TeleScope: Flow-Level Video Telemetry using SDN

Yu Wang, Chayut Orapinpatipat, Hassan Habibi Gharakheili, Vijay Sivaraman
Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia
Emails:{yu.wang1@student., c.orapinpatipat@student., h.habibi@, vijay@}unsw.edu.au

***Abstract*—Internet Service Providers (ISPs) are struggling to cope with the growing volume of streaming video traffic in their network, and the problem will only exacerbate as Virtual Reality applications proliferate. To classify and manage bandwidth for video streams, current practise is to either sample traffic for offline analysis or deploy middle-boxes for in-line packet inspection – such solutions are inaccurate and/or expensive. In this paper we present Telescope, a low-cost system comprising a commodity SDN switch and a commodity server, to identify and profile individual video flows at line-rate. We develop an architecture that dynamically manages flow-table entries to classify video flows with minimal mirroring of packets; we prototype our solution using a Noviflow OpenFlow switch, coupled with the Bro packet inspection engine and our application on a Ryu controller; lastly, we validate our solution with real video streams in a campus WiFi network, and test its scaling to thousands of video flows using a hardware-based traffic generator. We believe our solution offers great potential for real-time video classification in an operational network at very low cost.**

I. Introduction

Internet Service Providers (ISPs) world-wide are facing a formidable challenge in coping with the volume of video traffic in their network – streaming video already accounts for more than half of Internet traffic in most countries; higher resolutions (HD and UHD) are slated to increase from 53% in 2015 to over 80% by 2020; and virtual reality (VR) is predicted to increase 61-fold by 2020 [1]. Over-provisioning the network is not an option, since the video traffic adapts its rate to use up any and all available network capacity [2]. ISPs are therefore feeling the need to get a better understanding of the video flows traversing their network, so they can better manage bandwidth sharing amongst video flows without degrading user quality-of-experience.

Over the past two decades, various network telmetry and monitoring tools (SNMP [3], NetFlow[4], sFlow[5]) and specialized boxes (DPI hardware and software) have been used to retrieve information of network resources and traffic profiles. The SNMP mechanism is limited to coarse-grained statistics at the port-level. NetFlow is a flow analysis technology whereby aggregated information of IP flows are captured and stored in a cache and then exported based on active/inactive timeouts. NetFlow has two fundamental issues: (a) It does not offer a real-time statistics, since the information of active flows are available after a delay of minimum one minute, and (b) it is not scalable, since the data-plane hardware needs to decode packets, extract information and create entries in the cache, which is then updated and exported. According to Cisco [6], NetFlow export at 10,000 flows-per-second causes around 7% additional CPU utilization. At 65,000 fps, additional CPU utilization jumps to about 22%. sFlow is used for statistical traffic analysis, since it provides packet and counter sampling to an external collector. Therefore, the accuracy of sFlow may not be comparable to NetFlow and depends a lot on selected sampling rate. On the other hand, specialized monitoring boxes such as deep packet inspectors hardware (e.g. Sandvine) or software (e.g. Vedicis) are indeed advanced telemetry tools, since they process "all" data-plane traffic but require large investments in the range of $100K - $1m, depending on the speed of interfaces (i.e. 1G or 10G) and the overall system capacity and feature set. Further, deep packet inspection approaches fail if payload is encrypted.

The research community has developed many proposals for traffic monitoring and identification [7], [8], [9], [10], [11], [12], [13], using various methods ranging from inspecting few bytes in the payload, to processing headers or characterizing the signatures of packets streams. While these methods hold promise for characterising network traffic, their cost/benefit trade-offs are not validated, particularly since they largely use custom (and expensive) hardware for real-time classification and often have little flexibility (i.e. data-plane traffic is statically configured to be analysed in-line or mirrored to an external engine). We believe that Software Defined Networking (SDN) can provide a flexible and low-cost approach to monitoring video traffic that can be realized using any vendor equipment supporting standard OpenFlow interfaces. By dynamically adapting the flow-table rules on the OpenFlow switch, only one or a few initial packets of a video stream can be mirrored for analysis, and the stream can be subsequently characterized based on flow-table counters polled periodically.

In this paper, we present *TeleScope*, a low-cost SDN-based telemetry tool that enables ISPs to profile and characterize video flows in real-time. Our contributions are:

- We develop a "bump-in-the-wire" architecture for characterizing video traffic in terms of end-points, bandwidth, duration, and resolution. Our method protects data-plane forwarding performance, is resilient to control plane failures, and keeps mirror traffic load to a minimum.
- We prototype our solution using off-the-shelf components including a NoviFlow SDN switch, the Bro packet inspection engine, and the Ryu SDN controller.
- We validate our solution in a campus WiFi network to identify video flows from Netflix, Youtube, iView, and Facebook, and evaluate its scalability to thousands of flows using a hardware traffic generator from Spirent.

The rest of this paper is organized as follows: §II describes

prior work on network monitoring solutions, and §III describes our solution approach that captures and evaluates flow-level information. In §IV we describe our prototype implementation used to validate our solution, while in §V and §VI we evaluate the efficacy and scalability of our system. The paper is concluded in §VII.

II. Related Work

Traffic classification: The large body of literature about traffic classification is an evidence of a growing interest from the research community towards this topic. Surveys of existing classification techniques have revealed *accuracy, computing cost and scalability* problems as major impediments [7], [8], [9]. Among widely used approaches; (a) port-based classifiers have become less reliable since modern sophisticated applications use non-standard or random port numbers, (b) payload inspectors come at high cost of processing and fundamentally fail by encrypted contents, (c) statistical or behavioural classifiers are fairly light-weight due to their focus of flow-level information and employ machine learning algorithms to detect various traffic. Most of existing techniques try to classify all traffic by application type, we instead focus on those traffic that contribute to majority of network load. Our TeleScope instead relies on combination of header-based (i.e. IP header 5-tuple) and behavioural information to detect and monitor heavy video flows.

SDN-based monitoring: There are several measurement and monitoring proposals that share our flow-based approach empowered by SDN [14], [15], [16], [17]. OpenSketch [14] proposes a clean slate redesign of data plane to support monitoring in SDN. This however requires replacement or upgrade of data-plane which translates into more investment for network operators. Others instead, consider standard OpenFlow-based approaches for traffic monitoring. For example, PayLess [15], FlowSense [16] and OpenNetMon [17] insert rules and collect per-flow counters in response to standard `PacketIn` and `FlowRemoved` messages respectively. PayLess tries to balance the accuracy of flow-level statistics against the cost of control-plane overhead, by adjusting the time-out attribute of rules, based on their counters (i.e. a large byte-count shortens the time-out). However, we believe that these reactive interactions between the controller and the switch (i.e. `PacketIn` and `FlowRemoved`), impose an expensive cost to the network operator, since it can cause disruption in data-plane if the controller or control-plane fails. We therefore use no `PacketIn` message in our TeleScope architecture for insertion of rules. Further, per-flow counters are polled by the controller proactively in adaptive time intervals.

The work in [18] proposes an interactive tool based on SDN to monitor and visualize the network nodes, links utilization and the state of rules inside each node. The network administrator can configure the time-out of rules as well as the frequency of statistics collection. Likewise, our TeleScope provides the network administrator with an intuitive web-based user interface, abstracting the state of video traffic. Network rules in our system are configured and managed automatically by the network controller. The work in [19] conducted empirical studies to evaluate the accuracy of flow-based measurement operations on OpenFlow switches that are available in the market. Authors found that flow-level counters are not accurate. Our experiments in §VI also shows an average error of less than 1% in byte-counts from a commercial grade NoviFlow switch.

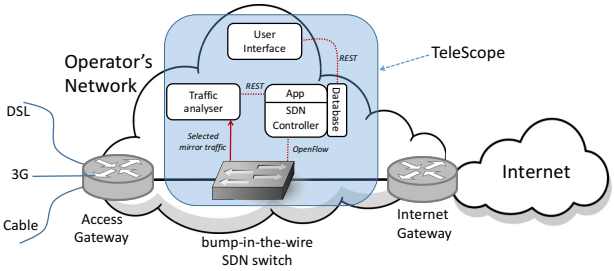

Fig. 1. System architecture

III. System Design and Architecture

We now outline our solution architecture that operates at the flow-level method to detect, measure and monitor video traffic in the network. We first outline the major architectural choices and trade-offs (§III-A), then describe the operational scenario (§III-B), the traffic analyser (§III-C), network rules management (§III-D), and the telemetry algorithm and APIs (§III-E) .

A. Architectural Choices and Trade-Offs

The aim is to design a video telemetry tool for ISPs at low cost and in a programmatic way, so that the network resources are used as efficiently as possible for providing a rich visibility into video traffic. Unlike other proposals that require the use of deep packet inspection (DPI) or other techniques for classifying all network traffic, we advocate the use of dynamic characterization of heavy-load video traffic at the flow level. This requires us to inspect headers of only a tiny fraction of data plane traffic, thereby limiting the processing cost and network bandwidth overheads. The type of flows that need to be processed are chosen proactively and can change according to the video applications. We manage and process flows from cloud-based software, instead of embedding the processing unit into the data-plane that requires large investment and is difficult to maintain. Lastly, thank to our "bump-in-the-wire" architecture, any latency or failure in the control-plane would not disrupt the data-plane and network forwarding process functions independently.

B. Operational Scenario

Fig. 1 shows our system architecture. Each ISP serves its subscribers over various access technologies (e.g. DSL, 3G, or Cable) via an access gateway. The Internet gateway offers the Internet connectivity to the ISP's network. We propose an SDN switch is installed as "bump-in-the-wire" on the link which video monitoring is desired. The SDN-enabled switch is managed by a controller. We propose a "traffic analyser that interacts with the SDN controller via northbound APIs. It

issues requests to the SDN controller on which selected flows are inspected. The controller then programs the bump-in-the-wire SDN switch with rule(s) to mirror selected traffic flows toward the traffic analyser. Therefore, the traffic analyser will be able to actively probe video traffic with specific headers as well as measuring the load of selected flows. Whenever traffic analysis is concluded, then traffic mirroring can be stopped by modifying the pertinent rule(s) inside the SDN switch.

C. Traffic Analyser

The SDN switch is proactively programmed by rules to mirror traffic (to the traffic analyser) that has certain headers (e.g. source IP belongs a known address block). Note that ISPs have a broad visibility into the range of IP addresses that come from popular content providers to their network subscribers. Much of the video enters from Content Delivery Networks (CDNs) such as Akamai, to which the ISP network connects either at public or private peering points; some fraction of video, that is not distributed via CDNs or for which a replica is currently not available from the CDNs, is fetched directly from the source (such as Netflix head-end); lastly, video traffic may also be handed to the ISP from another ISP (via the network-to-network interface). When the mirrored traffic comes in to the traffic analyser, an algorithm is run to inspect the flow headers, e.g. recording source and destination entities. The traffic analyser next requests the controller to install rule(s) corresponding to that video flow. This reactive rule ensures that no more data-plane traffic is forwarded to the traffic analyser. In what follows we describe the rules in more detail and elaborate on their specific match and action fields.

D. Network Rules Management

We advocate a hybrid (proactive/reactive) management of network rules inside the SDN switch. Since specific video traffic is characterized by known packet headers, we propose to push proactive rules into the SDN switch to make a copy of certain packets and forward them to the traffic analyser. Note that these rules will ensure normal forwarding of traffic, along with sending a "mirror" copy to the traffic analyser. Note that in our system no PacketIn message is generated. This allows the SDN switch to provide standard forwarding of data-plane without being affected by the video telemetry process. Low-priority proactive rules are typically configured by a zero time-out to persist forever and match source IP in a specified range. For example, Netflix owns an AS number 2906 that comprises over 130 IP subnets. Thus, having rules that match source IP address of these subnets would capture all Netflix traffic transferred in the ISP's network.

Detection of flows, based on source IP subnets is not sufficient to provide the network administrator with a fine-grained telemetry tool. By receiving the first packet of a flow, the traffic analyser will gain full visibility into the flow (i.e. source/destination IP address and Layer-4 port number). The traffic analyser will therefore instruct the controller to install a pair of higher-priority rules that match 5-tuple header fields. Since the flow table of the SDN switch has a limited size, reactive rules are configured with a non-zero idle time-out (i.e. a reactive rule is removed after a period of inactivity). This allows the controller to reduce the number of state over time. Note that reactive rules no longer mirror traffic to the analyser. This immediately causes the cost of our inspection process is reduced (we will quantify our processing cost in §VI).

E. Telemetry Algorithm and APIs

The primary aim of our TeleScope is to measure, determine and present an abstract state of video traffic to the network administrator via a visual and intuitive user interface. When proactive rules are installed, per-flow statistics (counters and timers) are queried by the controller, stored in a time-series database and tracked in real-time from the user interface. Therefore, we propose a northbound application that operates on the controller for collection of per-flow usage. Further, an algorithm is run to determine attributes of a given flow, i.e. ensuring that it represents a video stream, identifying the content provider (e.g. Netflix, Youtube, or Facebook) that it comes from and estimating the resolution at which the video stream is played (i.e. SD, HD or UHD).

Usage Collection: As reported by other researchers [14], it is crucial to choose an optimal frequency to collect flow-level counters from the data-plane. We found that the usage collection request for all rules causes the switch to return a multi-part reply (e.g. The Noviflow switch returns usage statistics of 25 flows per each reply). This imposes a significant delay to the controller to collect the flow-level usage counter of all reactive rules, specially with large number of entries (e.g. over 10000). We therefore choose to use group tables to collect aggregate counters at a more granular level (i.e. few seconds) without losing per-flow counters. We instead propose that a per-flow usage be collected in an adaptive manner. This means that we adapt the frequency of counters collection based on the number of reactive rules that are mapped to a group table in the switch, to balance the controllers workload against the accuracy of measurement. We group reactive rules based on their content provider's identity (e.g. Netflix or YouTube).

Network APIs: We have designed two APIs that are exposed by the northbound application of the controller, namely (a) [Request] flow insertion and (b) [Query] usage collection. The RESTful flow insertion API is consumed by the traffic analyser by a `POST` method that passes a JSON data including 5-tuple (srcIP, srcPort, dstIP, dstPort, protocol) and the data-path-ID of the SDN switch. The controller next inserts the pertinent rule into the flow-table of the switch. The usage collection API is used via a `GET` request and returns a JSON message containing four sections: usage, controllerStats, stats and flows. The usage section contains aggregated byte counts each provider (e.g. Netflix, Facebook, etc). The controllerStats contains metadata of the controller such as version, up-time, and name. The stats provides the couter of individual flows, while the flows section only contains the most recent detected flows.

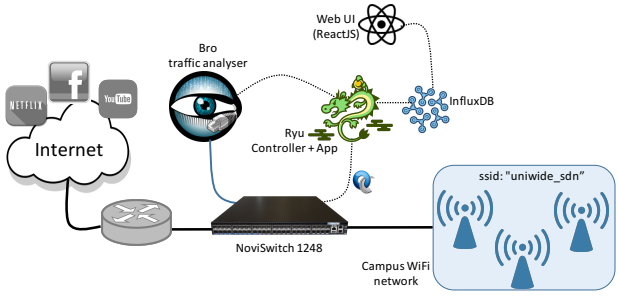

Fig. 2. TeleScope prototype

IV. Prototype Implementation

We have implemented a fully functional prototype of our "TeleScope" system that uses our proposed bump-in-the-wire architecture and APIs to provide the ISP with a video telemetry tool. Our system includes an application of the Ryu controller, and an event handler script of the traffic analyser (Python Binding for Bro) and web-GUI (ReactJS) all operated by the ISP. To emphasize their distinction, each component operates on a separate virtual machine in our laboratory cloud environment that is empowered by industrial scale hypervisor VMware Esxi 6.0. All VMs run Ubuntu server 14.04 LTS and are allocated by one core of CPU, 4 GB of memory and 32 GB of disks.

Our implementation is currently functional in two settings: (a) an SDN-enabled campus network (emulating an ISP network) spanning over 4000 WiFi access points (described in this and the next section), and (b) a point-to-point link over which an industrial scale traffic generator Spirent feeds traffic into our set-up. Our implemented design is depicted in Fig. 2, and `http://129.94.5.46` shows our user-interface live.

SDN switch: Our bump-in-the-wire switch runs on a high performance programmable switch of Noviflow model 1248, and as shown in Fig. 2, exposes standard OpenFlow APIs to the controller and feeds the traffic analyser with selected mirror traffic.

Network Controller: We used the Ryu (v4.0) OpenFlow controller for operating the TeleScope, and developed Python application to implement the algorithm and APIs presented in §III-E (these APIs are exposed via RESTful interfaces to the traffic analyzer and the user interface, as shown in Fig. 2). Successful API calls result in appropriate actions (e.g. network rules insertion and counters collection) at the SDN switch serving data-plane.

Traffic Analyser: We chose an open-source tool, Bro (v2.4.1) [20] for analysing mirror traffic. We developed scripts using Python Binding intermediating between the Bro (parsing 5-tuple header information) and the TeleScope application (calling flow insertion API).

Web Interface: provides the front-end for network administrators to visualize the state of video traffic in their network, and is implemented in ReactJS that uses Rubix template and D3 library. Snapshots are shown in Fig. 3-5, and we encourage the reader to see it live at `http://129.94.5.46`. Upon clicking on "Demo" button at top-right corner, the administrator sees top five recent video flows, aggregated load of video traffic per each content provider, and total number reactive flows.

Recent Flows

Tag	Server IP	Client IP	Total Bytes (MB)	Estimated Rate (Mbps)	Duration (s)
youtube	203.5.76.206	129.94.5.89	7.04		19
facebook	31.13.95.12	129.94.5.87	2.79		32
iView	203.2.218.24	129.94.5.90	40.98	0.70	172
netflix	23.246.29.137	129.94.5.88	70.75	1.59	251

Fig. 3. Recent video flows.

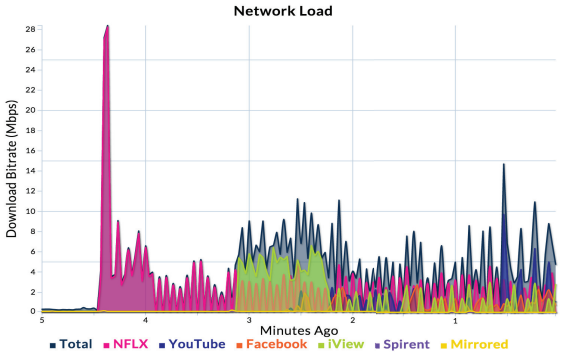

Fig. 4. Load of the network and video traffic.

V. Validation in Campus WiFi Network

We have deployed our "TeleScope" system in a campus network emulating an ISP access network. A guest SSID was created and runs in parallel with the regular University WiFi network, giving us coverage of over 4000 wireless access points across campus to which any user device can connect using existing University login credentials. All wireless traffic is delivered at Noviflow SDN switch (representing the bump-in-the-wire switch). We run our own DHCP server with a /26 public-IP address block taken from our University, and default all outgoing traffic into the campus backbone network. Our controller (Ryu augmented with our TeleScope application) runs on a VM in our laboratory cloud environment.

We connected four user devices across our university campus, including two laptops, an Android phone and an iPhone. For our first set of experiments, we created a scenario with concurrent access to validate our solution: one laptop is playing a movie from Netflix (listed at the bottom row in Fig. 3 with an IP address of 129.94.5.88 and shown by pink line in Fig. 4), and more than a minute later the user of other laptop is watching "The World" program on iView (listed at the second last row in Fig. 3 with an IP address of 129.94.5.90 and shown by green line in Fig. 4), and a while later the iPhone and the Android phone are accessing videos on Facebook and YouTube respectively (listed on top two rows in Fig. 3). Note that one laptop is also running a non-video traffic in the background (captured by black line on top of video lines in Fig. 4).

Fig. 3 shows the abstract information of the most recent video flows that are classified by the TeleScope, including

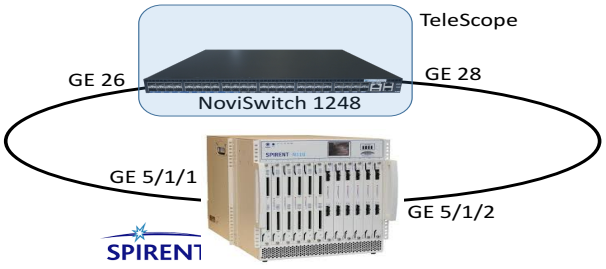

Fig. 5. Performance evaluation set-up

the identity (i.e. tag) of content providers (e.g. Netflix or Facebook), IP addresses of client/server, total volume in Mega Byte, estimated bitrate in Mbps and the duration of each video stream. The last three columns of this tab are updated every 2 seconds. We also note that the TeleScope starts reporting of bitrate only 60 seconds after commencement of each video flow, since streaming video applications typically transfer a large amount of content during their initial buffering period (e.g. the first spike in the load of Netflix stream in Fig. 4). As it is seen in Fig. 3 (the top two rows), the rate of YouTube and Facebook video streams have not yet reported, as their duration (19 and 41 seconds) is less than a minute. The estimated bitrate is computed using an exponential moving average.

Fig. 4 depicts a real-time state of total network load (black line) as well as aggregated load across each video tag (e.g. pink line for Netflix) in a 5-minute window. This allows the network administrator to have an abstract measure of real-time load contribution from each video provider in their network.

Our web interface also shows the total number of video streams per each tag at real-time and the quality (resolution) at which each Netflix video is being played (not included here due to space constraints). We observed that Netflix video streams maintain two concurrent TCP sessions. It cycles through these as the movie plays, making new ones when it closes off old ones.

The Netflix has provided public information of average hourly data usage for different video quality. We have used this reference to identify the resolution of each Netflix video based on its estimated bitrate. This provides the network administrator with a distribution of video quality across the network. The administrator may feed these information into a policy enforcement tool to shape the traffic profile (e.g. limiting the rate of UHD video streams during peak hours to allocate network resources among all users in a fair manner).

VI. Scaling Performance Evaluation

The prior section has shown the usability of our video telemetry tool; in this section we evaluate the efficacy of our system by stressing it with a large number of emulated flows using a traffic generator. We use the Spirent TestCenter [21], which is a high-precision commercial-grade hardware traffic generator with sophisticated capabilities for configuring traffic profiles, synchronising traffic on multiple ports, and accumulating statistics based on pattern filters. Our Spirent main chassis SPT-11U (firmware v4.24.1026) is equipped with a 12-port GE HyperMetric test module. We connected two

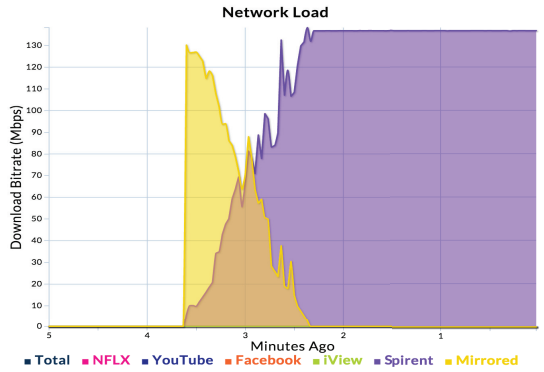

Fig. 6. Network load (11200 flows arrive at the rate of 140 fps)

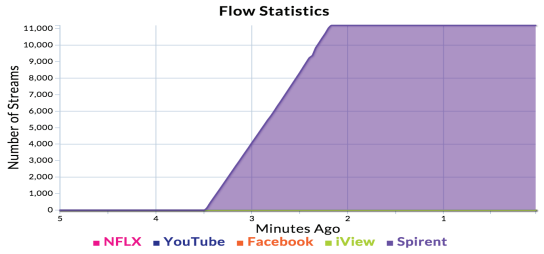

Fig. 7. Flow statistics (11200 flows arrive at the rate of 140 fps)

Gigabit ports of the Spirent to two ports of the NoviSwitch , as depicted in Fig. 5. In our experiment, port GE5/1/1 of the Spirent was representative of video servers and Gigabit port 26 on the NoviSwitch was typically the ingress port, while port 28 was the egress feeding back to the Spirent. Therefore, port GE5/1/2 of the Spirent was representative of clients.

We have written a TCL script to automate the process of traffic emulation. Each group of transmitters and receivers was allocated a distinct /28 public-IP address block, meaning 14 pairs of transceivers. Each pair of transceivers established 10 parallel stream blocks and each stream block operated on a separate layer-4 port number. This generated 140 concurrent flows per second. Further, the port number of each stream block kept circulating using a modifier of range 80 per second, resulting 11200 flows in total. Each flow was sending traffic at a constant rate (CBR) which was uniformly distributed between [800, 1200] Kbps (representative of a 360p video). The emulation was run for 300 sec.

Fig 6 shows the network load at 2s intervals. The user interface of our Telescope shows (by purple line) a steady-state load of 136.81 Mbps, very close to the rate of 137.29Mbps reported by the Spirent statistics (i.e. an error of less than 0.5%). We note that the throughput of mirrored traffic (shown by yellow line) peaks at 130.17 Mbps and falls to zero gradually in 81 seconds. This is not surprising, because our approach only needs the first packet of each new video flow to be sent to the traffic analyser for inspection of its headers, which then a reactive rule is inserted to stop the packet mirroring. The mirror load is directly impacted by the number of concurrent video flows, their rate of arrival and also the speed of header processing in the traffic analyser. If the traffic analyser does not process the first packet of a new flow faster

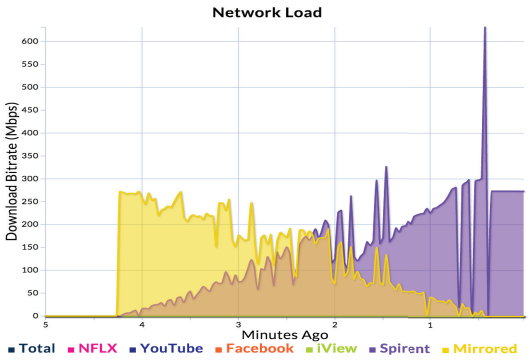

Fig. 8. Network load (31920 flows arrive at the rate of 280 fps)

Total load (offered)	**Total load (measured)**	**Total volume (offered)**	**Total volume (measured)**
274.55 Mbps	273.56 Mbps	6.25 GB	6.22 GB
Mirror load (peak)	**Mirror Bytes (total)**	**Mirror duration**	
271.96 Mbps	4.04 GB	230 sec	

TABLE I
SCALING RESULTS

than the arrival of subsequent packets on that flow, then more packets get mirrored (i.e. more cost of processing). Upon insertion of the reactive rule, no packet from that flow is mirrored and our application thereafter uses byte-counts to monitor flow activity without inspecting packet contents.

At the end of our experiment, the Spirent statistics revealed that 4.48 GB of data were transferred. Our TeleScope application also, measured total 4.46 GB (across 11200 flows) of which 0.68 GB were mirrored. This translates into a high accuracy of 99.55% in measuring the Spirent traffic for this experiment. Moreover, we note that total volume of mirrored traffic accounts for less than 16% of total volume of data-plane. Note that this volume of mirror traffic does not increase, if our experiment lasts longer. This mean that the processing cost is dramatically reduced over a larger time-frame.

Fig 7 shows the number of flows (i.e. representative of video streams) detected by our TeleScope. It can be seen that our telemetry tool performs in a responsive manner and the number of flows rises linearly with the slope of 140 flow-per-second, corroborating with the rate of traffic offered by the traffic generator. It takes 80 seconds for the traffic generator to cover the whole range of port modifier and henceforth no new flow is generated.

Impact of Scalability: We now quantify the performance of the TeleScope by increasing the number of total flows as well as the arrival rate of flows. Since there is a limit of total 32000 flows per each physical port of our traffic generator, we therefore increase the range of port modifier to 114 and double the number of stream blocks (i.e. 20). This setting offers 280 flows-per-second to the TeleScope and generates 31920 flows in total, offering over 270 Mbps of network load. Fig. 8 shows the network load and Table I depicts the result of various measurements.

The total load and volume of the Spirent traffic are measured at a perfect accuracy of 99.64% and 99.52% respectively. Due to heavy rate of flow arrivals, the mirror load has a longer duration (230 seconds) that results a large volume (4.04 GB), since it requires more processing power of the traffic analyser to inspect headers and classify traffic.

VII. CONCLUSIONS

Managing video traffic for ISPs is a critical yet challenging problem. This paper has examined the use of SDN to classify and measure video streams at flow-level granularity. We have shown that our telemetry tool is effective in detecting video flows, without sacrificing the accuracy or incurring high costs of processing. We have validated our solution in a campus network with real video traffic. We have also evaluated the scalability of our system using hardware-based traffic generator. Our results show that flow-based telemetry can achieve most of the benefits of packet-based monitoring, but at dramatically reduced processing costs.

REFERENCES

[1] "Cisco visual networking index: Forecast and methodology, 20152020," Cisco Systems Inc., Tech. Rep., June 2016.
[2] T-Y. Huang et al, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proc. of ACM SIGCOMM*, Chicago, Illinois, USA, Aug. 2014.
[3] J. Case et al, "Simple Network Management Protocol (SNMP)," Internet Engineering Task Force, RFC 1157, May 1990.
[4] B. Claise, "Cisco systems netflow services export version 9," Internet Engineering Task Force, RFC 3954, Octoer 2004.
[5] "sFlow consortium," http://sflow.org/.
[6] "Network performance analysis," Cisco Systems, Tech. Rep., 2005.
[7] A. Dainotti et al, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, pp. 35–40, January 2012.
[8] S. Valenti et al, "Datatraffic monitoring and analysis," E. Biersack, C. Callegari, and M. Matijasevic, Eds. Berlin, Heidelberg: Springer-Verlag, 2013, ch. Reviewing Traffic Classification, pp. 123–147.
[9] T. Bujlow, V. Carela-Espaol, and P. Barlet-Ros, "Independent comparison of popular DPI tools for traffic classification," *Computer Networks*, vol. 76, pp. 75 – 89, 2015.
[10] M. S. Seddiki, M. Shahbaz, S. Donovan, S. Grover, M. Park, N. Feamster, and Y.-Q. Song, "Flowqos: Qos for the rest of us," in *Proc. of ACM HotSDN*, Chicago, Illinois, USA, Aug. 2014.
[11] N. Namdev and S. A. Sanjay Silkari, "Recent advancement in machine learning based internet traffic classification," *Procedia Computer Science*, vol. 60, pp. 784 – 791, 2015.
[12] B. Ng, M. Hayes, and W. Seah, "Developing a traffic classification platform for enterprise networks with sdn: Experiences & lessons learned," in *Proc. IFIP Networking*, Toulouse, France, May 2015.
[13] L. He, C. Xu, and Y. Luo, "vtc: Machine learning based traffic classification as a virtual network function," in *Proc. ACM Workshop on SDN-NFVSec*, New York, NY, USA, Mar. 2016.
[14] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with OpenSketch," in *Proc. of NSDI*, Berkeley, CA, USA, April 2013.
[15] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "Payless: A low cost network monitoring framework for software defined networks," in *Proc. of IEEE NOMS*, May 2014.
[16] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "FlowSense: Monitoring network utilization with zero measurement cost," in *Proc. of Springer PAM*, March 2013.
[17] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "Opennetmon: Network monitoring in openflow software-defined networks," in *IEEENOMS*, Krakow, Poland, May 2014.
[18] P. H. Isolani, J. A. Wickboldt, C. B. Both, J. Rochol, and L. Z. Granville, "Interactive monitoring, visualization, and configuration of openflow-based sdn," in *Proc. of IFIP/IEEE IM*, May 2015.
[19] L. Hendriks et al, "Assessing the quality of flow measurements from openflow devices," in *Proc. of Traffic Monitoring and Analysis (TMA)*, Louvain La Neuve, Belgium, April 2016.
[20] "Bro project," https://www.bro.org/.
[21] "Spirent traffic generator," http://www.spirent.com/.