Meticulous Measurement of Control Packets in SDN

Yash Sinha
Dept. of Computer Science & Information Systems
Birla Institute of Technology & Science, Pilani, India
h2016077@pilani.bits-pilani.ac.in

Shikhar Vashishth
Dept. of Computer Science & Automation
Indian Institute of Science
Bangalore, India
shikhar.vashishth@csa.iisc.ernet.in

K Haribabu
Dept. of Computer Science & Information Systems
Birla Institute of Technology & Science, Pilani, India
khari@pilani.bits-pilani.ac.in

ABSTRACT

The data packet statistics sent by OpenFlow compliant switches cumulatively includes statistics about control traffic which is used for network control and management. This reduces the accuracy of calculation of QoS metrics and thus hampers network monitoring. We present here a novel algorithm to accurately measure the fraction of control packets in SDN within 3% error rate.

CCS Concepts

•**Networks** → **Network measurement;** *Network monitoring;*

Keywords

Network Monitoring, Software Defined Networks (SDN)

1. INTRODUCTION

A fraction of traffic in the network is responsible for network control & management like monitoring, enforcing security, calculating QoS metrics etc. It consists of packets for network protocols like MDNS, NDP, MLD, DHCP etc. Often these packets are generated & absorbed at the intermediate switches. The OpenFlow compliant switches send cumulative statistics of sent & received packets to the SDN controller that includes control packets. Such packets, although not a part of the end-to-end data traffic, get counted and act as noise in the data packet statistics. Thus, the accuracy of network monitoring services (such as calculation of QoS metrics) that depend on these statistics gets hampered.

In this paper, we propose an algorithm to measure the fraction of the control traffic in Software Defined Networks. Since the number of control packets transferred periodically depends on the number of switches & hosts in the network, we demonstrate a measurement technique, using spanning tree information about the topology. The reported number of control packets for each port within the switches falls within 3% error rate.

SOSR '17, April 3–4, 2017, Santa Clara, CA, USA

DOI: http://dx.doi.org/10.1145/3050220.3060604

2. IMPACT OF CONTROL PACKTS IN SDN

Many services run in the SDN controller continuously in the background that generate a lot of control packets, e.g. topology discovery, network monitoring via packet injection, pushing configurations etc. Therefore, more amount of control packets are generated in SDN than traditional networks. Pakzad et al.[4] have shown that considerable number of LLDP packets are injected by the controller for discovering links. Because of critical load on the controller & scalability issues of an SDN [5], authors in [3] have advocated a need to make a trade-off between resource overhead & measurement accuracy. Thus, PayLess [1] proposes a frequency adaptive statistics collection scheduling algorithm & Pakzad et al.[4] propose a new approach to reduce processing cost due to topology discovery in the controller with a minimum reduction of 67% in terms of messages. For auto configuration in SDN [2], extensions to current protocols such as DHCP-SDN have been proposed that will lead to even greater fraction of control traffic. Thhrough these works, we emphasize that considerable amount of control traffic is generated that distorts traffic statistics in SDN.

3. METHODOLOGY & EVALUATION

3.1 Intuition and Basis

We present the intuitions we got after analyzing network traffics from various emulations of different topologies such as fat tree, mesh etc. with varying node numbers.

1. The message exchanges are periodic in nature e.g., a router may send messages every 30 sec. to adjacent routers for network discovery. So, with the help of number of cycles elapsed, the number of messages exchanged can be known.
2. The number of control packets in a subnet being proportional to the number of switches & hosts in the network can help to estimate the total number of control packets.
3. Using the spanning tree information, the number of control packets through each link can be estimated

In our initial explorations, we found out that around 3200–3500 packets are generated every 30 minutes even in a small emulated network of 2 hosts & 2 switches, which is around 3-9% of the total traffic. Cumulatively, over a period of time, these statistics affect the calculation of metrics. Further, each of the protocols have separate rate of control packet generation. So, instead of identifying a list of triggers of control packet generation for so many protocols, (which will be even hectic in a real network), an approximation method to detect a periodic time interval is more practical & deployable.

3.2 Algorithm

The major steps of the algorithm are as follows:

1. Get the spanning tree information, T of the entire network from the controller.
2. Calculate the number of network devices in the subnetwork of each interface (port) of every network device.
3. Estimate the time period of the periodic message exchanges (τ) & the count of control packets transferred between two switches, & between one host & one switch respectively (A_τ & B_τ), in τ time period.
4. For time t, based on the count of network devices for each interface calculate the number of packet transmitted using the following formula: $\boldsymbol{N = (A_\tau \times \alpha + B_\tau \times \beta) \times (t/\tau)}$

Algorithm 1 Control packet Measurement

α: Count of switches in an interface's subnetwork
β: Count of hosts in an interface's subnetwork
A_τ: Count of control packets transferred between two switches in τ time period
B_τ: Count of control packets transferred between one host & one switch in τ time period
$T.\alpha_i.p_j.len$ is the number of switches connected to p_j of α_i
$T.\beta_i.p_j.len$ is the number of hosts connected to p_j of α_i

1: **procedure** MEASURE(T(α, β), A_τ, B_τ, t, τ):
2: **for all** switches α_i in spanning tree T do **do**
3: **for all** switch ports p_j do **do**
4: $N(p_j) = (A_\tau \times T.\alpha_i.p_j.len + B_\tau \times T.\beta.i.p_j.len) \times (t/\tau)$
5: **end for**
6: **end for**
7: **end procedure**

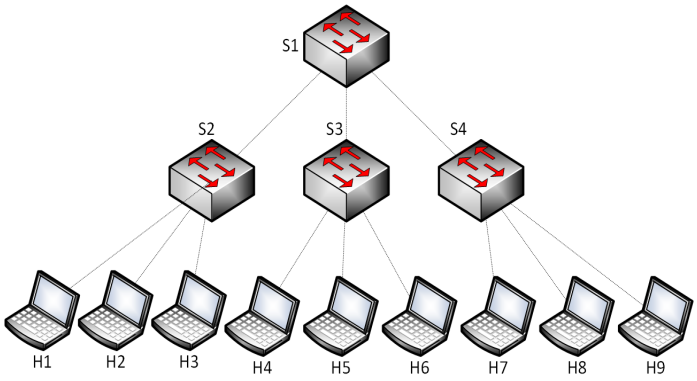

Figure 1: Emulated Topology

3.3 Results

Getting spanning tree information We run spanning tree protocol at the Ryu controller using OpenFlow 1.3. Control of values are possible by sending switch a Port Modification message. Initially, to receive BPDU packets at the controller, we install the required flow entry which discharges Packet-In BPDU packets in each switch. To control sending/receiving of BPDU packets, MAC learning is employed. On completion of the connection between the controller & each OpenFlow switch, the interchange of BPDU packets starts & selection of root bridge & port role & state setting take place.

Table 1: Error rates for various topologies

Topology	#Switches	#Links	Degree	Error
Tree	4	12	4	2.948%
Ring	16	32	3	1.8%
Star	5	8	2.4	1.77%
Mesh	4	14	3.5	3.15%

Estimating time period & other constants For estimating time period, we emulate a network where no traffic, except control traffic is generated. With the help of controller, we poll the switches every 5 seconds for PortStats for 30 minutes. The time period of the periodic pattern observed. For Mininet emulated networks of varied size & complexity, we found that on an average 345 (A_τ) packets are exchanged between two switches in every 450 (τ) seconds. The number of packets exchanged between a switch & a host was on an average 115 packets (B_τ).

Table 2: Control packet measurement

Switch/Port	Switch 1	Switch 2	Switch 3	Switch 4
Port-1	2851	471	477	474
Port-1 est.	2070	460	460	460
Port-1 error	3.19%	2.34%	3.56%	2.95%
Port-2	2833	470	473	480
Port-2 est.	2070	460	460	460
Port-2 error	2.57%	2.12%	2.74%	4.16%
Port-3	2826	470	468	476
Port-3 est.	2070	460	460	460
Port-3 error	2.34%	2.12%	1.71%	3.36%
Port-4	-	7170	7081	7245
Port-4 est.	-	6900	6900	6900
Port-4 error	-	3.76%	2.55%	4.76%

4. CONCLUSION

While emulating various topologies (ring, tree, mesh, star etc.) with different number of switches (4–16), links (8–32) and switch degrees (avg. 3.1), we measured the control packets transferred and compared it with qdisc linux utility for accuracy (Table 1). We successfully report the control packets within 3% error rate approximately for the networks one of which is shown in Figure 1 and Table 2, thus increasing accuracy of calculation of QoS metrics like packet loss.

5. REFERENCES

[1] CHOWDHURY, S. R., BARI, M. F., AHMED, R., AND BOUTABA, R. Payless: A low cost network monitoring framework for software defined networks. In *Network Operations and Management Symposium (NOMS), 2014 IEEE* (2014), IEEE, pp. 1–9.

[2] KATIYAR, R., PAWAR, P., GUPTA, A., AND KATAOKA, K. Auto-configuration of sdn switches in sdn/non-sdn hybrid network. In *Proceedings of the Asian Internet Engineering Conference* (2015), ACM, pp. 48–53.

[3] MOSHREF, M., YU, M., AND GOVINDAN, R. Resource/accuracy tradeoffs in software-defined measurement. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking* (2013), ACM, pp. 73–78.

[4] PAKZAD, F., PORTMANN, M., TAN, W. L., AND INDULSKA, J. Efficient topology discovery in openflow-based software defined networks. *Computer Communications 77* (2016), 52–61.

[5] TOOTOONCHIAN, A., GORBUNOV, S., GANJALI, Y., CASADO, M., AND SHERWOOD, R. On controller performance in software-defined networks. *Hot-ICE 12* (2012), 1–6.