FloCon: Flow Conservation Framework for Monitoring SDN Networks

Jesús Antonio Puente Fernández, Luis Javier García Villalba

***Abstract*—** Software Defined Networks (SDN) is a new concept of network architecture. It intends for being more flexible and simplifies the management in networks respect the existing ones. All these aspects are possible because the separation of control plane (controller) and data plane (switches). OpenFlow is the most important protocol for SDN networks that provides the communication between control and data plane. Due the existence of an interface, it is possible to deploy monitoring tools to obtain the network status and retrieve a statistics collection. Therefore, achieve the most accurate statistics depend on the strategy to monitor the network. In this paper we describe an optimization for the traffic monitoring in SDN networks. Moreover, it decreases the number of monitoring queries to improve the network traffic and also it reduces the overload. The experiments shown that the results of monitoring with and without the optimization demonstrate the feasibility of our proposal. Finally, the conclusions and future challenges are presented.

***Keywords*—**About four key words or phrases in alphabetical order, separated by commas.

I. Introduction

COMMUNICATION technologies has experimented a large evolution from the 80´s till became at the present as Software Defined Networks (SDN). Nevertheless, there are some concepts that shape the base of this technology development. On one hand, these advances are: central control, active networks and networks virtualization. On the other hand, the great progress is in the control-data plane separation. Central control goes back at the beginning of the 80´s and is focused in the information transport over the same channel. This technology offered several advantages in terms of simplicity, however it was quite fragile, insecure and vulnerable. Active networks [1] [2] appeared during the 90´s. This kind of networks allows performing custom assignments in the data packets that travel around the switchers. The most popular example of active networks is Middleboxes, which main tasks are providing firewall, proxy functions and application services. Finally, networks virtualization [3,4] plays an important role with this way of management. Network virtualization has similarities to computer virtualization, where different devices can use and share hardware resources to different Operating Systems OS in a host. The aim of network virtualization is to isolate multiple logical networks, each of them with completely different addressing and forwarding mechanism, but sharing the same physical infrastructure. Clear instances of network virtualization are Virtual Local Area Networks (VLAN).

SDN is a new network architecture model that gathers the advances mentioned previously. It separates control plane and data plane in network devices to enable a programmable behaviour and removes the rigidity of the static protocols. Also, it proposes a central control by a high-level software application that allows improving network management in a fast way because this centralized control. Therefore, network administrator can have a centralized and programmable control of the traffic behaviour inside the network without requiring an individual configuration of hardware devices. SDN is used in different scopes as virtualization, home networking, security, data centres, Internet of things [5] among others.

Apart from this, OpenFlow [6,7] is the most used protocol in SDN networks. The Open Networking Foundation ONF [8] is the organization that is responsible for the OpenFlow specification maintenance. This protocol is used in different areas like home networking, data centers, and mobile mobility among others. In addition, we describe a short introduction of the OpenFlow protocol in this paper.

Network capacity depends on the well-done work of the controller among other components like switches, links, etc.. This element has the ability to detect or advance errors or misuse inside the network. These errors can be resolved with the measurement of some network metrics like data rate, lost packets or delay. The monitoring of this information is a task that increases the overload of the network via hardware or software. In this way, SDN and OpenFlow open new perspectives in monitoring field and some challenges [9] like the workload in both planes. On one hand, if the controller requests information to every switch continuously, it provides accurate data and could management the network in a better way. On the other hand, this process can generate an overload on the resources of the controller processing and also can affect negatively in the most demanding tasks.

In this paper, we present a new monitoring framework that intends to decrease the number of monitoring queries in the switches of the topology without compromising the accuracy of the monitored values. Moreover, this architecture provides a “plug & play” module design to add new algorithms or

Jesús Antonio Puente Fernández, Luis Javier García Villalba are the Group of Analysis, Security and Systems (GASS), Department of Software Engineering and Artificial Intelligence (DISIA), Faculty of Information Technology and Computer Science, Office 431, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases, 9, Ciudad Universitaria, 28040.

strategies to monitor the network. In addition, this work describes active monitoring methods obtaining metrics as data rate, loss rate and delay. For this, we try to reduce the request of the switches that do not offer valuable information. Remaining values of the non-monitored switches will be calculated with the information of neighbour switches. The experiments show the effectiveness of our algorithm in the reduction of monitoring queries and the accurate values respect the difference of the use/non-use of the reduction. Finally, we monitoring a video streaming delivery through a SDN network to demonstrate the feasibility of the framework.

The rest of the paper is outlined as follows: section II defines the OpenFlow architecture. Section III are described the related works of monitoring tools. Then, Section IV explains the monitoring framework. Section V contains all simulations and results. Finally, Section IV concludes with a short discussion and conclusions.

II. OpenFlow Architecture

The OpenFlow architecture [6] is based on three main entities as shown in Figure 1: an OpenFlow switch (data plane), an external controller (control plane) and the OpenFlow Protocol [7] that is the responsible for the communication between controller and switches through a secure channel.

In data plane, one or more flow tables and a group table are used for the packet lookup and data forwarding within an OpenFlow switch. Each flow table is based on a group of match fields, counters and instructions. Match fields manage if an incoming packet must be process by the switch or must be send it to the Controller in a process called matching. In the counters are set up the number of packets that match. Finally, the instructions field determines the actions to use to when a matching occurs.

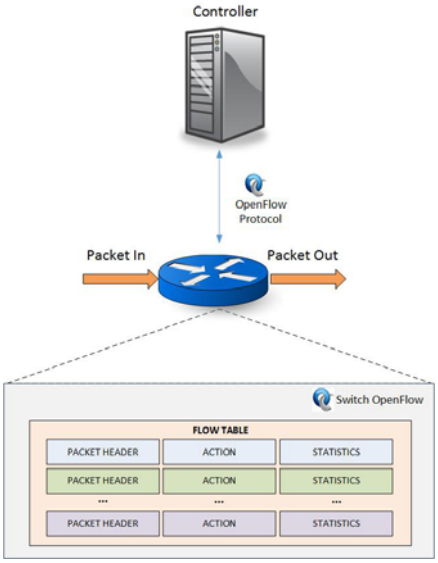

Fig. 1 OpenFlow architecture

Matching process is divided in two steps. First of them consists on checking the packet header with the match table of a single flow table. The second step depends on this checking. If these two values (packet header and match table) are similar, the corresponding actions of the instructions field are taken. In other case, the OpenFlow Protocol specifies the required and optional actions that a single switch can take. Three possible instructions compose the required actions: send the packet by a given port, send the packet to the controller or drop the packet. Even these actions are obligatory, in case of the incoming packet does not match with a match field, the switch can be configured to send it to the controller or drop it.

Some authors [9][10][11] have classified the abstractions of network resources like southbound and northbound interfaces (Figure 2). Southbound interfaces have the function to abstract the functionality of the programmable switches and connect with the software that is executing in the controller. The most representative example of southbound interface is OpenFlow. Over these kinds of interfaces is running a Network Operating System (NOS) that describes all software tools that allow creating applications and controlling the network behaviour. Examples of NOS are NOX/POX [12], Maestro [13], Beacon [14] and Floodlight [15]. Northbound interfaces allow creating applications or high-level network policies and sending it to the NOS. Examples of northbound interfaces is Frenetic [16][17], Procera [18], Netcore [19] and McNettle [20].

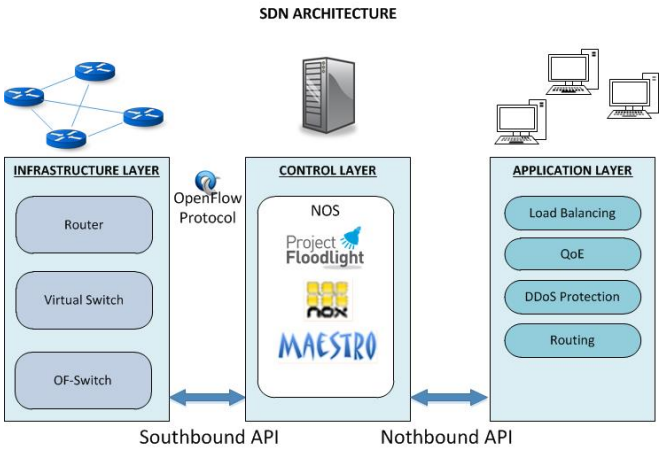

Fig. 2 Northbound and Southbound interfaces

III. Related Works

Monitoring tools play an important role in the network such as an unexpected traffic increase, a Distributed Denial of Service (DDoS), an illegitimate traffic flow among others. Nowadays, several network-monitoring tools are proposed to measure network statistics. It is possible to classify (in different sets) all these tools based on the operations that they use. Of course, each method has its advantages and its disadvantages in terms of performance. In this way, if the process is adding traffic into the network it will nominate as an active method, otherwise as passive method.

On one hand, passive methods do not modify the network performance. The main characteristic of this method is the existence of traffic observers placed in the network devices.

These observers are called as agents and its most important task is sending the information of the network through a monitoring protocol. On the other hand, active methods send packets into the network at the same time as the normal traffic. These packets are called as probe packets and its function is measure statistics as round-trip time, delay among others. This solution impacts directly in the network performance due to the traffic increment.

In typical network architectures exist some protocols to monitor the devices that build the network. Examples of these protocols that use passive method are SNMP [21] and NETCONF [22]. Likewise, it exists flow based monitoring tools as jFlow [23], NetFlow [24] and sFlow [25], which estimate either complete or sampled traffic statistics and send it to a control agent.

In SDN networks, the principal proposals related with network monitoring are SuVMF [26] that offers a novel architecture for SDN large-scale networks based on a Software-Defined Unified Virtual Monitoring Function. This idea also uses a passive method to monitor and consists of three important entities that are responsible for monitoring management, intelligent control and modules that filter and transform the data. The statistics collection utilizes several passive methods as sFlow, SNMP among others. The idea proposed in [27] uses a passive method to measure the network performance. It uses beacons to send probe packets and install additional flows in switches. Then, these beacons are used to estimate the packet lost rate and delay. A hybrid solution between passive and active methods is the framework besought in [28]. This work proposes a network-monitoring framework based on an orchestrator module with a flexible method to retrieve network statistics. Moreover, it creates a user profile based on its needs, retrieving statistics with passive (data rate and error rate) and active (probe packets to the switches) methods.

Works like OpenNetMon [29] monitors per-flow metrics in OpenFlow networks. It is specialized in throughput, delay and packet loss to determinate if the end-to-end Quality of Service parameters are met to find suitable paths. It looks for the minimum number of queries to switches in order to obtain the metrics. The time interval of these polls depends on the increase-decrease of its past values. Apply different strategies to query for statistics can help to reduce the overhead in switches as well as in networks. In this order, OpenTm [30] proposes to follow a non-uniform querying distribution respect uniform schemes. It shows that this strategy is highly faster than the existing ways of traffic estimation in IP networks. Another strategy to have into account is to duplicate the traffic and send it to a monitoring agent. This idea is used in MonSamp [31] where the authors create two agents named as collector and analyser to read continuously the flows in the switches. The algorithm will increase or decrease the flow rules in the switches based on the capacity of the monitor and the network links congestion. The properties of these monitoring strategies can be used in parallel with other modules to get better results in terms of QoS among others. In [32], a framework for optimized multimedia routing in combination with a monitoring module is presented to provide QoS in different multimedia services. Its best advantage is the easy way to add routing algorithms to get the best path in the data sending.

IV. FLOCON: FLOW CONSERVATION FRAMEWORK

Large-scale enterprises, network operators and service providers needs the best used of traffic and resource monitoring to obtain their best support. Because of this, monitoring research is a network field in which there is a big investment in order to obtain the best solution in each case. In addition, we must take into account the possibility to add new monitoring hardware and its overrun. In other words, the best components and use of them you have the best accurate results you will obtain.

To achieve an optimized monitoring has several advantages in terms of CPU and memory overload among others. Hence, if we complement the idea of monitoring plus SDN architecture, we will obtain a powerful tool to gather accurate and instant data from networks. Physically, the implementation does not depend on a specific set of hardware or components of a specific vendor but the such equipment has to be compatible with a specific southbound API as OpenFlow. Once the compatibility has been achieved, the controller is able to monitor the whole network.

Nevertheless, several risks and problems have to be considered using a SDN controller for network monitoring. The most problematic statement is to minimize the CPU load for not interfering in the rest of controller´s processes since the periodic monitoring of all switches of the topology impacts directly in the performance and results. Moreover, it has to take into account the applicability of the network since certain networks demands throughput, others speed, etc. Finally, retrieved results must be present in a clear way and the most up-to-date possible.

Taking into account the above, the aim of our idea is search for the best and the most accurate results at the same time that we optimize network computability. We propose FloCon, an optimized monitoring framework based on the flow conservation algorithm as is illustrated in Figure 3. This framework is based on different modules following the SDN architecture as is described in section 2. It consists on the Infrastructure Layer composed of switches, routers and other elements that are responsible for the forwarding functions). The Control Layer takes the decisions of the network behaviour and sends the taken instructions to the infrastructure layer through the Southbound API (e.g. OpenFlow protocol). In the same way, the Control Layer provides functionality to the Application Layer thanks to a Northbound API (Rest API) with high level policies.

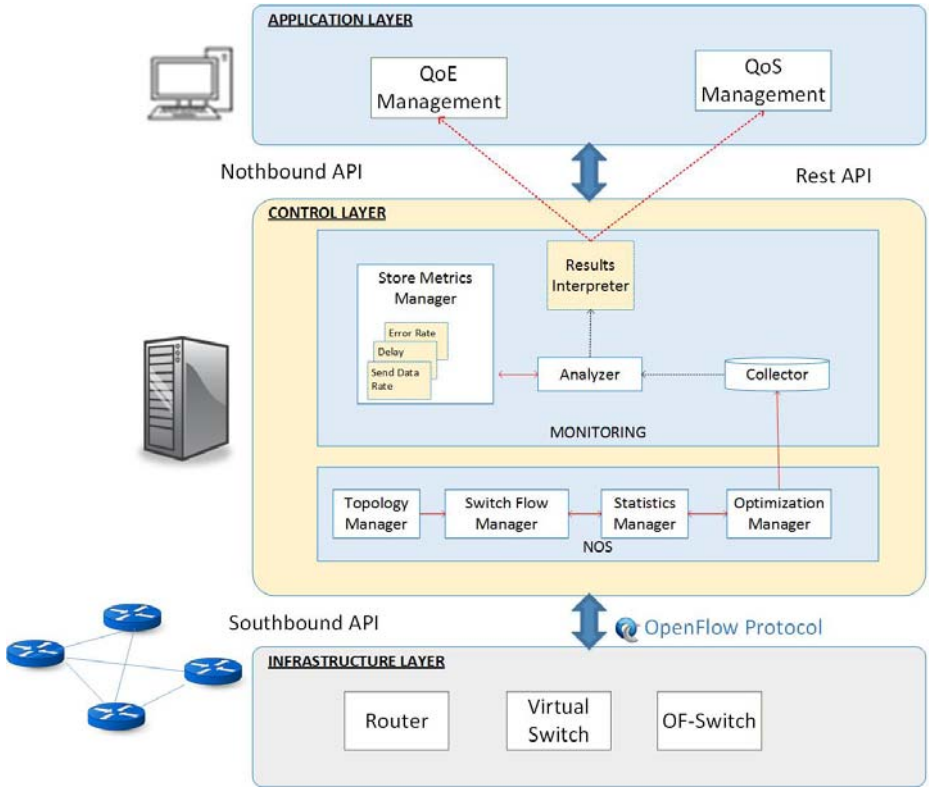

Fig. 3 Flow Conservation Framework

The following list describes the framework modules (control layer) in a deeper way:

- Topology Manager. This module identifies the network devices, network links and their capabilities through LLDP (Link Layer Discovery Protocol) which sends OpenFlow discovery messages across the network. The retrieved topology is organized as is described in section 4.A Network Notation. The information of the topology is sent to the rest of the modules of the control layer.
- Switch Flow Manager. This module receives the information provided by the Topology Manager module in order to set the flow tables in case of a packet matching. Moreover, this module provides the information of all switches ports that the topology is composed.
- Statistics Manager. This module receives the optimized topology from the Optimization Manager and send OpenFlow query messages to gather the statistical information from the network switches. Once this operation has finished, the statistics information is sent to the Collector module.
- Optimization Manager: This module is responsible for applying the selected algorithm from the list of available algorithms (Store Metrics container). Its main functionality is to build an optimized topology from the original one liberating switches based on determined features to reduce the number of statistics queries.
- Collector. This module clears and filters all information coming from the Statistics Manager and stores the selected information previously set in the configuration file. The "cleaned" information is sent to the Analyzer module to perform data processing.
- Analyzer. Data processing is performed in this module with the information received from the Collector to provide high level data. This high level data is used to detect and check both individual (swtich) and global view of the network. In this way, this module is able to provide a prediction or a trend based on time studying previously values of the network behavior.
- Store Metrics Manager. This module consists on a container for storing monitoring algorithms such as the flow conservation algorithm explained in Algorithm 1. The aim for this module is the effortless insertion, modification and updating including own or third parties modules or algorithm implementations.
- Results Interpreter. Once the Analyzer module has finished with its processing, this module provides functions to detect anomalies such as ACL-violations, IDS/Firewall alerts, DDoS attack, intrusion detection among others. Therefore, it is able to regularize the traffic (increase/decrease) if one of the previous anomalies takes places in the network.

A. Network Notation

Our network abstract model is represented as a directed graph called G=(S, L) where $S=\{s_1,s_2,\ldots,s_n\}$ denotes the set of switches and $L = \{l_1,l_2,\ldots,l_n\}$ represents the set of links between switches. We let ns = |S| and nl = |L| denote de number of switches and links in the topology respectively.

Also, we define degree of a switch (number of incident links) like deg(s_i) where $s_i \in S$. We assume that ns and nl are finite and deg(s_i) > 0. All this information is gathered in Table I.

TABLE I
NETWORK NOTATION MODEL

Symbol	Description
G=(S,L)	Network graph
S	Set of switches in the network Graph G
L	Set of links in the network Graph G
$s_1,s_2,\ldots,s_n$	Generic switches in the network Graph G
$l_1,l_2,\ldots l_n$	Generic links in the network Graph G
ns = |S|	Number of switches in G
nl = |L|	Number of links in G
deg (s_i)	Number of links in switch s_i

Also, we assume in our model that the monitored data is grouped in packet flows F. Thus, all data packets that are sent in G belongs at least, to one packet flow Fi ϵ F. Each Fi has a path from its source switch Si to its destination switch Sj.

B. Flow-Conservation Algorithm Optimization

Majority of monitoring strategies request the state of all ports in network devices. This method to obtain network information is very accurate but at the same time too inefficient. Therefore, to reduce monitoring queries in resources, we propose not to monitor switches that contain only two ports inside them as is explained in Algorithm 1. In each monitoring period, OPTIMIZE_FUNCTION goes through all switches that compound the topology and checks how many neighbour switches are connected with him: if it has only two adjacent switches (deg(si) == 2 {incoming port, outgoing port}), it does not request for statistics, otherwise it demands the switch status with CALCULATE_STATISTICS.

Algorithm 1: Optimize topology function

Input: network graph *G(S,L)*

Result: optimized statistics from graph *G(S,L)*

```
1.  procedure OPTIMIZE_FUNCTION
2.      for each switch s_i ϵ S do
3.          // Checks if the number of incoming and outgoing
        links is 2
4.          if deg(s_i)==2 && not(contains_host(s_i))
5.              CALCULATE_OPTIMIZE_STATISTICS (s_i)
6.          else CALCULATE_STATISTICS(s_i)
7.          end if
8.      end for
9.  end procedure
```

The calculation of optimized switches depends on the links of its adjacent switches. Algorithm 2 describes the procedure to calculate it. It sets the value of the data rate in the port j of the adjacent switch sk that is connected with him (switch si). Consider that G is directed, all incoming packets flows Fi of switch si (arrive in incoming port I) will be forward to the other port (outgoing port O) as Figure 3(a) shows. This affirmation satisfies the flow-conservation law that describes that the sum of the traffic flow going into a switch Si is approximately the same as the sum of the traffic leaving si in a 2-grade node. Formally, the flow-conservation low is set as Equation 1.

$$\sum_{j=1}^{|F|} F_j(I_{Si}) - \sum_{j=1}^{|F|} F_j(O_{Si}) \approx 0 \qquad (1)$$

For this reason, we omit the monitoring request of this switch and then reduce the total number of queries in each monitoring period. These requests are sent from the OpenFlow controller through the secure channel in a message called OFPT_STATS_REQUEST. Once the switch receives this message, it replies its state to the controller with another message named OFPT_STATS_REPLY.

Algorithm 2: Calculate Optimized Statistics function

Input: switch S_i

Result: send data rate d_{sij} for switch s_i and port j

```
1.  procedure CALCULATE_OPTIMIZE_STATISTICS
2.      for each port s_j ϵ s_i do
3.          // Read the sent bytes of the adjacent port j of the
        adjacent switch k in period time t with
4.          s_ij^t = bytes in OFPT_STATS (s_k, port (j))
5.          if (t > t_init) then
6.              d_sij = (s_ij^t - s_ij^(t-1))/t_mon;
7.          end if
8.      end for
9.  end procedure
```

In Algorithm 3, CALCULATE_STATISTICS sends request to every port in each switch and process the response to obtain the data rate. After that, it calculates the data rate with the bytes from the source port j of the source switch s (sijt) respect the same port and same switch of the previous monitoring period (sijt-1). Each port has its statistics to measure, so the data rate in each one is different as Figure 3(b) describes.

V. SIMULATIONS AND RESULTS

In this section, we present all results from the simulations of our effective monitoring. The simulations are tested using the topology described in Figure 4 which is composed of 9 OF-Switches (s1, s2, s3, s4, s5, s6, s7, s8 and s9) and 2 host (h1 and h2) connected to s1 and s9 respectively. In order to check the optimization of this work, there are some switches connected with another single switch such as s4, s6 and s8.

Testing set was executed in a network simulator called Mininet v 2.1.0 [33] using a python script, which contains the topology to examine. The advantage of this tool is that enables the creation of custom topologies in a computer in few steps. The simulation was executed in a virtual machine within a laptop that is composed by an Intel core i5 2.4 GHz, 8GB DDR3 RAM, OS X 10.10. Inside the VM, there is a Linux (Ubuntu v. 13.04) guest Operative System (64 bits, 4 Gb RAM, 2 CPUs, 8 GB HD).

A simulation consists in a delivery of a video from host 1 to host 2 through the network using VLC video server and RTP/UDP as streaming protocol as the same time that the

monitoring module is running. The task of the monitor is to detect the traffic increasing through the links of the path where the information is sent. The features of the video (called "Highway_cif" [34]) used in the streaming are joined as follows; format: MPEG 12, frames: 2000, duration: 80 seconds, size: 2.97 MB while the monitoring time is 200ms. The connection between the client host and server host is a fixed path composed of [{switch: s1 - port: p2}, {switch: s2 - port: p2}, {switch: s3 - port: p2}, {switch: s4 - port: p2}, {switch: s9 - port: p1}]. The test is repeated twice in the same conditions changing the monitoring method in each one. In the first one, the optimization proposed in this paper is used while in the other, it applies the method proposed in [28] which monitories all switches. Finally, the results are compared with these two techniques to determinate which method is the best.

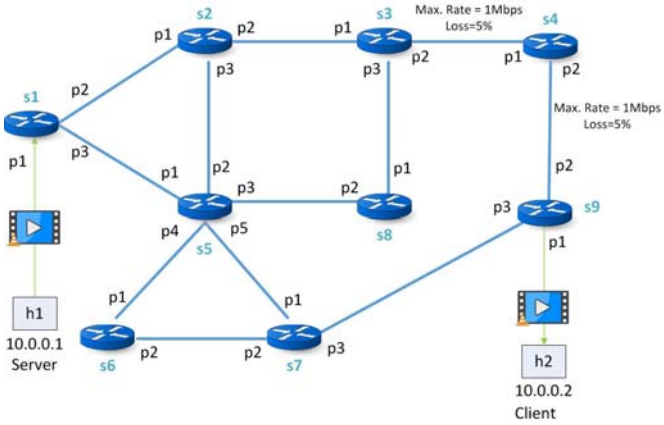

Fig. 4 Topology tested

The results from these tests are divided into two metrics, from one side, the number of request to get the status of each switch and on the other side, the data rate of the switches that compose the streaming path. The number of monitoring queries in the non-optimized simulation was 5115 while in the optimized was 3454. The difference is 1661 queries less, so there is a reduction about 33% of request respect a non-optimized monitoring simulation.

Figure 5(a) and Figure 5(b) shows the traffic flow between two simulations in which one applies the optimization respect another that does not use it. It describes the data flow (in bps) through the links s3-s4 ({switch: s3 - port: p2}, {switch: s4 - port: p1}) in Figure 5(a) and s4-s9 ({switch: s4 - port: p2}, {switch: s9 - port: p2}) in in Figure 5(b). As expected, these links experiment an increase of data flow due to the transmission of the video between h1 and h2.

As soon as the transmission finishes (around 210 requests), the only data traffic is because of messages between the switches and the controller. As Figure 5(a) shows, the difference between the links S3-S4 with/without the optimization is 30.4833 Bytes (30,48 KB) and 30.6622 bytes (30,66 KB) in the link S4-S9 in Figure 5(b), so the difference is insignificant.

VI. Conclusion and Future Works

This paper presents an effective SDN monitoring module that reduces the number of monitoring request using the OpenFlow protocol. The results of the simulations confirm that the number of monitoring queries applying the algorithm proposed in this work is less than the monitoring of every switch in the topology.

On one hand, there are some advantages when this optimization is in use. For example, in a linear topology composed by n switches connected two by two, the number of monitoring queries will be minimum. On the other hand, there are some limitations of this optimization. In a custom topology without 2-grade switches, this algorithm could not be applied, so the result will be the same.

The future challenges comprise new strategies to reduce the monitoring queries even more that the idea proposed in this paper. Moreover, including other metrics as error rate and delay of packets. Furthermore, optimizations for measuring delay can be develop to measure the retard in the network devices as well as the error rate performed by the packet lost.

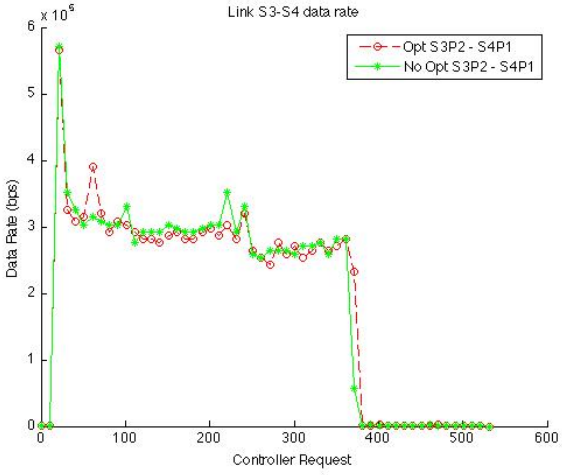

Fig. 5.a Optimized vs No Optimized S3-S4 data rate

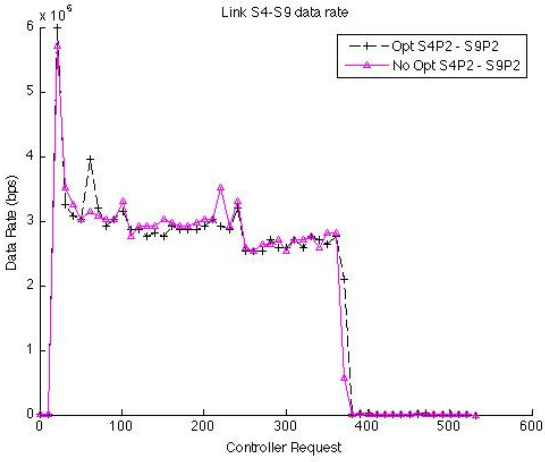

Fig. 5.b Optimized vs No Optimized S4-S9 data rate

REFERENCES

[1] B. Schwartz, A. W. Jackson, W. T. Strayer, W. Zhou, R. D. Rockwell, and C. Partridge, "Smart packets: Applying Active Networks to Network Management," ACM Transactions on Computer Systems, vol. 18, pp. 67, February 2000.

[2] B. Schwartz, A. W. Jackson, W. T. Strayer, W. Zhou, R. D. Rockwell, and C. Partridge, "Smart Packets for Active Networks," 1999 IEEE Second Conference on Open Architectures and Network Programming, March 1998.

[3] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, G. Parulkar. FlowVisor: A Network Virtualization Layer. October 2009.

[4] Barona López, L. I., Valdivieso Caraguay, A. L., García Villalba, L. J., & López, D. (2015). Trends on virtualisation with software defined networking and network function virtualisation. Networks, IET, 4(5), 255-263.

[5] Valdivieso Caraguay, A. L., Benito Peral, A., Barona Lopez, L. I., & García Villalba, L. J. (2014). SDN: Evolution and Opportunities in the Development IoT Applications. International Journal of Distributed Sensor Networks, 2014.

[6] N.McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communications Review. March 2008.

[7] Openflow Switch Specification v1.1.0. February 2011.

[8] Open Networking Foundation. [Online]. Available: https://www.opennetworking.org/

[9] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for Network Update," ACM SIGCOMM Computer Communication Review, vol. 42, pp. 323–334, October 2012.

[10] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are We Ready for SDN? Implementation Challenges for Software-Defined Networks," IEEE Communications Magazine, vol. 51, pp. 36–43, July 2013.

[11] H. Kim and N. Feamster, "Improving Network Management with Software Defined Networking," IEEE Communications Magazine, vol. 51, pp. 114–119, February 2013.

[12] N.Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, S. Shenker. NOX: Towards an Operating System for Networks. ACM SIGCOMM Computer Communicatios Review. vol. 38, no 3, p. 105-110 July 2008.

[13] Z Cai, A. Cox T. Eugene. Maestro: A System for Scalable Openflow Control. March 2008.

[14] Beacon: a Java-based OpenFlow Control Platform. [Online]. Available: http://www.beaconcontroller.net/

[15] Project Floodlight: Open Source Software for Building Software-Defined Networks. [Online]. Available: http://www.projectfloodlight.org/

[16] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A Network Programming Language," in Proceedings of the The 16th ACM SIGPLAN International Conference on Functional Programming, (New York, NY, USA), pp. 279–291, ACM, September 2011

[17] N. Foster, A. Guha, M. Reitblatt, A. Story, M. Freedman, N. Katta, C. Monsanto, J. Reich, J. Rexford, C. Schlesinger, D. Walker, and R. Harrison, "Languages for Software Defined Networks," IEEE Communications Magazine, vol. 51, pp. 128–134, February 2013.

[18] A. Voellmy, H. Kim, and N. Feamster, "Procera: A Language for High-Level Reactive Network Control," in Proceedings of the First Workshop on Hot topics in Software Defined Networks, (New York, NY, USA), pp. 43–48, ACM, August 2012.

[19] C. Monsanto, N. Foster, R. Harrison, and D. Walker, "A Complier and Run-time System for Network Programming Languages," ACM SIGPLAN NOTICES, vol. 47, pp. 217–230, January 2012.

[20] A. Voellmy and J. Wang, "Scalable software defined network controllers," ACM SIGCOMM Computer Communication Review, vol. 42, no. 4, pp. 289–290, August 2012.

[21] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "Simple Network Management Protocol (SNMP)." RFC 1157 (Historic), May 1990.

[22] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)." RFC 6241 (Historic), June 2011.

[23] A. C. Myers, "JFlow: Practical Mostly-static Information Flow Control," in Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '99, (New York, NY, USA), pp. 228–241, ACM, 1999.

[24] B. Claise, "RFC 3954 - Cisco Systems NetFlow Services Export Version 9." RFC 3954, October 2004.

[25] P. Phaal and M. Lavine, "Sflow version 5," July 2004.

[26] T. Choi, S. Kang, S. Yoon, S. Yang, S. Song, and H. Park, "SuVMF: software-defined unified virtual monitoring function for SDN-based large-scale networks". In Proceedings of The Ninth International Conference on Future Internet Technologies (p. 4). ACM. June 2014.

[27] M. Shibuya, A. Tachibana, and T. Hasegawa, "Efficient Performance Diagnosis in OpenFlow Networks Based on Active Measurements," in The Thirteenth International Conference on Networks ICN 2014, pp. 268–273, February 2014.

[28] Á. L. Valdivieso Caraguay, J. A. Puente Fernández, and L. J. García Villalba, "An Optimization Framework for Monitoring of SDN/OpenFlow Networks", International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC), 2015.

[29] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network Monitoring in OpenFlow Software-Defined Networks," in Network Operations and Management Symposium (NOMS), pp. 1–8, IEEE, May 2014.

[30] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "OpenTM: Traffic Matrix Estimator for OpenFlow Networks," in Passive and Active Measurement, pp. 201–210, Springer, January 2010.

[31] D. Raumer, L. Schwaighofer, and G. Carle, "MonSamp: A distributed SDN application for QoS monitoring," in Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 961–968, IEEE, September 2014.

[32] Á. L. Valdivieso Caraguay, J. A. Puente Fernández, and L. J. García Villalba,"Framework for Optimized Multimedia Routing over Software Defined Networks". Computer Networks (in press), 2015.

[33] Mininet. (https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#limits).

[34] Highway.http://www2.tkn.tu-berlin.de/research/evalvid/cif/highway_cif.264.