from 10% of link capacity and increased by 10% on each run up to 50%. As $L_{threshold}$ value increases gradually, the overhead on the system is decreasing. Our argument is that close monitoring is not necessary if the link utilization is low, but close monitoring is required when link utilization reaches a critical threshold. The threshold can be adjusted based on the network administrator's priorities

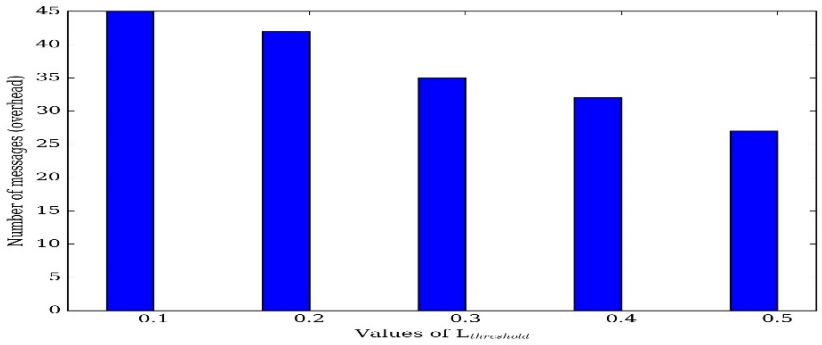

Figure 16. Overhead effect on L_{ratio} values of $L_{threshold}$

Scenario 3

In order to compare the results of the adaptive approaches to other studies, 3 level three topology was created via Mininnet in Figure 3 and traffic timing diagram as shown in Figure 4 which are the same as PayLess [16] study. In this scenario UDP flows for a total duration of 60s between hosts were generated using Iperf. Figure 4 shows also the timing diagram; start, throughput and end time for each stream.

For link utilization, we used the monitoring results of the most heavily used link which is the one between switches s1 and s3 in Figure 4. Results are gathered and compared using two different techniques (periodic and adaptive polling) and also compared to actual traffic. $L_{threshold}$, β and $\Delta F'_{swTh}$ was set to 0.01, 0.125 and 0.05 respectively.

As seen in Figure 17, adaptive and periodic polling techniques closely follow actual traffic. Table 7 summarizes the results of the measurements. When we compare both polling techniques with respect to actual traffic, it is seen that adaptive polling is more accurate than periodic polling. When we compare the overhead among PayLess, periodic probe and the adaptive design, the adaptive approach achieved 6.7% less overhead than PayLess and 43.3% less overhead than periodic polling. With respect to actual traffic, on average adaptive algorithm achieved 5.4% more accurate results compared to periodic polling.

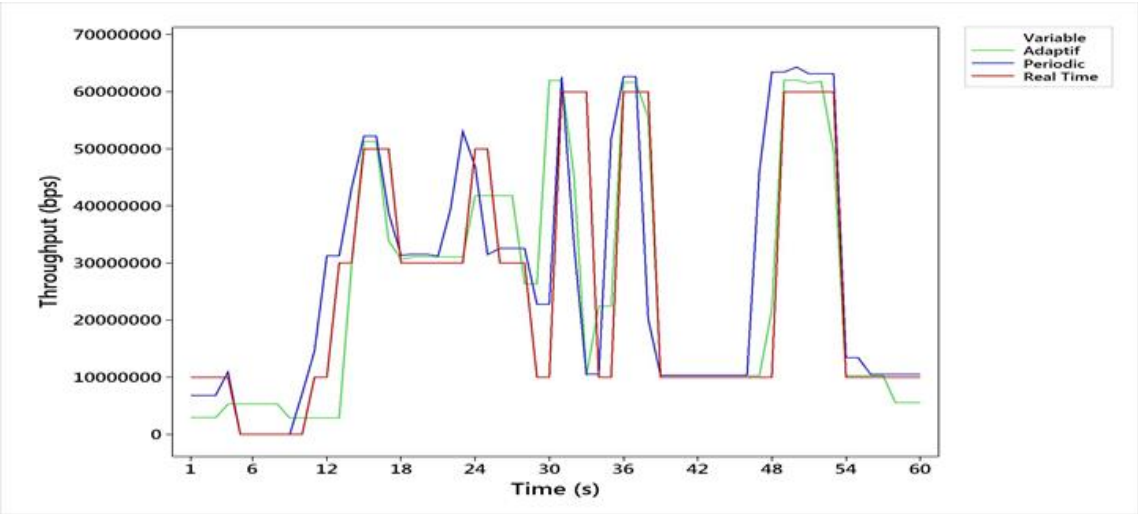

Figure 17. Utilization measurements (UDP)

Table 6. Comparison of Adaptive design, Periodic polling and PayLess measurement

Polling technique	Accuracy (%)	Overhead (%)
Adaptive (UDP)	62.6	43.3
Adaptive (TCP)	43.8	59.5
Periodic	57.2	100
PayLess	-	50

CONCLUSION

In the study, we have introduced an adaptive monitoring approach for SDN. With this approach, we aimed to collect port-based statistics with low overhead and high accuracy. Presented adaptive monitoring approach uses past traffic patterns to predict the next traffic behavior to adjust polling rate and also takes into account the link capacity, utilization rate and fluctuations in traffic to dynamically adjust polling period to increase the efficiency of the monitoring mechanism. By employing these techniques decreased monitoring overhead was achieved while maintaining an acceptable accuracy. All the measurement approaches eventually provide average traffic passed between a given period of time. Our goal in this study is to provide a parametric dynamic adaptive mechanism that can be used under different network conditions. Parameters in the proposed approach can be adjusted accordingly if accuracy is more important than overhead or vice versa.

We created different scenarios to analyze the adaptive approach over the parameters used. At the same time, we made observations on different traffic patterns to make these analyzes more accurate. For F', we showed the trade-off between the current link usage ratio and the network traffic change ratio. Using the β parameter, we observed how these two ratios affect the outcome in network monitoring. For ΔF, we showed that there is a trade-off between the traffic exchange rate and the overhead it brings to the system using the $\Delta F'_{swTh}$ parameter. For L_{ratio}, Using the $L_{threshold}$ value, we can set a particular link usage threshold. Thus, we can provide more stable network management by not making unnecessary queries.

We have evaluated and compared the performance of proposed adaptive monitoring approach with PayLess and periodic polling methods. Results show that the proposed adaptive approach can provide higher statistical collection accuracy than periodic inquiries (within the appropriate interrogation range). Proposed adaptive monitoring approach caused 46% less overhead than the periodic polling method and 6.7% less overhead than PayLess approach. At the same time accuracy of the proposed method is 5.4% better than periodic polling approach.

It is necessary to investigate the dynamic updating of the parametric values used in the continuation of the study. In this way, the relationship between accuracy and additional load can be adjusted automatically according to the system condition. At the same time, it is necessary to examine the gains to be gained when this proposed approach is adapted to the selective tracking approach to monitoring critical keys, not all keys. This proposed approach will be used instead of classical periodic monitoring methods in applications such as traffic engineering and service quality, and more efficient use of the network will be ensured.

REFERENCES

Balasubramanian, V., Aloqaily, M., Reisslein, M. (2021). An SDN architecture for time sensitive industrial IoT. *Computer Networks*, 186; DOI: 10.1016/j.comnet.2020.107739

Castillo, E.F., Rendon, O.M.C., Ordonez, A., Granville, L.Z. (2020). IPro: An approach for intelligent SDN monitoring *Computer Networks,* 170; DOI: 10.1016/j.comnet.2020.107108

Chowdhury, S.R., Bari, M.F., Ahmed, R., Boutaba, R. (2014). Payless: A low cost network monitoring framework for software defined networks. *Network Operations and Management Symposium (NOMS)*, IEEE, 1–9.

Floodlight Controller (2017). http://www.projectfloodlight.org. (Accessed Date: March 12, 2017)

Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., Shenker, S. (2008). NOX: towards an operating system for networks. *SIGCOMM Computer Communication Review,* 38(3):105-110.

Hernandez, E.A., Chidester, M.C., George, A.D. (2001). Adaptive sampling for network management. *Journal of Network and Systems Management*, 9(4): 409–434.

Huang, L., Zhi, X., Gao, Q., Kausar, S., Zheng, S. (2016). Design and implementation of multicast routing system over SDN and sFlow. *8th IEEE International Conference on Communication Software and Networks*, 524-529.

José, S.V., Pere, B.R. (2017). Reinventing NetFlow for OpenFlow Software-Defined Networks. In: *arXiv preprint arXiv:1702.06803*.

Mohan, V., Reddy, YJ., Kalpana, K. (2011). Active and passive network measurements: a survey. *International Journal of Computer Science and Information Technologies*, 2(4): 1372-1385.

Open Networking Foundation. OpenFlow Switch Specification Version 1.5.1. 2015. https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf (Accessed Date: March 04, 2019).

Özer, H., Okumuş, İ.T. (2019). Yazılım tanımlı ağlarda izleme. *Kahramanmaraş Sütçü İmam Üniversitesi Mühendislik Bilimleri Dergisi*, 22: 26-33.

Phan, XT., Martinez-Casanueva, ID., Fukuda, K. (2017). Adaptive and distributed monitoring mechanism in software-defined networks. *13th International Conference on Network and Service Management (CNSM)*,1–5.

POX Controller (2017). https://github.com/noxrepo/pox (Accessed Date: July 12, 2017)

Shah, S. A. R., Bae, S., Jaikar, A., Noh, S.-Y. (2016). An adaptive load monitoring solution for logically centralized sdn controller. *18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 1–6.

Shirali-Shahreza, S., Ganjali, Y. (2013). Empowering Software Defined Network controller with packet-level information. *2013 IEEE International Conference on Communications Workshops (ICC)*, 1335-1339.

Yang, L., Ng, B., Seah, W.K., Groves, L., Singh, D. (2021). A survey on network forwarding in Software-Defined Networking. *Journal of Network and Computer Applications*, 176; DOI: 10.1016/j.jnca.2020.102947

Li, M., Chen, C., Hua, C., Guan, X. (2019). CFlow: A learning-based compressive flow statistics collection scheme for SDNs. *IEEE International Conference on Communications (ICC)*, 1–6

Liu, C., Malboubi, A., Chuah, C.-N. (2016). OpenMeasure: Adaptive flow measurement & inference with online learning in SDN. *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 47–52.

Su, Z., Wang, T., Xia, Y., Hamdi, M. (2014). FlowCover: low-cost flow monitoring scheme in software defined networks*. IEEE GLOBECOM'14,* 1956-1961.

Su, Z., Wang, T., Xia, Y., Hamdi, M. (2015). Cemon: a cost-effective flow monitoring system in software defined networks. *Computer Networks,* 92:101-115.

Terzi, D., Terzi, R., Sagiroglu, S. (2017). Big data analytics for network anomaly detection from NetFlow data. *(UBMK'17) 2nd International Conference on Computer Science and Engineering*, 592-597.

The Mininet Platform (2018). http://mininet.org (Accessed Date: December 12, 2018)

Tootoonchian, A., Ghobadi, M., Ganjali, Y. (2010). OpenTM: Traffic matrix estimator for OpenFlow networks. *Proceedings of the 11th International Conference on Passive and Active Measurement,* 201–210.

Van Adrichem, N.L., Doerr, C., Kuipers, F.A. (2014). OpenNetMon: Network monitoring in openflow software-defined networks*. IEEE Network Operations and Management Symposium (NOMS),* DOI: 10.1109/NOMS.2014.6838228

Yu, C., Lumezanu, C., Zhang, Y., Singh, V., Jiang, G., Madhyastha, HV. (2013). FlowSense: Monitoring network utilization with zero measurement cost, *Passive and active measurement (PAM)*, 7799: 31–41.

Yu, M., Jose, L., Miao, R. (2013). Software defined traffic measurement with OpenSketch. *The 10th USENIX Symposium on Networked Systems Design and Implementation, NSDI'13*, 29–42.