FlowVista: Low-bandwidth SDN monitoring driven by business application interaction

Bartlomiej Siniarski, John Murphy and Declan Delaney
UCD, School of Computer Science, Belfield, Dublin 4, Ireland
bartlomiej.siniarski@ucdconnect.ie, john.murphy@ucd.ie, declan.delaney@ucd.ie

***Abstract*—Large enterprises are moving towards Software Defined Network (SDN) adoption, where business-critical applications are being deployed on top of highly programmable network components and orchestrated by a single or multiple controllers using network protocols, such as OpenFlow or NetConf for communication. Just like in traditional networks, low-cost monitoring solutions need to be developed for SDN. The issue of inaccurate network traffic tagging and amount of overhead associated with using traditional monitoring techniques based on packet sampling or Deep Packet Inspection (DPI) need to be addressed. These traditional techniques are no longer satisfactory for monitoring in SDN. This paper presents FlowVista - a lightweight solution to identify flows using the SDN Northbound interface and addressing flows rather than packets. The high flow matching accuracy is achieved by interacting directly with business application residing in the application layer and translating its high level information to low level network flows in a non-invasive fashion. FlowVista is tested using Voice over IP (VoIP) as a business-critical application in both real and simulated networks. In addition, the portability across various platforms, low bandwidth utilization characteristics and integration with many components native to SDN makes FlowVista suitable for those who want to analyze SDN traffic without having to attach expensive tools that often have limited visibility and granularity.**

I. Introduction

Traffic monitoring is an integral element of networks and systems management. Monitoring tools strive to provide information that can be later used to evaluate the Quality of Service (QoS), detect faults and improve security. Traffic monitoring in traditional networks is a well researched area, where many applications were developed in order to aid network administrators in their day-to-day operations. Some of them include solutions built with the use of packet sampling or Deep Packet Inspection (DPI) implemented in software or hardware. However, while trusted by in industry the computational requirements of traffic monitoring continue to impact network performance significantly. Resource intensive techniques, such as sFlow [1], NetFlow [2] or Zabbix [3] require extra CPU and bandwidth, leaving administrators with a constant compromise between accuracy and overhead. The amount of additional resources required to implement monitoring techniques listed above, especially in the multi-vendor environments, where switches or routers may be supplied by various manufacturers, are directly associated with high cost and potential compatibility issues.

The real reason for monitoring in the first place is to ensure the correct operation of business-critical applications and resolve application performance problems before services are noticeably affected. One of the main goals in this work is to provide the reader with set of techniques that can aid faster troubleshooting of networks, but also provide a platform to automate corrective decisions. This work delivers all the means necessary to deploy low-cost monitoring solution without compromising between accuracy and overhead in Software Defined Networks (SDNs).

Nowadays, Customer Relationship Management (CRM) services, Voice over IP (VoIP), video conferencing, and other business-critical applications are deployed on top of SDN in order to achieve the global visibility of the network. SDN enables highly programmable networks with a high level of transparency unlike traditional networks allowing the management of flows on each component in the network. Network impairments can be calculated using statistical information provided by each SDN enabled switch. Unfortunately, borrowing monitoring techniques developed for traditional networks and placing them in SDN, may cause a large overhead due to large amount of queries exchanged between the controller and switches in the network. New techniques should be developed in order to reduce the amount of resources required to monitor application specific traffic. One approach to minimize the overhead is to filter out traffic based on feedback arriving directly from the business application, so that the application becomes the main source of information used to decide on which flows in the network should be monitored. Based on this feedback we can identify switches that require monitoring. The management decisions are still made by the controller, not the service itself, however, the controller gains greater visibility in the network. Ideally the monitoring and decision making should be completed without the need for packet classification. This work presents FlowVista: SDN monitoring platform equipped with out-of-the-box techniques used to identify traffic flows in a non-invasive fashion.

In our view VoIP is in an early adoption of SDN, hence it became our main use case in our experimental evaluation. The arrival of VoIP technology was well received by medium and large enterprises due to a significantly lower usage cost of VoIP phones in comparison to expensive and hard to manage legacy phones. An issue of low call quality however, is still not solved in large enterprise as weak links often contribute to overall low call quality. Furthermore, the legacy and current monitoring techniques are unable to find the exact location of quality

degradation or they are overweighted as they monitor all traffic in the network, even flows that may not be of an interest to the network administrator. This research addresses those issues by accurately identifying flows within the network. By monitoring these flows using non-invasive techniques, such as [4] and [5] the cost of monitoring is significantly decreased.

This work makes the following contributions:

1) We propose techniques to identify and filter out application specific flows based on information provided by business application using Northbound APIs, without performing DPI or packet sampling.
2) We show that flows can be identified accurately using only those functions shipped with SDN and network protocols, such as OpenFlow or NetConf, making it robust and suitable for large scale multi-vendor environments.
3) We evaluate the proposed technique by deploying a realistic VoIP telephony system on top of SDN in the lab environment using both physical and simulated testbeds.

II. Related Works

Currently, there are several choices for traffic monitoring in traditional networks, that are very often applied in the SDN. Among the most popular are sFlow, NetFlow and Zabbix - tools that are installed on dedicated machines, switches or routers and often rely on flow replicators deployed in the network. Our study of those solutions shown that these tools use either systematic or random packet sampling approach on all flows that are in fact very accurate, however in order to gain high accuracy, large amount of packets need to be sampled and forwarded to analytic engines, which may affect the performance of the network. Ideally, there should be no compromise between the accuracy and cost, or at least the side effects of gaining high accuracy should be minimized.

In order to reduce resource consumption, approach different to packet sampling and DPI needs to be developed. One of such ways may be to use high level identifiers, similar to a solution presented by Polcak et al. [6], where the results showed that High Level SDN approach is not only faster, but operational costs can be reduced. The performance of any monitoring solution can be further improved by removing enhanced hosts and tools such as DPI engines, IPFIX probes and other types of collectors.

Jarschel et al. [7] shows the benefits of application control plane, where applications running on top interact with the network itself. The application-state information is used by the controller to choose the best path in the network. Authors were able to achieve reduced bandwidth consumption in the initial ramp-up stages of the transmission when applying to the YouTube streaming application, which is very dynamic, just like VoIP traffic. The initial results were satisfactory however, this approach requires dedicated machine or array of machines used to gather helpful information from the running applications and additional computing resource in the control plane to perform packet analysis task.

sFlow, can be used to monitor business-critical applications such as voice, data, video, without having to employ multiple monitoring applications for that purpose. sFlow is implemented in hardware (network switches/routers) and hence it can operate at line speeds without impacting the switch performance considerably. The sampling is done at the hardware ASICs, which makes it simple and accurate. The packet flow sampling mechanism carried out by each sFlow instance must ensure that any packet observed at a data source has an equal chance of being sampled, irrespective of the packet flow to which it belongs. Taking a sample involves either copying the packet's header, or extracting features from the packet. The biggest drawback of sFlow, as well as Cisco's NetFlow is that all devices in the network need to support those technologies for a comprehensive and complete network analysis.

OpenNetMon [8] - OpenFlow based controller module uses the idea of per-switch monitoring to enable fine-grained traffic engineering. In order to obtain network metrics, probe packets are send every measuring round. However, it is not possible to measure the performance of each link on a per-flow basis, but only the performance of probe packets being injected onto each path in the network, where there could be many flows in one path.

In another work [5] authors propose the per-link monitoring and investigate the ability for the SDN controller to report intermediate MOS (iMOS) for a given set of calls by calculating accurate loss rates and using the simplified E-model formula detailed in the paper. That work uses OpenFlow switch statistics request, and evaluates the cost associated with sending such messages to all switches in the network. It establishes that the overhead is very low, yet it linearly increases as the number of switches in the network increase. The limitation of that work is that every active and non-active switch is queried every second, and even that it outperforms OpenNetMon as no probe packets are injected, other methods for flow selection should be developed in order to reduce the overhead.

A critical aspect that is lacking in the literature is a lightweight solution/mechanism to identify possibly interesting traffic. This narrows the monitoring scope further reducing the effect of monitoring in the network.

III. System Design

A. General architecture and concept

There are four major components in the envisaged SDN architecture presented in Fig. 1. Application layer remains placed on top of the control plane and runs applications such as VoIP, Video, FTP or other applications required by the business. In this work we are assuming that all of the northbound applications are equipped with Application Programming Interface (API) extensions that allow other services to communicate with them. If an API service is not a part of a business application, it would have to implemented in order to work with FlowVista. This can be seen as a limitation, however in today's cloud driven network environment, we observe a growing trend in API service implementation, as once you have an API you can easily move and launch your cross platform applications and scale up. Application

layer communicates with the control layer using Northbound APIs. In the SDN, applications communicate directly with the controller, however if FlowVista monitoring module is enabled, it acts as a proxy between the application and the controller.

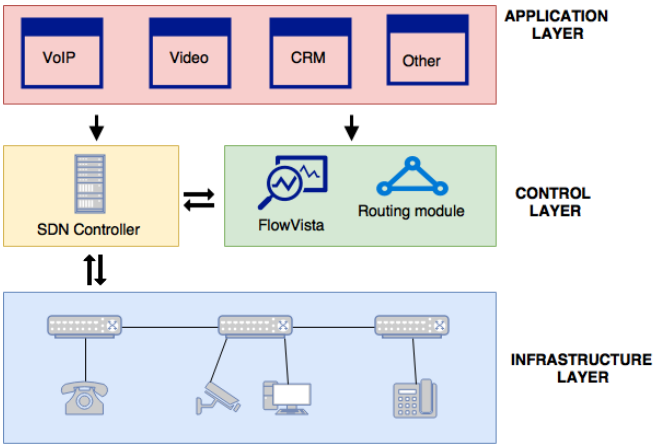

Fig. 1: Business application interacts with FlowVista implemented in the control layer and translates high level information to low level rules using network protocols such as OpenFlow or NetConf.

As shown in Fig. 1, two controller modules are used in this work. The FlowVista is responsible for matching flows and gathering statistics from SDN switches and routing module is used to assure the most efficient path calculation, however the implementation of routing module is not a subject in this paper. The communication between modules and the application is achieved through Northbound API, hence no further proxy is required. OpenFlow protocol is used for southbound communication between the control layer (SDN controller) and the infrastructure layer (physical network devices) in this work, however FlowVista is not limited to OpenFlow only. In this work we were able to avoid using any DPI tools, Quality of Experience (QoE) and monitoring hardware middle-boxes, that impose extra overload on network devices and links in the network. In addition to this and unlike that which is presented by et al. Thorpe [9] there were no modifications made to the switch firmware. Instead, native SDN methods were combined in a novel way and used to calculate the per-link and per-application QoS at the minimum cost. This type of architecture utilizing both Northbound and Southbound APIs allow the user to not only detect flows from A-B, but also identify unique flows in a multi-flow session.

B. Working example using Asterisk

Asterisk [10] is an open source framework for building communications applications and is currently used by nearly 50% of all open source VoIP telephony applications and services. The proposed techniques for filtering application specific flows can be better presented using VoIP as an example of business-critical application.

When a new call is established by Asterisk, usually two channels are created for each call to allow bi-directional communication, which means that two or more flows (in case of a video call) are created in the network. When a call is registered, a list of active calls gets updated in the Asterisk server. This list contains high level call information such as all active channels including user names, IP addresses, MAC addresses or other fields such as office location or priority settings. FlowVista is able to query the list of channels and extract call characteristics. It will then check if the most recent call is currently being monitored, and if it is not it will process the high level information by converting it to SDN controller friendly format - that is from high level JSON to OpenFlow match rules in our case. Flows in this scenario are matched using IP and MAC address of each user. At this stage, traditional models would use DPI or gather sample packets from each flow on the switch in order to find application specific traffic, however as we established this operation comes with an extra load on the network. Instead, FlowVista, queries all switches for matching flows using Southbound API and if successfully matched the list of switches containing matched flows is generated. Correct identification of active switches allows further calculation of flows for which the network statistics should be obtained, ideally every second. The decision to set the polling frequency to one second was motivated by the results of a study in [5], however it should be adjusted based on the traffic characteristics. The main advantage of the proposed system is that we are able to filter out only those switches carrying a VoIP channel. It means that unless we explicitly ask the module to monitor other flows, there will be no queries send from the monitoring module to those switches that are not directly involved in the application specific traffic. It has been noted by us, that most applications placed on top of SDN, tend to impose the lifetime for network flows, usually denoted in seconds. It means that if a call is ended, and the value of flow lifetime is set to for example 100 seconds, the monitoring module will continue to query the flow for statistics until the flow expires. This issue is addressed in FlowVista's implementation by gathering feedback directly from the application in real-time, so that when call is terminated, FlowVista will stop the monitoring of that specific flow immediately. Once flow statistics are obtained, iMOS for each call can be calculated as show by Siniarski et al. [5]. The workflow described above is visualised in Fig. 2.

IV. Implementation and Experimental Evaluation

To evaluate the correctness, scalability and overall performance of FlowVista, multiple tests were carried out in two isolated environments. First, we aimed to show that proposed techniques can be implemented in VoIP over SDN, hence an industry-grade VoIP system was built using physical equipment only. The absence of emulation factors allowed us to obtain the most realistic results for a small scale Enterprise VoIP network. It was also necessary to test the proposed solution in terms of scalability, hence in later stages

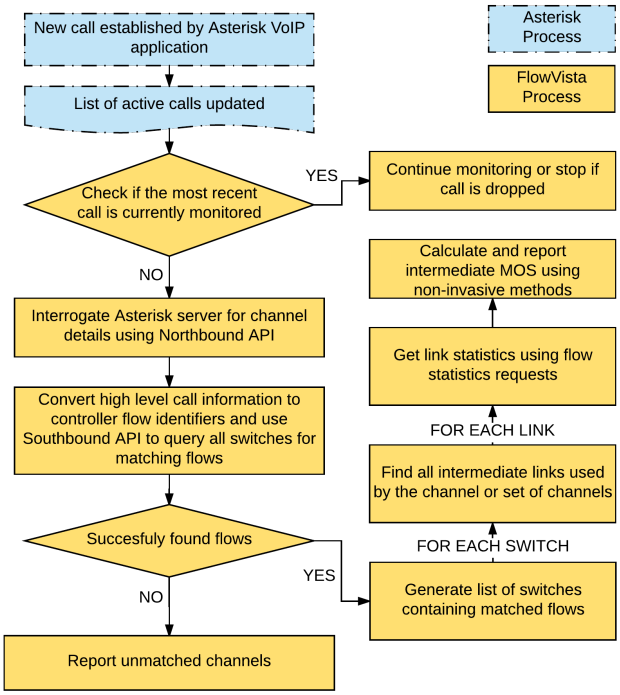

Fig. 2: Sample workflow using FlowVista and Asterisk as a business application.

of experimental evaluation, the network emulator was used to create a much larger network.

A. Physical environment

The layout of physical testbed used in this work is depicted in Fig. 3. It consists of several dedicated hardware and software components. The industry-grade SDN controller - OpenDaylight [11] is running on a 8-core, 24Gb RAM, Dell T5500 server. Asterisk VoIP server is deployed on a 4 core Dell Optiplex 960. Top rack switch (Netgear FS562T) is directly connected to 4 OpenFlow enabled ZodiacFX switches, Asterisk VoIP server and the SDN controller. Two dedicated machines were connected to the edges of the network and used to run Zoiper - VoIP client widely used by businesses around the globe. Zoiper clients use Asterisk server to set up calls, just like in a real-life VoIP telephony, except that the VoIP system is now deployed over SDN, so it uses OpenFlow 1.3 protocol to establish connections and manage network operations.

B. Emulated environment

The high cost of VoIP and SDN enabled switching equipment limits the size of network topology that can be deployed for testing in the lab environment, hence further scalability and performance tests were carried out in an emulated network using Mininet [12]. The linear topology of 4 switches was used for flow matching performance tests in section IV-C1 and the topology of 15 switches was used for the bandwidth overhead calculation in the experiments section IV-C2.

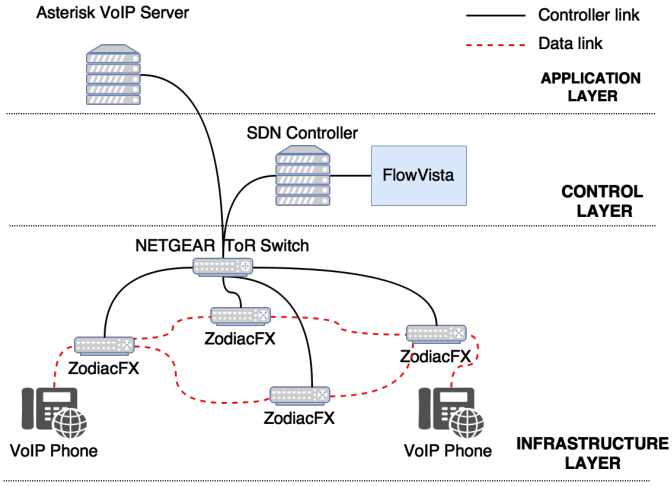

Fig. 3: Physical testbed is built using dedicated hardware. Network switches in the network are OpenFlow enabled and connected to SDN controller, VoIP phones and VoIP server.

Both physical and emulated topologies were managed by the same version of SDN controller - OpenDaylight Beryllium.

C. Experiments description

1) Flow matching performance: Two test were designed to verify FlowVista's ability to accurately obtain channel information produced by Asterisk and finding matching flows on network switches.

In the first test 4 ZodiacFX switches were used to measure the time it takes to match network flows depending on the number of simultaneous searches and the number of active flows installed on the switch. This test was designed to establish a baseline in our experiments. It is important to note that the maximum number of flows supported by ZodiacFX switches is approximately 512 depending on the network layer used for matching. Since, layer 2 identifiers were used to match flows, the maximum of 350 flows were possible to install on each switch. In this test, the number of simultaneous searches was increased from 1 to 10 and at each step the number of active flows installed on the switch was increased from 1 to 350 in steps of 1.

In the second test, 4 physical switches were used initially, however since the maximum number of flows is relatively low to satisfy scalability tests, emulated environment had to be used in order to extend the evaluation of the time it takes to match flows depending on the number of active flows on the switch. In this test, flow rules had been incrementally added on each switch, where single match request was send after every 10 successful installations. There were two objectives in this test. Firstly, what is the performance of FlowVista in terms of the time it takes to match flows as the number of active entries on the switch increases. This is an important metric that can be used in early stages of network design. Secondly, what is the relationship between the number of switches in the network and the flow matching time.

In both tests we ensured that FlowVista sends match requests for flows located at the bottom of each flow table, so that the switch has to perform a full table search to account for the worst case scenario.

2) Bandwidth overhead: Total of three performance tests were designed, to measure the effects of using FlowVista versus other well known solution, in this case - sFlow, all in terms of bandwidth overhead. Each test involved generating VoIP traffic between random hosts in the network and gathering network statistics using both sFlow and FlowVista. Initially, every switch in the network was sFlow enabled, however sampling and polling functions on switches that are not involved in VoIP traffic forwarding were switched off, in order to accurately match the number of switches queried by the proposed solution. Furthermore, the same switches were selected for both sFlow and FlowVista.

The main goal of these experiments was to compare proposed solution with another monitoring technique widely used in the SDN space. Another objective is to understand the amount of bandwidth utilized during the execution time of monitoring functions in the SDN, and how this can be reduced. Sampling rate and polling frequency settings used for each test are presented in Table I.

TABLE I: Sampling rate and polling frequency settings used in experiments detailed in section IV-C2

Test #	sFlow sampling rate	sFlow polling frequency	FlowVista sampling rate	FlowVista polling frequency
1	1:1	1	N/A	1
2	1:10	10	N/A	1
3	1:100	10	N/A	1

V. Results

A. Flow matching performance

Fig. 4 shows that the time it takes FlowVista to match network flows highly depends on the number of simultaneous searches, which can be further optimized using multiprocessing in the implementation of the controller. It is interesting to note that the time it takes FlowVista to match flows on ZodiacFX switches remains unchanged after 30 active flows are installed on the switch until the maximum capacity of flows is reached. It means that ZodiacFX switches are well capable to work at 100% load and support maximum number of entries without the performance loss, however it also raises scalability questions.

It was therefore crucial to evaluate the performance of FlowVista in a much larger environment using Open vSwitches, which support greater amount of flows. Fig. 5 shows the time it takes to match single flow in the network of 1-4 switches, where each switch contains 1-60000 active flows. The upper boundary reflects the average number of flows observed in large enterprise network. The number of flows was incremented by 10 at each step during which the full table search was performed. In the worst case scenario it took a total of 60 milliseconds to find 4 matching flows

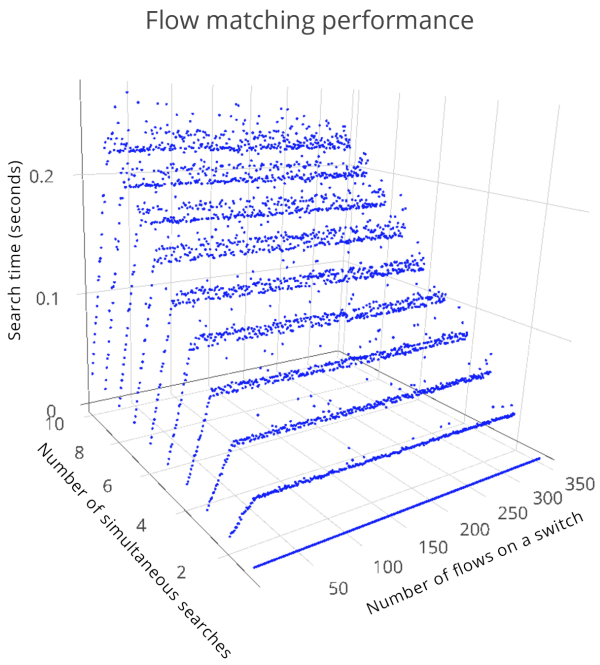

Fig. 4: Flow matching performance using a network of 4 ZodiacFX switches

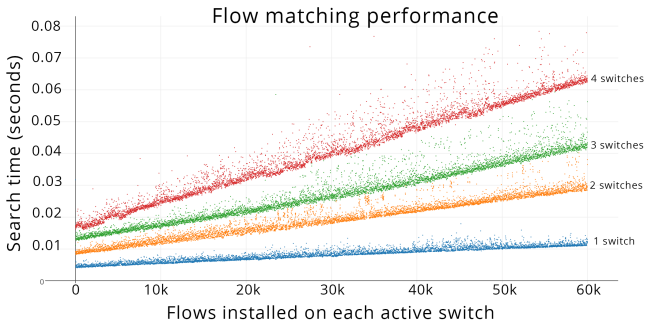

Fig. 5: Flow matching performance in the emulated environment using Open vSwitch and Mininet for up to 60000 active flows on each switch

(one match per switch) with 60000 flows installed on each switch. In addition to this, the performance starts to degrade after approximately 1000 flows in the switch table, which is why the degradation was not observed when using ZodiacFX switches that only allow up to 512 entries and are optimized to support this number of flows.

B. Bandwidth overhead

In the first test, where sFlow's sampling ratio is set to be the most accurate and polling frequency is equal to 1 second, FlowVista significantly outperforms sFlow, as depicted in Fig. 6. The adjustment of sampling rate and polling frequency for sFlow in test 2 effectively increased the performance of sFlow and resulted in comparable performance to FlowVista.

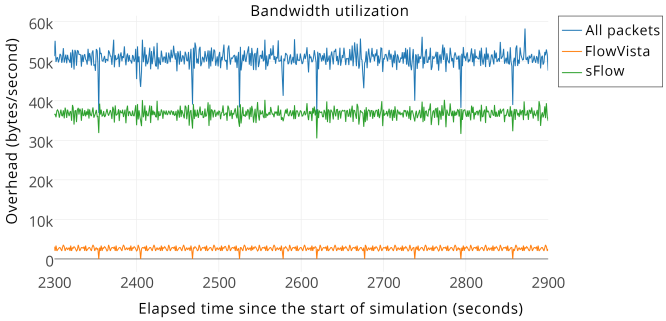

Fig. 6: Switches containing matched VoIP traffic are queried for statistics by sFlow and FlowVista. sFlow's polling rate is set to 1 second and sampling rate is set to 1:1. The polling rate for FlowVista is set to 1 second

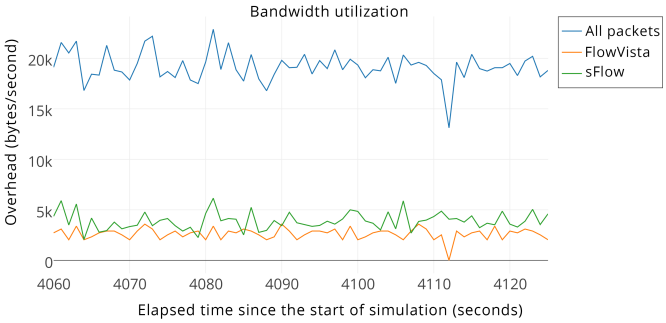

Fig. 7: Switches containing matched VoIP traffic are queried for statistics by sFlow and FlowVista. sFlow's polling rate is set to 10 seconds and sampling rate is set to 1:10. The polling rate for FlowVista remains at 1 second

It means that the overhead associated with running FlowVista is almost identical, however sFlow samples every 10^{th} packet, every 10 seconds. This configuration may be acceptable for certain types of traffic, for example large data transfers, however in a highly dynamic environment, such as VoIP, where MOS needs to be calculated at least every second, using sFlow would not be feasible. Results for this test are depicted in Fig. 7. Packet sampling rate for sFlow was further reduced to 1:100 and the polling frequency remained at 10 seconds in test 3. The overhead associated with using sFlow for monitoring purposes is now lower than in FlowVista, however increasing sampling frequency is associated with the loss of accuracy. Results for test 3 are depicted in Fig. 8. For sFlow to be competitive with FlowVista for accuracy in dynamic environment it requires multiple times the bandwidth usage. sFlow can be configured to be competitive with regards to bandwidth, however, a loss of accuracy will be associated with reduced sampling.

VI. Conclusion

Low-bandwidth SDN monitoring techniques were developed, tested and proven to work in both physical and emulated SDN environments. Network administrators are able to use FlowVista to identify and filter out application specific

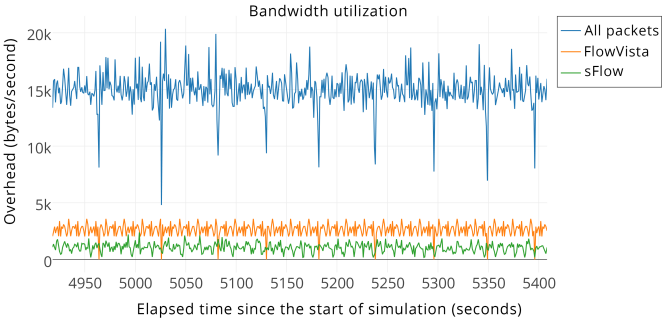

Fig. 8: Switches containing matched VoIP traffic are queried for statistics by sFlow and FlowVista. sFlow's polling rate is set to 10 seconds and sampling rate is set to 1:100. The polling rate for FlowVista remains at 1 second

flows based solely on information provided by the business application and network protocol of choice. Using native SDN functions in a novel way can lead to accurate SDN monitoring on per-flow and per-application basis. Furthermore, it is possible to classify application flows without having to use expensive DPI tools or packet sampling. The portability and extensibility of FlowVista makes it a robust platform for other network management tools in the SDN space.

VII. Future Work

Future work will include building path calculation methods and priority queuing schemes in which the decision is based upon feedback from the monitoring module.

Acknowledgment

Supported, in part, by Science Foundation Ireland grant 10/CE/I1855 and by Science Foundation Ireland grant 13/RC/2094 and by Enterprise Ireland Grant IP20140344.

References

[1] I. Corporation. (2003-2017) sflow - network monitoring. [Online]. Available: http://sflow.org
[2] Cisco. (2003-2017) Netflow - network monitoring. [Online]. Available: http://www.cisco.com
[3] Z. LLC. (2001-2017) Zabbix - the enterprise-class monitoring solution for everyone. [Online]. Available: http://www.zabbix.com
[4] B. Siniarski, C. Olariu, P. Perry, T. Parsons, and J. Murphy, "Real-time monitoring of sdn networks using non-invasive cloud-based logging platforms," in *PIMRC*. IEEE, 2016.
[5] B. Siniarski, C. Olariu, P. Perry, and J. Murphy, "Openflow based voip qoe monitoring in enterprise sdn," in *IM*. IEEE, 2017.
[6] L. Polčák, L. Caldarola, A. Choukir, D. Cuda, M. Dondero, D. Ficara, B. Franková, M. Holkovič, R. Muccifora, and A. Trifilo, "High level policies in sdn," in *ICETE*. Springer, 2015, pp. 39–57.
[7] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, and P. Tran-Gia, "Sdn-based application-aware networking on the example of youtube video streaming," in *EWSDN*. IEEE, 2013, pp. 87–92.
[8] N. L. Van Adrichem, C. Doerr, and F. A. Kuipers, "Opennetmon: Network monitoring in openflow software-defined networks," in *NOMS*. IEEE, 2014.
[9] C. Thorpe, C. Olariu, and A. Hava, "imos: Enabling voip qos monitoring at intermediate nodes in an openflow sdn," in *SDS*. IEEE, 2016.
[10] I. Digium. (2003-2017) Asterisk. [Online]. Available: http://www.asterisk.org
[11] L. Foundation. (2017) Opendaylight - open source sdn platform. [Online]. Available: https://www.opendaylight.org
[12] OpenSource. Mininet. [Online]. Available: http://www.mininet.org