Technische Universität Berlin

Forschungsberichte
der Fakultät IV – Elektrotechnik und Informatik

A First Look at OpenFlow Control Plane Behavior from a Test Deployment

Dan Levin
Andreas Wundsam
Anja Feldmann
Srini Seetharaman
Masayoshi Kobayashi
Guru Parulkar

Technische Universität Berlin /
Deutsche Telekom Laboratories

Bericht-Nr. 2011 – 13
ISSN 1436-9915

A First Look at OpenFlow Control Plane Behavior from a Test Deployment

Dan Levin∗, Andreas Wundsam‡, Anja Feldmann∗, Srini Seetharaman◇, Masayoshi Kobayashi†, Guru Parulkar†
∗: TU Berlin / Deutsche Telekom Laboratories, Germany ‡: ICSI, Berkeley
◇: Deutsche Telekom Inc., R&D †: Stanford University

***Abstract*—OpenFlow is widely being deployed across campuses in the US, Europe, and Asia. However, little is known about the performance and traffic characteristics of such networks in practice. In this paper, we present preliminary behavior and performance observations made from traffic collected at the Stanford experimental OpenFlow network, the first deployment of its kind operating with user-generated traffic. We base our study on OpenFlow control plane traffic and data plane measurements, collected over multiple weeks. We find that control plane traffic, while typically low, can, at times, make up a significant portion of overall network traffic. We observe that the flow arrival rate has a large impact on network performance. Finally, we identify several tradeoffs and practical limitations in network measurement and monitoring approaches enabled by OpenFlow.**

I. Introduction

Over the past two years, a growing number of researchers have adopted OpenFlow into their campus networks as a means for practically evaluating and experimenting with the principles of *Software-defined Networks* (SDNs). OpenFlow [7] decouples network decision making from forwarding to enable programmability of the control plane while retaining the benefits of standardized forwarding hardware. It defines a standard API that connects the *controller* (the decision making component) to *switches*. The custom programmability of the controller facilitates evaluation and deployment of new approaches to traffic control, management, and monitoring. Example applications include network load-balancing [12], replay troubleshooting for networks [13], and online traffic measurement [6], as well as network virtualization [9].

While OpenFlow deployments are expanding across more campuses, little is known about the behavioral characteristics of such networks in practice. OpenFlow control plane behavior in particular – messages exchanged between controller and OpenFlow switches to install and update the forwarding state – remains largely unstudied in operational environments. Understanding OpenFlow control plane behavior is essential for grasping the fundamental scalability and behavioral aspects of the applications it enables. Measurements can help guide expectations for performance metrics like control plane overhead and flow setup latency. Service disruptions can reveal important considerations for improving SDN robustness. Also, as researchers consider using OpenFlow to improve network visibility [3], [6], we report tradeoffs and limitations of different potential OpenFlow measurement approaches.

The experimental OpenFlow network at Stanford University is the earliest of the campus networks currently in deployment and serves as the setting for our study. We base our study on multiple weeks of recorded OpenFlow control plane traffic and selected data plane measurements, including switch port counters, flow-setup time, and CPU utilization from each of the OpenFlow switches. Our analysis shows that during times of stable activity, in the first week of July and March, the OpenFlow control traffic in the Stanford network makes up 3-4% of the data plane traffic. We also observe service disruptions, when the control traffic can exceed the data plane traffic, indicating potential for future work to handle failure modes and optimization of control traffic. Lastly, our analysis also illustrates tradeoffs between visibility and overhead in OpenFlow network monitoring approaches based on observed control plane flow summary statistics reporting.

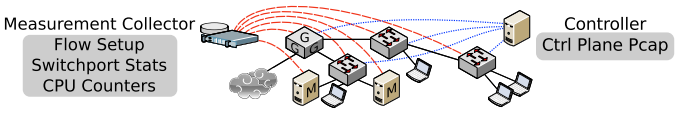


Fig. 1. OpenFlow network and data sources. Dotted lines represent out-of-band control plane links to switches. Dashed lines show out-of-band data plane active measurement. Each switch has 2 in-band active measurement probes "M". Switch "G" acts as gateway to the public Internet.

We, first, in Section II, discuss the environment and dataset on which our study is based, and the methodology used to collect it and verify its integrity. In Section III, we present three main findings of our analysis: Control traffic overhead properties, the impact of flow arrival rate on flow setup time, and observed tradeoffs and limitations in flow summary reporting. We place our study into context of other related work in Section IV, and conclude in Section V.

II. Data and Methodology

Our study is based on data from the OpenFlow experimental network deployed at Stanford University. Conducting measurements on this experimental network poses challenges which shape the methodology we use in both gathering and subsequently analyzing our data.

A. Stanford University Experimental Network

The Stanford OpenFlow experimental wired network exists within the 3A wing of the William Gates Building and handles all workstation traffic for approximately 20 individuals. The deployment uses OpenFlow v1.0 switch hardware and a single commodity server running a different controller at two different time periods, whose logics we later describe.

At any time, the network includes up to 7 OpenFlow switches and hardware devices, representing multiple vendors. The topology is consistent with a workgroup network as found within a campus or enterprise. Figure 1 shows a schematic of the network during presented measurement periods along with relevant measurement points and data sources.

B. Analysis Requirements and Data Sources

To perform our analysis, we take advantage of the experimental nature of the network. Its extensive instrumentation includes the ability to passively and actively measure both control plane traffic and monitor data plane activity, and thus enables us to correlate events over time and by device. At the OpenFlow controller, we collect a complete packet-level trace using the venerable *tcpdump* tool [2]. This trace provides a complete overview of all OpenFlow messages exchanged between any switch and the controller.

From this trace, we determine flow-level statistics such as per-flow forwarding table updates, flow duration, flow size (packets or bytes), flow arrival rate, and total number of active flows. In order to reconstruct OpenFlow control messages from raw packet traces, we use a custom tool built on the `oftrace` [1] library.

Additionally, the measurement server collects, through a dedicated measurement network (dashed lines in Figure 1), several switch health and performance metrics, such as CPU utilization, and port counters. The switches are queried via SNMP in 2 minute intervals.

At the dataplane, measurement probes actively monitor the *flow setup time* (FST). We define FST as the latency experienced by the first packet of a new flow while its forwarding configuration is determined and installed. To measure the FST, we send a ping through the switch in regular 10 second intervals. Depending on the controller logic, the *ICMP request* and its *ICMP response* may each incur a FST above the known baseline link latency.

Certain challenges arise due to the experimental prototype nature of the network and devices, complicating approaches to obtain a consistent view of the network and capture the data necessary for analysis. Our foremost challenge is that the network serves primarily as a testbed for OpenFlow experimentation. Devices regularly come and go, topologies change, and firmware and configurations are often updated. Over 94 days of network activity, 64 days show at least one link topology change for a single switch and 40 days incur link topology changes at two or more switches. Note, also, that different OpenFlow-enabled devices support different types of monitoring. Some switches, for example, do not support SNMP which prevents us from validating OpenFlow control traffic overhead and measuring overall network data rates. Devices which cannot support a required measurement are excluded from the respective analysis.

Depending on the particular analysis, we process, compare, and correlate different data sources. To compute control plane overhead, we use control plane pcap traces processed with *oftrace* to reveal active switches and switch ports. We then validate traffic rates derived from these pcap traces against discovered active switch port counters, collected via SNMP. Similarly, to analyze the performance as experienced by the user, we correlate flow setup latency measurements, switch CPU utilization, and *oftrace* control plane traces. *Oftrace* and SNMP counters enable our analysis of flow summary statistic monitoring tradeoffs and limitations.

C. Dataset selection

Our analysis is based on data collected between May 2010 and April 2011. For consistent analysis, we sub-select time periods from our collected data when the network link topology remains stable. We base the selection on topology information inferred from the OpenFlow Control Plane itself, as well as weekly deployment status reports [10]. To this end, we collect the active switches and ports from the OpenFlow `PKT_IN` messages received at the controller. Furthermore, we infer link connectivity by comparing per-switch port inbound and outbound traffic rates over the switch ports of every switch. This gives us confidence that for a given period, we understand the topology, even when documentation is incomplete.

Consequently, we choose 3 time periods for further analysis: a full week from July 1st, 2010 (T_{full}), and two three-day periods starting July 8th, 2010 (T_{2010}), and March 24th, 2011 (T_{2011}), respectively. According to the status reports, T_{full} represents a period of higher activity with a service disruption, while T_{2010} and T_{2011} represent periods of normal operation.

Table I summarizes the control and data plane network and traffic characteristics observed for these time periods. During T_{full} and T_{2010}, **SNAC** acts as the controller, while during T_{2011} **Bigswitch** is used. **SNAC** uses purely reactive end-to-end L2 shortest-path routing, while **Bigswitch** uses hop-by-hop shortest-path routing with proactively installed table entries for reverse flow direction.

In 2010, the network comprises 7 switches, 5 of which can be monitored via SNMP. In 2011, 4 switches are active, 2 of which are monitored via SNMP. During all periods, we observe more than 200 unique MAC addresses active on the network, and around 50 unique internal IP addresses. The number of external IP addresses varies. Average data-plane throughput, as measured via SNMP from the active switch ports drops from 2.7 Mbit/s during T_{full} to 1.40 Mbit/s during T_{2010}, and 0.982 Mbit/s during T_{2011}. The OpenFlow control plane traffic, as recorded at the controller, drops more significantly: from 0.479 Mbit/s and 388 msgs/s (T_{full}) to 0.315 Mbit/s and 263 msgs/s (T_{2010}) and 0.110 Mbit/s and 81.6 msgs/s (T_{2011}). Note that overall control plane and data plane traffic volumes can not be compared directly, as the control plane also includes traffic from switches that cannot not be monitored via SNMP, hence the need for dataset sub-selection. Furthermore, averaging control traffic overhead over long time periods can be misleading, as we observe that some periods show relatively low overhead, while other periods, e.g., during service disruptions, show greatly increased overhead.

Label	Time period	Controller / Switches	% unique addresses	DP traffic	CP traffic	OF msgs
T_{full}	2010-07-01 00:00:00 2010-07-07 23:59:59	**SNAC** 7 Switches (4 SNMP)	IP: 129 int / 17476 ext 552 MACs	211.24 GB 2.79 Mbit/s	36.22 GB 0.479 Mbit/s	234,625,394 388 msgs/s
T_{2010}	2010-07-08 00:00:00 2010-07-10 23:59:59	**SNAC** 7 switches (4 SNMP)	IP: 65 int / 4295 ext 295 MACs	45.44 GB 1.40 Mbit/s	10.234 GB 0.315 Mbit/s	68,362,186 263 msgs/s
T_{2011}	2011-03-24 00:00:00 2011-03-26 23:59:59	**Bigswitch** 4 switches (2 SNMP)	IP: 50 int / 14404 ext 202 MACs	31,82 GB 0.982 Mbit/s	3,589 GB 0.110 Mbit/s	21,144,775 81.6 msgs/s

TABLE I
DATA AND MEASUREMENT PERIODS (DP: DATA PLANE, CP: CONTROL PLANE, OF: OPENFLOW).

III. ANALYSIS

We now characterize the traffic observed in our measurements. Then, we discuss 3 aspects in which the OpenFlow control plane differs from that in legacy network devices: 1) control communication overhead, 2) flow-level user performance, and 3) visibility. We analyze tradeoffs and limitations in data plane monitoring via control plane flow summary statistics.

A. Control Traffic Characterization

The breakdown of control plane traffic by OpenFlow message type is shown in Table II. For all measured switches[1] during T_{full}, we observe that the control traffic is made up almost exclusively by four message types: `PKT_IN` (23%) toward the controller to obtain forwarding state, `FLOW_MOD` (35%) toward a switch to install forwarding state, `PKT_OUT` (28%) toward a switch, and `FLOW_EXPIRED` (14%) toward the controller to report statistics for an expired flow entry. Further investigation into the control plane message sources reveals a similar distribution for each individual switch. Comparing the values from T_{full} with T_{2010} for the gateway switch **switch** 5 we note higher percentage of `FLOW_MOD` messages. The data from T_{2011} on **switch** 3 exhibits more significant differences, due to different controller logic in use. No `FLOW_EXPIRED` messages are observed, and 47% of messages are `PKT_IN`.

Returning to T_{full}, we further break down the control plane traffic. We note that many OpenFlow control messages are triggered by ARP (6.7%) and UDP ($> 45\%$). Of the UDP triggered messages, most are caused by DNS[2]. When considering all $25,506,658$ flows *installed* during T_{full}, we observe 57% of these flows to be UDP, 33% to be TCP, and 5.7% to be ARP. Of the UDP flows, 85.3% flows are DNS (port 53). Of the TCP flows, 61.5% are HTTP (port 80), 19.56% are SSH (port 22), and 9.3% HTTPS (port 443). This large fraction of ARP and DNS traffic separate work paper [14], indicates a substantial potential for optimization to reduce control overhead, e.g., by anticipating and preinstalling flow table entries, or by handling ARP traffic locally at the OF controller. Furthermore, we observe the controller regularly sends `PKT_OUT` messages containing LLDP datagrams to facilitate discovery of link topology. These messages contribute to roughly 5-6% of the overall overhead. Rearchitecting the topology discovery process may lead to additional overhead reduction in OpenFlow networks.

[1]Only those with SNMP port counters, given in Table I

[2]Network-enabled power meters were found to generate significant ARP and DNS traffic.

B. Overhead Analysis

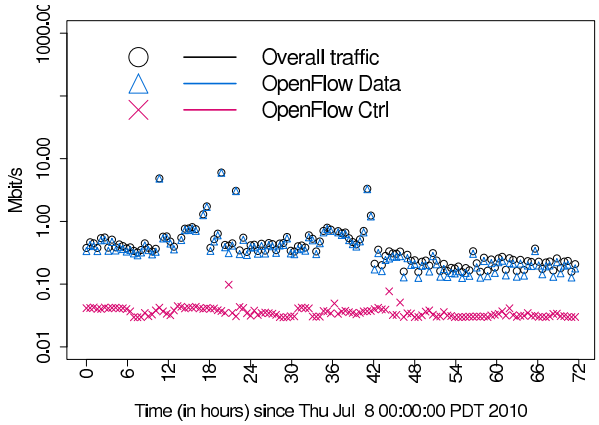
The control traffic overhead, i.e., control traffic as a percentage of data plane traffic, can be seen as an important performance and scalability factor for OpenFlow networks. Here, we present three different perspectives on the OpenFlow control overhead measured in an unoptimized experimental network: (i) Overhead aggregated across all switches for period T_{full}, (ii) Overhead from just the gateway switch July T_{2010} and compare against overhead at the gateway switch in March T_{2011}, (iii) Gateway switch overhead during a service disruption in July, a subset of T_{full}.

(i) Looking at overall network overhead, we choose the first week of July 2010, as this week includes both long periods of normal operation as well as a service disruption incident. The mean overhead for input and output switch traffic during that time is thus 17.3% and 12.9% for this particular week. Note, however, further investigation reveals that the distribution is heavily skewed between the time period of normal operation and the period with the disruption.

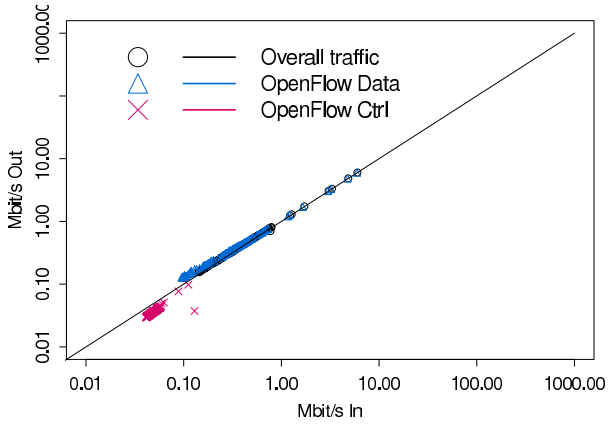
(ii) Next, we compare traffic from two time periods, measured just at the gateway switch, starting with period T_{2010}, over three days in July. Figure 2(a) shows a time series of data plane, control plane, and overall traffic rate measurements, aggregated into bins of approximately 15 minutes. Over this period, we observe the network to exhibit stable behavior. We also see a weak diurnal usage pattern in the data plane traffic, while the control traffic remains relatively constant with mean 0.036 Mbit/s throughput. The average control overhead for this period is 3.9%. We also observe that the relative control overhead decreases as data plane throughput increases and vice-versa, supporting our expectations of a control traffic baseline.

When comparing outbound and inbound data rates, shown in Figure 2(b), we note control plane traffic exhibits low rates, as expected. We observe more incoming than outgoing control plane traffic at the switch. This bias is explained by the higher percentage (58%) of `PKT_OUT` and `FLOW_MOD` messages, see Table II – both message types are sent from the controller to the switch. The data plane is slightly skewed towards outbound traffic, especially for low throughput periods, since low throughput periods are dominated by broadcast traffic, e.g., ARP. As data plane traffic increases, the correlation between inbound and outbound traffic at the switch increases, as most traffic observed at higher data rates is unicast.

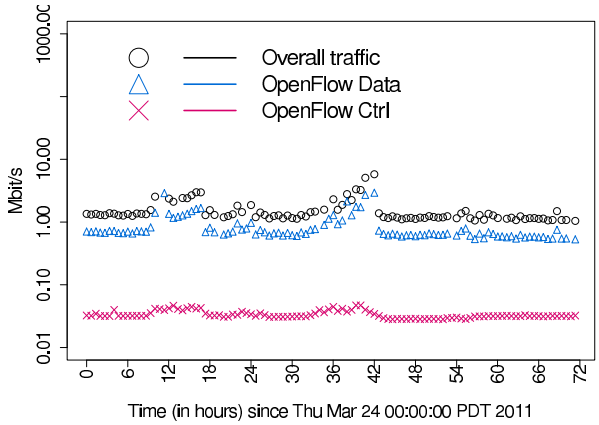
Next we focus on the gateway switch during T_{2011}, in March, and observe some notable differences shown in Figure 2(c) when compared to the period in July. In March, we see higher mean data plane traffic and output control at 1.01 Mbit/s



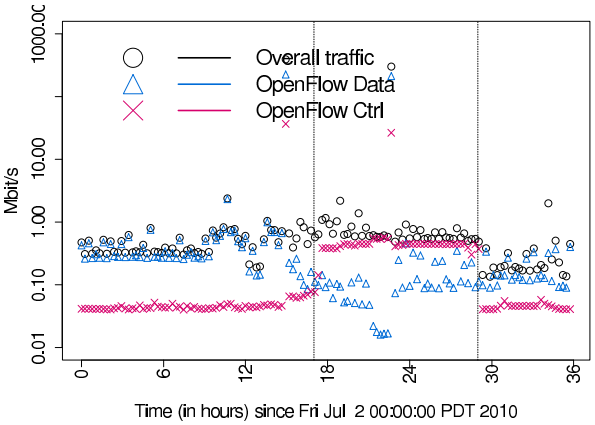
(a) Output traffic: gateway (**switch** 5), July



(b) In vs. out traffic: gateway (**switch** 5), July



(c) Output traffic: gateway (**switch** 3), March



(d) Output traffic: service disruption

Fig. 2. Control plane overhead during normal operation and service disruption

observed period	switches	total msgs	pkt in	pkt out	flow mod	flow expired	other
T_{full}	**switch** 2, 3, 4, **switch** 5	188,774,494	22.89%	28.55%	34.87%	13.51%	0.17%
T_{2010}	only gateway **switch** 5	13,913,584	15.63%	29.84%	28.17%	24.87%	0.5%
T_{2011}	only gateway **switch** 3	9,344,631	47.38%	18.21%	34.40%	*None*	*None*

TABLE II
CONTROL PLANE TRAFFIC BROKEN DOWN BY OPENFLOW MESSAGE TYPE

and 0.033 Mbit/s respectively. The average control overhead for output traffic is 3.3%, lower than the period in July. The diurnal usage pattern on Thursday and Friday is also more pronounced in both data plane and control traffic.

Decreased control overhead in March results from the different controller in use during this period. Unlike **SNAC** that uses exact 12-tuple match in each flow table entry, **Bigswitch** only specifies the MAC-layer and IP-layer headers (i.e., wildcards TCP-layer fields) thus, inserting a lower number of flow table entries compared to **SNAC**, reducing control traffic overhead. Furthermore, these two controllers differ in the forwarding logic they employ: **SNAC** uses purely reactive end-to-end L2 routing, while **Bigswitch** uses hop-by-hop routing with proactively installed table entries for reverse flow direction. Consequently, the nature of the controller traffic differs between these two time periods. One specific difference under **Bigswitch** is that `FLOW_EXPIRED` messages are disabled by configuration, reducing control overhead but depriving us of per-flow summary statistics. Table II shows that the raw number of OpenFlow messages in T_{2011} is lower, and dominated by `PKT_IN` messages with 47%.

(iii) As an experimental network built on prototype hardware, service disruption are inevitably encountered. On July 2 at the gateway switch, we document the effects of one such disruption on control and data plane traffic in Figure 2(d). This plot shows a time series of control and data plane inbound traffic rates at **switch** 5, aggregated over roughly 15 minute intervals. At approximately 18:00 PDT (see first support line) the volume of control traffic increases more than 10 times its value during normal operation. At the same time as the control traffic surges, the total data plane traffic rate drops to roughly 100 kbps. For approximately 12 hours during the disruption, control traffic surpasses data plane traffic until the switch is rebooted the following day (see second support line). During service disruptions, control plane overhead can be significant. The nature of this disruption (which we soon revisit in more

detail) reveals a positive feedback loop within the OF control plane implementation. OF control networks should thus be designed with these failure modes in mind.

C. Flow Setup and Performance from User Perspective

We now examine the impact of service disruption from a user perspective. Note that *flow setup time* plays a crucial factor in the network performance experienced by users. Depending on the flow definition and routing strategy used by a given network, many flow entries may need to be installed for a single user action, e.g., when loading a webpage containing several objects from different destination addresses. The more individual flow table entries required for a particular workload, the higher the overall incurred latency for that workload will be. This differs significantly from a MAC-learning non-OpenFlow switch where each object likely is part of the same source-destination MAC-level flow.

In addition to the propogation delay to the controller, the flow setup time in practice is tightly coupled to the switch CPU utilization. Switch CPU load is, in turn, affected by the rate at which new flows arrive at the switch. Thus, as the new flow arrival rate increases, the flow setup delay increases, potentially leading to positive feedback if new flows are initiated too quickly. This relationship can be seen in the context of the previously discussed service disruption on July 2, 2010. Figure 3 shows the observed flow arrival rate, CPU utilization, and flow setup time per switch.

In Figure 3(a), we note that the flow arrival rate peaks for just one switch (blue 'x' markers), as this switch is the first hop traversed by a large influx of new flows. This large influx of flows results in a spike in `PKT_IN`s sent to the controller. The controller responds with `FLOW_MOD` messages to each of the switches along the path of every flow, triggering a surge in the CPU utilization across each of the other monitored switches as seen in Figure 3(b). A switch from **Vendor B** (black square markers), must cope with this surge in CPU utilization and consequently, it exhibits a higher than normal flow setup time. Figure 3(c) indicates that each new connection traversing this switch (black square markers) experiences an additional delay of up to 300ms. This incident illustrates the relationship between switch CPU load and flow setup time which, depending on hardware implementation, can impact the user experience. It also reinforces the need to detect and limit propogation of local control plane disruptions.

D. Flow Summary Monitoring Tradeoffs and Limitations

Recent work has proposed leveraging OpenFlow to enable pervasive monitoring at arbitrarily defined flow granularities, at every switch in the network [6], [4]. The OpenFlow 1.0 switch specification explicitly requires per-flow packet, byte, and duration counters and defines at two explicit means for obtaining flow summary statistics from the switches.

`FLOW_EXPIRED` messages, captured in our control plane traffic traces, may optionally be reported by a switch on a per-flow basis, upon flow table entry expiration. Alternatively, `STATS_REQUEST` and `STATS_REPLY` messages let the controller request and collect flow table statistics from a given switch at any given time in a polling fashion. Given certain data plane traffic characteristics and monitoring needs, these complementary monitoring approaches present tradeoffs between achievable data plane monitoring resolution and incurred control traffic overhead. A network with short flows will reveal higher monitoring resolution via flow expiration messages. A network with longer flows will reveal higher monitoring resolution with polling, as long as the polling frequency is greater than the flow expiration rate.

We observe the mean rate of 44.25 `FLOW_EXPIRED` messages per second arriving at our controller for T_{2010} with mean message size of 151 bytes. By comparison the mean message size of a `STATS_REQUEST` and `STATS_REPLY` message for this same period are 81.5 and 412 bytes respectively. The median reported flow duration is 5 seconds. Given the short-duration of most flows, the additional monitoring resolution achievable in this network by high-frequency polling is questionable, and expensive, at over 3 times the messaging overhead. These observations underscore the need to optimize the OpenFlow monitoring approach according to data plane characteristics and monitoring objectives.

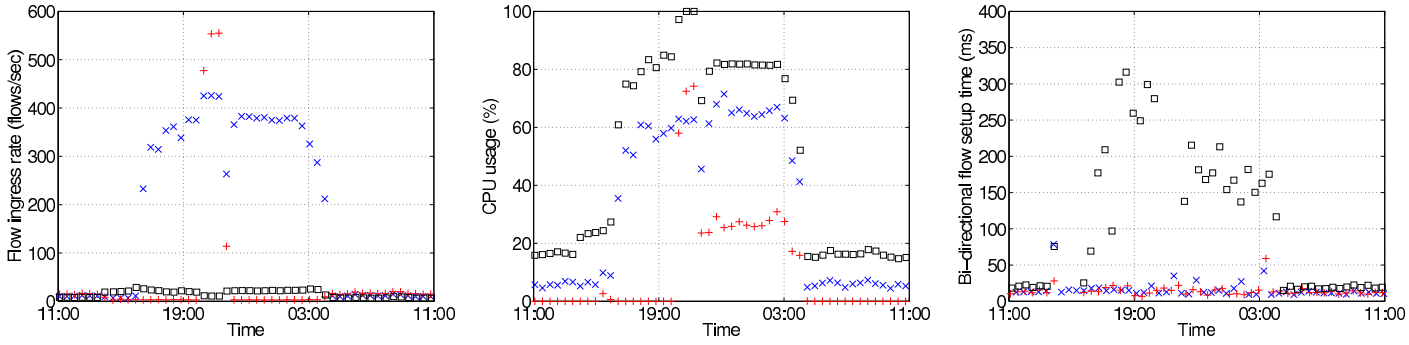
Our analysis also shows that reported flow summary statistics are not always trustworthy. In Table III, we show reported flow summary statistics inaccuracies for different vendors over different time periods. Over the first week in July, a significant number of flows are reported with zero bytes counted from **Vendor A** and **Vendor B**. For both vendors, a small fraction of flows, 0.06% are reported to have more packets than bytes. In summary, for this week, less than 82% of flows from **Vendor A** and fewer than 22% of flows from **Vendor B** are reported with valid byte counts per flow.

IV. Related Work

An earlier work discusses the Stanford OpenFlow-enabled wireless testbed called OpenRoads [14]. It provides a 1-week snapshot of the traffic seen in the production slice of the wireless network that coexists with other experiments. The authors do not capture a view of the control channel behavior or user experience. A recent work called DevoFlow [4] echos our motivation for understanding overhead inherent to SDNs, relying on reasoning approaches in the place of our control plane measurements.

In OpenTM [11], the authors propose to use OpenFlow to estimate the network traffic matrix, with low overhead, and more accurately than when relying on coarse port-level information. This relates to our work in using OpenFlow to obtain network traffic summaries. Similarly, MeasuRouting [8] uses the routing control provided by OpenFlow to improve the monitoring utility of certain subpopulations of the traffic. A recent proposal to enable online measurement of large traffic aggregates [6] leverages OpenFlow switch flow-level statistics collection. Similarly, DDOS [3] detection leveraging flow table statistics has also been investigated.

Several OpenFlow trial deployments exist as part of the GENI project [5] which focus on enabling experiments over a



(a) Flow arrival rate (b) CPU utilization (c) Flow setup time

Fig. 3. Flow-level performance in July, as measured by the flow-setup time, and its correlation to the CPU utilization and the flow arrival rate. Each switch is marked by a different color, consistently across the 3 sub-figures.

observed period	Vendor	# Flows	$n_{bytes} == 0$	$n_{bytes} < n_{pkts}$	$n_{bytes} < n_{pkts} * 64$
Full week from July 1	**Vendor A**	17,528,091	2,271,286 (12.96%)	2,281,980 (13.02%)	5,002,190 (28.54%)
Full week from July 1	**Vendor B**	7,975,170	5,485,542 (68.78%)	5,490,094 (68.84%)	6,239,563 (78.24%)
3 days from July 8	**Vendor A**	7,639,927	992,289 (12.99%)	996,673 (13.05%)	2,128,260 (27.86%)
3 days from July 8	**Vendor B**	3,599,089	2,606,680 (72.43%)	2,608,155 (72.47%)	2,904,512 (80.70%)

TABLE III
PROTOTYPE SWITCH FLOW REPORTING INACCURACIES

substrate that also hosts realistic traffic. Network operators at Indiana University are working on improving the usability of the GENI infrastructure by building a GENI Meta-Operations Center (GMOC) that will monitor networks and provide necessary alerts. Still, the Stanford deployment that we monitor and analyze is the first and to this day largest of its kind. Furthermore, no analysis of traffic patterns in an OpenFlow network has been reported to date.

V. SUMMARY

This paper presents a preliminary look into OpenFlow control plane traffic and behavior from the Stanford experimental deployment. We discuss the challenges of a measurement study in an evolving prototype environment and present results comparing periods from July 2010 and March 2011.

During normal operation OpenFlow control plane messages contributed between 3-4% of the data plane traffic. Exceptional events, e.g., a switch firmware bug on July 2nd, however can cause control traffic to exceed data plane traffic. This network disturbance exemplifies the importance of carefully managing the control plane behavior and control logic to ensure robustness. Data from 2011 suggests that control plane overhead can be significantly reduced by controller logic and configuration. Indeed, there is need and potential for future work to optimize control plane communication. The exact numbers are of course specific to the topology, controller and workload in the measured deployment. Still, the high percentage of observed control plane traffic caused by network support services like ARP, DNS and LLDP topology discovery suggests that these protocols may need to be handled specially to improve scalability, e.g., by preinstalling flow entries for DNS or handling ARP locally at the controller. We also highlight the need to consider visibility and overhead trade-offs and limitations of OpenFlow monitoring approaches.

In future work, we plan to further explore the potential of optimizing OpenFlow control plane traffic by aggregation or proactive installation of forwarding state as well as further potential for OpenFlow-based measurement approaches. We hope that these results may give a first data point for the community, providing an anecdotal view into control traffic behavior exhibited by this emerging technology.

REFERENCES

[1] oftrace. http://www.openflow.org/wk/index.php/Liboftrace.
[2] tcpdump. http://www.tcpdump.org/.
[3] R. Braga, E. Mota, and A. Passito. Lightweight ddos flooding attack detection using nox/openflow. In *IEEE Local Computer Networks (LCN)*, Oct. 2010.
[4] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. Devoflow: scaling flow management for high-performance networks. In *ACM Sigcomm*, 2011.
[5] GENI: Global Environment for Network Innovations. http://www.geni.net.
[6] L. Jose, M. Yu, and J. Rexford. Online measurement of large traffic aggregates on commodity switches. In *Proc. USENIX HotICE*, 2011.
[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38:69–74, March 2008.
[8] S. Raza, G. Huang, C.-N. Chuah, S. Seetharaman, and J. Singh. MeasuRouting: A framework for routing assisted traffic monitoring. In *Proceedings of IEEE INFOCOM*, March 2010.
[9] R. Sherwood, G. Gibb, K. kiong Yap, M. Casado, N. Mckeown, and G. Parulkar. Can the production network be the testbed. In *USENIX OSDI*, 2010.
[10] Stanford OF Network Status Reports. http://tinyurl.com/sofweekly.
[11] A. Tootoonchian, M. Ghobadi, and Y. Ganjali. OpenTM: Traffic Matrix Estimator for OpenFlow Networks. In *Proc. PAM*, pages 201–210, 2010.
[12] R. Wang, D. Butnariu, and J. Rexford. Openflow-based server load balancing gone wild. In *Proc. USENIX HotICE*, 2011.
[13] A. Wundsam, D. Levin, S. Seetharaman, and A. Feldmann. OFRewind: Enabling Record and Replay Troubleshooting for Networks. In *Proc. USENIX ATC*, June 2011.
[14] K.-K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, and N. McKeown. The Stanford OpenRoads deployment. In *Proceedings of the 4th ACM International workshop on Experimental evaluation and characterization (WINTECH)*, pages 59–66, 2009.