Analytical Modeling of End-to-End Delay in OpenFlow Based Networks

Azeem Iqbal, Uzzam Javed, Saad Saleh, JongWon Kim, Jalal S. Alowibdi, Muhammad Usman Ilyas, *Senior Member, IEEE*

***Abstract*—OpenFlow enabled networks split and separate the data and control planes of traditional networks. This design commodifies network switches and enables centralized control of the network. Control decisions are made by an OpenFlow controller, and locally cached by switches, as directed by controllers. Since controllers are not necessarily co-located with switches that can significantly impact the forwarding delay incurred by packets in switches. Only very few studies have been conducted to evaluate the performance of OpenFlow in terms of end-to-end delay. In this work we develop a stochastic model for the end to end delay in OpenFlow switches based on measurements made in Internet-scale experiments performed on three different platforms, i.e. Mininet, the GENI testbed and the OF@TEIN testbed.**

***Index Terms*—OpenFlow, stochastic modeling, end-to-end delay, Mininet, GENI, OF@TEIN.**

I. Introduction

The distributed control plane of the Internet Protocol (IP) in traditional networks is responsible for routing packets from source to destination. With the exponential spread of the Internet, traditional networks have become complex and difficult to (re)configure and manage [1]. Implementing complex and high level network policies requires network operators to translate those policies into a set of vendor specific device configurations for each each individual switch. Moreover, networks have to be easily reconfigurable to quickly changing network conditions. This level of re-configurability and adaptability to network loads is almost absent in traditional networks. Enforcing network policies in such dynamic networks is a very challenging task. Today's networks are vertically integrated, which means the control plane (responsible for routing decisions) and the data plane (responsible for executing routing decisions) reside within closed network devices running proprietary code that stymies innovation in networks. In current networks, designing, evaluating and then deploying a new network protocol can take at least 5 to 10 years.

Software defined networking (SDN) [2] is a new networking paradigm, which provides a solution to this limitation in current networks by smartly managing and configuring network devices. SDN makes the network programmable by separating the control plane of the network from the data plane. The data plane comprises of only switches with the capability of forwarding packets according to simple instructions received from the control plane. SDN simplifies administrators' control over entire networks through a centralized control plane running in software, hence reducing operational cost. It has also allowed researchers to experiment on deployed networks without causing any interference to their traffic.

The centralized control plane consists of a network controller with a *southbound* application programming interface (API), to communicate with down-stream network hardware comprising the data plane, and a *northbound* API, to communicate with network applications [3]. OpenFlow is an open protocol for the southbound API, promoted by the Open Networking Foundation (ONF) [4]. OpenFlow provides open inter-operability between network equipment of different vendors, thus commodifying network equipment. OpenFlow evolved from the Ethane Project of the Clean Slate project at Stanford University. In that project, network administrators applied one policy to all switches. The project's motive was to remove deficiencies in the design of the Internet. This program led to further projects, among them OpenFlow. OpenFlow was initially used in some campus networks in the United States. Figure 1 depicts the structure of an SDN using OpenFlow at the controller's southbound API. OpenFlow [5] is often considered synonymous with SDN, but it just a part of the SDN architecture.

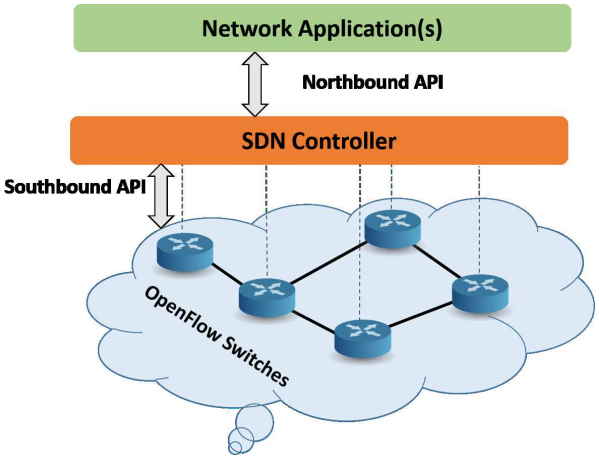


Fig. 1: SDN Architecture

A. Iqbal, U. Javed, S. Saleh and M.U. Ilyas were with the Department of Electrical Engineering, School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad – 44000, Pakistan e-mail: {13mseeaiqbal, 13mseeujaved, saad.saleh, usman.ilyas}@seecs.edu.pk.

J Kim. is with the School of Information and Communications, Gwangju Institute of Science and Technology (GIST), Cheomdangwagi-ro, Buk-gu, Gwangju, Republic of Korea e-mail: jongwon@nm.gist.ac.kr.

J.S. Alowibdi and M.U. Ilyas are with the Department of Computer Science, University of Jeddah, Dhahban – 23881, Saudi Arabia e-mail: jalal.alowibdi@gmail.com, usman@ieee.org.

A. Problem Statement

The OpenFlow protocol helps manage various elements of a network, e.g., implementing configuration changes in the data plane of complex networks typical of data centers and telco core networks. However, these configuration instructions must reach data plane elements in a timely manner. In traditional networks the control plane is distributed across individual switches which rarely affects the performance of the data plane. The delay incurred in OpenFlow-based data plane elements (e.g., switches and routers) to process packets increases due to the involvement of a central controller. The increased delay is due to: (1) the propagation delay of the communication channel between data and control plane, (2) the processing speed of the controller, and (3) the responsiveness of OpenFlow switches to find a matching flow table entry and/or enter and update flow entries [6]. Collectively, the sum total of these delays incurred by a packet at a switch is the store-and-forward delay. The goal of this study is to analyze and model the characteristics of the end-to-end delay between source and destination in OpenFlow enabled networks.

B. Prior State-of-the-Art

The present state-of-the-art in measurement and modeling of packet latency in OpenFlow enabled network can be divided into two categories, i.e:

1) Queuing theory based models [7] [8] [9]
2) Measurement based models [10] [7]

Queuing theory based models assume Poisson arrivals of packets and exponential distribution for traffic. In reality Ethernet traffic has been found to be self-similar (fractal) in nature. Leland *et al.* [11] demonstrated that Ethernet traffic cannot be accurately modeled by a Poisson process. Chilwan, Ameen *et al.* [8] provided a model built on queuing theory, but it was evaluated against simulations.

Ciucu and Schmitt [12] took an alternative approach to classical queuing theory by using network calculus. Network calculus has two branches: deterministic network calculus (DNC) and stochastic network calculus (SNC). DNC only provides worst-case bounds on performance metrics. The models built using network calculus used DNC, and their results are still far from practical use.

Measurement based models are only evaluated against simulations and small-scale lab setups. These models do not capture real network effects on the packets delays in OpenFlow enabled networks. Models based on measurements must be based on data from diverse setups, including Internet-scale experiments.

C. Proposed Approach

We developed a stochastic model based on measurements taken from three different OpenFlow switch platforms which include the Mininet emulator, the OF@TEIN testbed and the GENI testbed. These models will help us better understand the end-to-end delay characteristics of Internet traffic in networks using OpenFlow controlled switches. First, we made measurements on Mininet (a virtual network emulator), which we compared to measurements taken from physical switches, to assess its accuracy in terms of delay. Secondly, measurements for the end-to-end delay are taken on the Internet-scale OF@TEIN testbed that spans across nine different countries. Thirdly, we made measurements on the GENI testbed, which is also an Internet-scale testbed spread across the US. These two testbeds, i.e., OF@TEIN and GENI, are used to analyze the performance of OpenFlow enabled switches in terms of end-to-end delay in real traffic scenarios. Finally, we analyze the measurements from across these three platforms and developed a stochastic model for the end-to-end delay.

D. Key Contributions

The stochastic model that we have developed will help network designers and administrators anticipate expected end-to-end delays in WANs, overlay WANs and Internet links built using OpenFlow switches. According to the authors' best knowledge, this the first time that such a comprehensive measurement study for the end-to-end delay in Internet-scale OpenFlow networks has been conducted. Previous studies have been conducted either on SDN emulators like Mininet and OMNeT++. There was a gap in end-to-end delay measurements in large scale, production level SDN networks. We have also compared the results of two large scale testbeds OF@TEIN and GENI with the Mininet emulator.

II. Related Work

Very few studies have been conducted so far to see the effects of centralized control plane in SDNs in terms of end-to-end delay performance metric. Bianco *et al.* [13] measured throughput and latency of an OpenFlow softswitch built on an Ubuntu PC that performed L2 Ethernet switching using a Linux bridge (`bridge-utils`), L3 IP routing using Linux IP forwarding and an OpenFlow controlled virtual switch for different packet sizes and load conditions. He *et al.* [12] performed experiments on four hardware SDN switches to measure the latency involved in generation and execution of control messages. They focused their attention on the insertion delay and the effect of a rule's position number in the flow table. Huang, Yocum and Snoeren [14] measured and compared the latency across three different hardware switches and an Open vSwitch (OVS) [15] softswitch, and built an OVS-based emulator for physical switches whose latency closely mimics that of a particular physical switch. Sunnen [16] compared the delay performance of legacy switches to an NEC OpenFlow switch.

Levin *et al.* [17] compared the performance of centralized control planes to that of distributed control planes. This study was motivated by the question of whether centralized control planes can provide the same reliability and scalability as distributed ones. Heller, Sherwood and McKeown [18] also considered scalable and fault-tolerant OpenFlow control planes that used more than one controller.

There has also been some work on the development of performance analysis benchmark suites for control and data planes of SDNs. Among them is OFLOPS [19], an open framework for performance measurement of OpenFlow-enabled

switches. Cbench [10] is used to benchmark controller performance. These studies evaluated performance of OpenFlow network architectures using experiments on hardware or simulations on different OpenFlow platforms.

However, to the best of our knowledge, Jarschel *et al.* [7], Chilwan *et al.* [8], Yen *et al.* [9], Azodolmolky *et al.* [20], Bozakov *et al.* [21] and Samavati *et al.* [22] are the only studies to-date to have developed delay models for OpenFlow networks. Jarschel *et al.* [7], Chilwan *et al.* [8], and Yen *et al.* [9] developed delay models using queuing theory. However, Azodolmolky *et al.* [20] and Bozakov *et al.* [21] used network calculus for their delay models.

Both approaches made some unrealistic assumptions for analysis. Jarschel *et al.* [7], Chilwan *et al.* [8] and Yen *et al.* [9] assumed Poisson packet arrivals, whereas several studies have demonstrated that Ethernet traffic is self-similar (fractal) in nature, and is not accurately modeled as a Poisson process [23]. On the other hand, network calculus is a relatively new alternative to classical queuing theory. It has two branches: Deterministic network calculus (DNC) and stochastic network calculus (SNC). DNC, used by both Azodolmolky *et al.* [20] and Bozakov *et al.* [21], only provides worst-case bounds on performance metrics and yields result that are of little practical use [24].

Samavati *et al.* [22] provided the comparison of performance in OpenFlow networks of different topologies. They used graph theory for performance evaluation but did not consider controller-switch interactions in any significant detail. Bovy *et al.* [25] have done experiments for the end-to-end delay over traditional networks and found that classification of the numerous histograms demonstrates that about 84% are typical histograms that have a Gamma-like shape with heavy tails.

Mahmood *et al.* [26] proposed an analytical model to predict the average time that a packet spends in an SDN network. In this work, data plane is modeled as open Jackson network and the controller is modeled as an M/M/1 queue. Lin *et al.* [27] proposed an analytical model based on stochastic network calculus and verified results on OMNET+ simulations and laboratory testbench. Their model gave the lower bounds for the switch-controller end-to-end delay in a SDN network. Yang *et al.* [28] proposed a cross stratum optimization (CSO) architecture for data center services that require low delay, high availability and guaranteed end-to-end QoS. Their SDN solution used OpenFlow-based elastic optical nodes that provided global optimization and control across data center network resources to meet QoS demand. Yang *et al.* [29] adapted SDN to provide control over multiple resources for joint optimization of end-to-end services in 5G. Based on the proposed architecture, a resource provision scheme (RIP) using an auxiliary graph is used to schedule routes globally. They experimentally verified the architecture for multi-dimensional resources integration for cloud radio-over-fiber network (C-RoFN) on an OpenFlow-based enhanced SDN testbed.

III. Experimental Setups

We performed experiments on three different platforms, i.e., Mininet [30], the GENI[31] and the OF@TEIN tested [32]. All three platforms use switches based on OVS.

We conducted end-to-end delay measurements using ping command for three packet sizes, *small* (40 bytes), *medium* (576 bytes) and *large* ($1,500$ bytes). Three packet forwarding scenarios were considered in the experiments, i.e., (1) *proactive*, (2) *all-to-controller* and (3) *reactive*. In the proactive case, controllers populate the flows tables in network switches before the start of communication between sender and receiver. In the all-to-controller case, every packet arriving at any switch is forwarded to the controller to take the decision and after that a flow entry is sent back to the switch. The same is repeated at every switch in the path from sender to receiver. In the reactive case, only a fraction of packets require controller intervention due to periodic timeout and removal of flow entries from switch flow tables. Packets were sent at a rate of 10 packets/sec and time out was set to 2 seconds for all experiments. Reported measurements of each experiment are the average of $10,000$ values.

A. Mininet

Mininet [30] is an emulator for quick prototyping of large SDN networks on a single computer. It lets users launch a virtual network with switches, hosts and an SDN controller that enable the rapid development and testing of SDN applications. We used Mininet 2.2.1, running Open vSwitch v2.5.0, on a PC with an Ubuntu Linux operating system (OS), Intel Core 2 Duo 3.0GHz processor and 2GBs of RAM. A linear topology with four switches as shown in Figure 2 was used. Three scenarios were considered for experiments:

- **Proactive:** When the controller populates the switchs flow table ahead of time.
- **Reactive:** When a switch does not find a flow table entry for a fraction of incoming round trip flows and consults the controller.
- **All-to-controller:** When all packets are forwarded through controller.

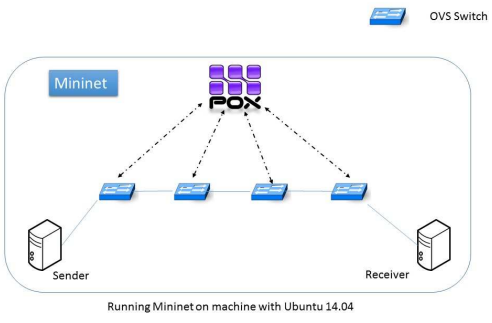


Fig. 2: Mininet experiment setup.

B. GENI Testbed

The Global Environment for Networking Innovation (GENI) [31] is a large scale tested that provides a virtual environment

for running network experiments on Internet-scale and permits a great degree of flexibility. GENI compute and network resources can be obtained from a great number of nodes / sites, principally distributed across the United States. In GENI we can make custom topologies by connecting different resources using Layer-2 connectivity. GENI allows us to install operating systems on VMs using pre-configured images or even install custom operating systems and software on allocated resources. It also gives users control over the operation of network switches to control the flow of traffic. GENI also gives us the ability to run Layer-3 and above protocols by installing necessary software in the resources allocated to the experimenter. GENI has its own instrumentation and measurement tools that can be used to monitor resource functionality and take measurements more effectively.

We categorized the experiments we conducted on GENI testbed into two types:

Type 1 All OpenFlow switches were reserved on a single GENI site and all four controllers were put on another separate GENI site apart from the switches' site. This configuration simulates a setting in which all switches are part of the same autonomous system.

Type 2 In this scenario all four controllers were put on individual sites while OpenFlow switch sites remained unchanged. This configuration decorrelates controller-switch communication delay and simulates a setting in which all four switches belong to different autonomous systems.

1) Type 1 All Controllers on Same Site: In this case we created a linear topology of four switches on GENI. These four switches were reserved on the *KENTUCKY PKS2* site of GENI which is hosted by the University of Kentucky, Lexington, KY. Figure 3 shows the experimental setup for the first type of experiment. We reserved four VMs running POX controllers [33] on the *CENIC InstaGENI* site located in Los Angeles, CA.

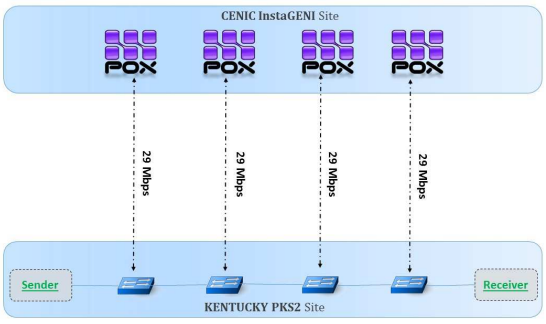


Fig. 3: GENI Case 1 Experiment Setup

Every switch was connected to one of the controllers as depicted in Figure 3. Any packet received by a switch for which it did not have a matching flow table entry (flow table entry miss) is forwarded to its assigned controller. Every switch was running Open vSwitch v2.5.0. We measured round-trip times (RTT) using ping command at a rate of 10 packets/sec and took $10,000$ measurements for each considered packet size. These measurements were taken for three packet sizes of 40, 576 and $1,500$ bytes.

2) Type 2 All Controllers on Different Site: In this case switch locations were kept unchanged, i.e., all switches were on the **KENTUCKY PKS2** site. In this case four controllers were running on the *UCLA*, *Illinois*, *Ohio* and *CENIC* GENI sites. Like before, all controllers ran the POX controller. The reason behind putting controllers on different sites was to decorrelate the controller switch communication delays, as they might be for four switches belonging to four separate autonomous systems. As before, we used Open vSwitch v2.5.0 and measured RTTs by taking $10,000$ measurements for each packet size of 40, 576 and $1,500$ bytes.

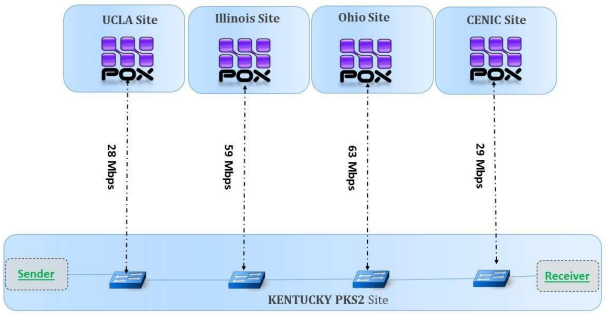


Fig. 4: GENI Case 2 experiment setup

C. OF@TEIN Testbed

The OpenFlow @ Trans-Eurasian Information Network (OF@TEIN) [32] is a testbed that provides a virtual playground for doing SDN related experiments. This testbed has been developed by the Networked Computing Systems (NetCS) Laboratory at the Gwangju Institute of Science and Technology (GIST), South Korea, and is still evolving. It comprises of ten international sites spread over nine different countries, including Pakistan, South Korea, Malaysia, Vietnam, India, Philippine, Indonesia, Thailand and Taiwan. Figure 5 depicts the physical infrastructure underlying the OF@TEIN testbed. All sites in OF@TEIN are accessed and managed through OpenStack [34] Horizon. The experimenter can reserve resources at any of the ten sites. The testbed uses an OpenDaylight [35] controller running at GIST to install flows in flow tables of network switches.

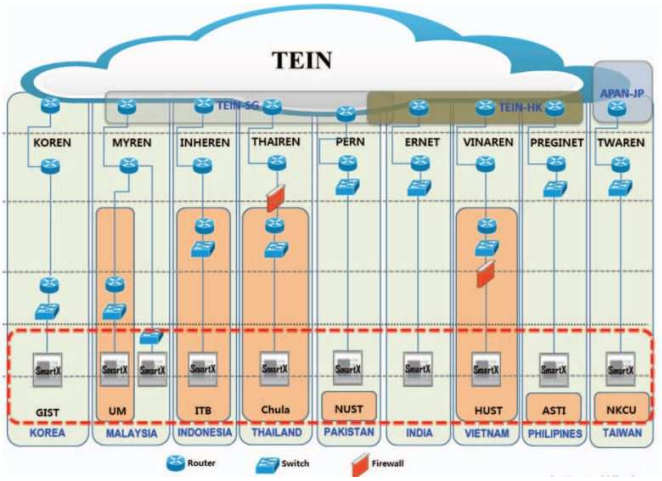


Fig. 5: OF@TEIN testbed physical infrastructure.

Figure 6 is the experimental setup used to take measurements for end-to-end delay on the OF@TEIN testbed. One VM was reserved at each of the two sites, the Philippines and Malaysia. The VM at the PH site in the Philippines was configured as a sender and the VM at the Malaysia site, hosted by MYREN, as a receiver. A POX controller was running at GIST, Gwangju, South Korea. The network topology consisted of four switches. Two switches were at the MYREN site and other two switches at the PH site and both sites were connected through GRE tunnel. We measured the RTT to avoid clock synchronization issues present in measuring the one-way delay.

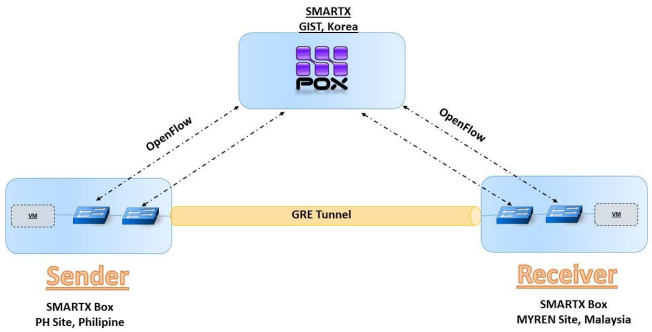


Fig. 6: OF@TEIN experiment setup.

IV. Results

A. Goodness-of-fit Criteria

Two widely used criteria for information based model selection in computational learning theory are the Akaike information criterion (AIC) and the Bayesian information criterion (BIC). The goal of these models is to make predictions about models based on given data that best generalize the underlying distribution of data.

1) Akaike Information Criterion: In 1979, Akaike introduced the AIC that is considered among the first model selection criteria. It is one of the most widely used model selection criteria among practitioners. In traditional statistical modeling, the maximum likelihood estimate (MLE) is used to predict the unknown parameters of a model having a specific dimension and structure. Akaike built AIC on the MLE to provide a mechanism to predict the parameters of the model having unknown dimension and where the dimension is found from the data. Using the Akaike framework, model estimation and selection can be accomplished simultaneously.

The main idea behind AIC is that if we have a true distribution P of data and we want to compare it with two other models M_1 and M_2, then the better one will be the one that has lower Kullback-Leibler divergence (KLD) with true distribution P. But in most real scenarios we do not have the true distribution from which the data points were drawn, in which case we estimate the $P-M_1$ and $P-M_2$ from the given data. Then the model with the lowest AIC value is preferred. Equation 1 is the mathematical expression to calculate AIC.

$$AIC = 2K - 2\log(L), \tag{1}$$

where,

K is the number of predictors
L is the maximum likelihood value

In Equation 1 the term $2K$ is the penalty for adding extra predictors. The other term, $-2\log(L)$, tells how closely the model fits the data.

2) Bayesian Information Criterion: The BIC is another well known criterion used in statistical modeling to check which model best fits available data. BIC is considered a standard approach for model selection for large data sets. BIC penalizes models that have a large number of free parameters. The best model is the one that has the lowest BIC value.

Consider a random sample $Y_1, \ldots, Y_n$, and two competing models $f_1(y, \Theta_1, \ldots, \Theta_{m1})$ and $f_2(y, \phi_1, \ldots, \phi_{m2})$ that we want to fit to the data. If $L_1(\theta_1, \ldots, \theta_{m1})$ and $L_2(\phi_1, \ldots, \phi_{m2})$ are likelihood functions, then the BIC will be defined as follows:

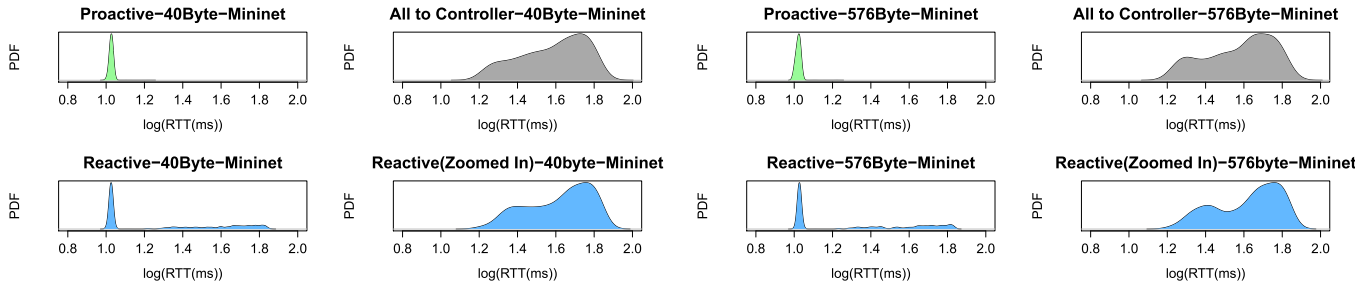
$$BIC = 2\ln \frac{L_1(\theta_1, \ldots, \theta_{m1})}{L_2(\phi_1, \ldots, \phi_{m2})} - (m_1 - m_2). \tag{2}$$

3) PDF Estimation Using AIC and BIC: To determine which distribution best fits the data we used the fitdistrplus package [36] for R. We use this package to estimate parameters for different distributions from our data. After we estimate parameters for all considered distributions, we can compare the AIC or BIC to determine which distribution fits the data better. The distribution with the smallest value of AIC and BIC is the one that best generalizes the underlying data distribution. We estimated PDF for three packet sizes, 40, 576 and $1,500$ bytes, for all three platform, i.e., Mininet, GENI and OF@TEIN.

We have estimated PDFs using AIC and BIC for four cases as follows:

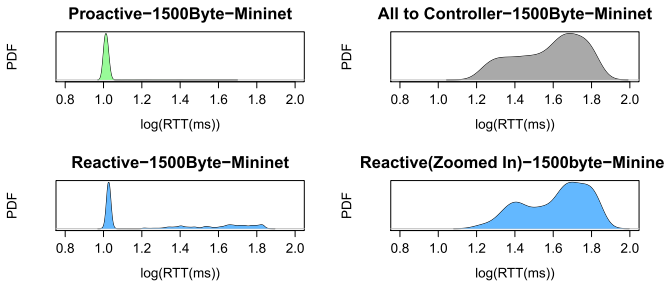
- **Proactive Delay PDF:** This is the PDF for the case when all packets find hits in flow tables of all switches they pass through.
- **All-to-controller Delay PDF:** This is the case when all packets generate misses in flow tables of all switches they pass through.
- **Reactive Delay PDF:** The PDF for this case is a Gaussian mixture model with the following two modes:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2016.2636247, IEEE Access



(a) Mininet - PDF for 40 bytes

(b) Mininet - PDF for 576 bytes



(c) Mininet - PDF for $1,500$ bytes

Fig. 7: PDFs for end-to-end delay in Mininet.

– **All hits:** This is the mode of the PDF due to delays of packets that find hits in flow tables of all switches and do not require controller intervention.
– **One-or-more Misses:** This mode of the PDF is due to delays of packets that encounter a flow table miss in at least one switch.

B. Mininet

We set a hard time-out of 2 secs for flow table entries in Mininet OVS switches. Figure 7 plots the PDFs for all three packet sizes.

Table I shows the AIC and BIC values for all three considered packet sizes. For the proactive case, AIC and BIC are lowest for log-normal[1] distributions for all three packets sizes, i.e., the end-to-end delay in an OpenFlow network follows a log-normal distribution. For the all-to-controller case, AIC and BIC are lowest for the Weibull distribution, which suggests that for all three packets sizes the end-to-end delay follows a Weibull distribution. For the all hits mode of the reactive case, AIC and BIC are lowest for the log-normal distribution for all three packets sizes, which suggests that the end-to-end delay in this case follows a log-normal distribution. For the all-or-more mode of the reactive case, AIC and BIC are lowest for the Weibull distribution, which suggests that for all three packets sizes the end-to-end delay in this case follows a Weibull distribution. Figure 10 shows the empirical and estimated PDF plots for all cases in Mininet.

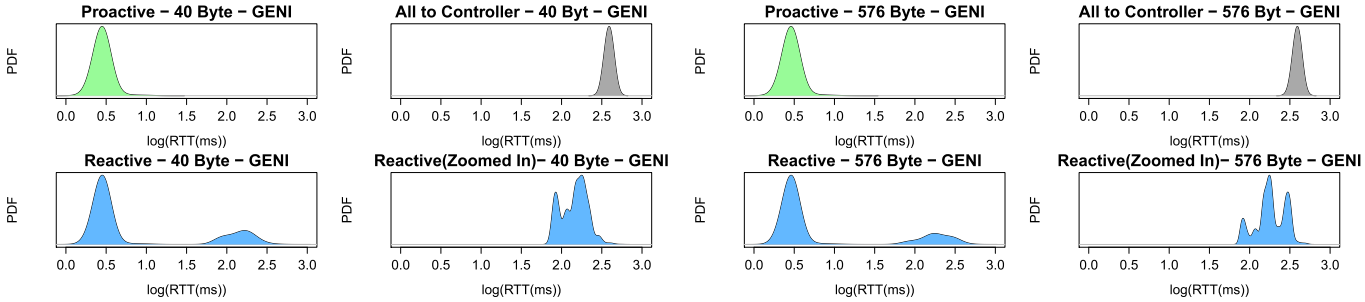
[1]When the logarithm of a random variable is distributed according to a Gaussian it follows the log-normal distribution

C. GENI Testbed

Like before, the hard time-out for flow table entries was set to 2 secs. After that, PDFs were plotted for every considered packet size, as shown in the Figure 8. Table II shows the AIC and BIC values for all three considered packet sizes. For the proactive case, AIC and BIC are lowest for Normal distributions for all three packets sizes, meaning that the end-to-end delay in an OpenFlow network follows a log-normal distribution. For the all-to-controller case, AIC and BIC are lowest for the log-normal distribution, which suggests that for all three packets sizes the end-to-end delay follows a Weibull distribution. For the all hits mode of the reactive case, AIC and BIC are lowest for the log-normal distribution for all three packets sizes, which suggests that the end-to-end delay in this case follows a log-normal distribution. For the all-or-more mode of the reactive case, AIC and BIC are lowest for the log-normal distribution, which suggests that for all three packets sizes the end-to-end delay in this case follows a Weibull distribution. Figure 11 shows the empirical and estimated PDF plots for all cases in GENI.

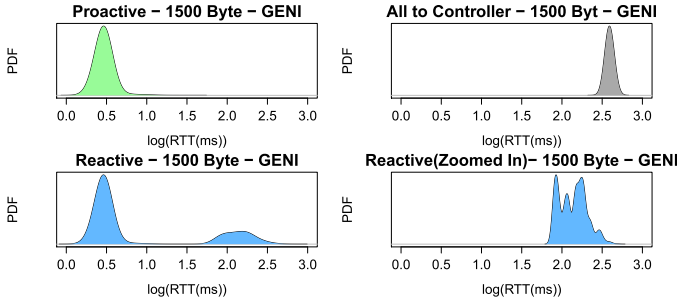
D. OF@TEIN Testbed

The hard time-out for flow table entries was set to 2 secs. After that, PDFs were plotted for every considered packet size, as shown in the Figure 8. Table III shows the AIC and BIC values for all three considered packet sizes. For the proactive case, AIC and BIC are lowest for Normal distributions for all three packets sizes, meaning that the end-to-end delay in an OpenFlow network follows a log-normal distribution. For the all-to-controller case, AIC and BIC are lowest for the log-normal distribution, which suggests that for all three packets



(a) GENI - PDF for 40 bytes

(b) GENI - PDF for 576 bytes



(c) GENI - PDF for $1,500$ bytes

Fig. 8: PDFs for end-to-end delay in GENI.

TABLE I: PDF estimation for Mininet.

Comparison based on AIC												
Mininet	**40 Byte**				**576 Byte**				**1500 Byte**			
	Weibull	Gamma	Lognormal	Normal	Weibull	Gamma	Lognormal	Normal	Weibull	Gamma	Lognormal	Normal
Proactive	20721.28	1966.508	1603.499	2838.487	21726.28	1980.123	1713.678	2878.543	20721.28	1966.508	1603.499	2838.487
All to Controller	16338.03	16425.39	16521.57	16421.56	16315.45	16476.32	16621.53	16525.33	16425.65	16420.89	16516.72	16413.81
Reactive[Switch]	370206.2	2802.091	1543.293	5525.703	415560.4	1705.901	1341.1055	4630.861	20721.28	1966.508	1603.499	2838.487
Reactive[Controller]	3416.519	3451.025	3471.003	3461.125	3402.248	3444.611	3467.717	3428.952	3424.185	3464.891	3464.891	3459.23
Comparison based on BIC												
Proactive	20735.71	1980.929	1617.920	2852.908	20875.71	1995.999	1705.135	2879.873	20735.71	1980.929	1617.920	2852.908
All to Controller	16349.23	16436.59	16532.77	16432.76	16367.32	16464.43	16665.34	16425.65	16343.16	16432.09	16527.92	16425.01
Reactive[Switch]	370220.6	2816.426	1557.628	5540.038	415574.8	1720.236	1355.440	4645.196	20735.71	1980.929	1617.920	2852.908
Reactive[Controller]	3424.594	3434.594	3479.079	3450.295	3410.314	3420.314	3475.783	3437.018	3432.275	3472.981	3491.17	3467.32

TABLE II: PDF estimation for GENI.

Comparison based on AIC												
GENI	**40 Byte**				**576 Byte**				**1500 Byte**			
	Weibull	Gamma	Lognormal	Normal	Weibull	Gamma	Lognormal	Normal	Weibull	Gamma	Lognormal	Normal
Proactive	24525.48	15341.07	13436.53	20784.43	23829.50	14881.91	13213.93	19775.45	26297.86	17535.03	15367.85	23385.74
All to Controller	19119.62	19027.04	19046.91	19003.08	19389.42	19272.5	19287.9	19258.38	19130.63	18998.83	19015.05	18981.94
Reactive[Switch]	19028.43	12357.49	10874.11	16597.41	20213.50	12893.10	11249.25	17564.29	18415.34	11944.55	10650.63	15639.81
Reactive[Controller]	26994.64	26712.90	26684.41	27070.26	25429.37	25450.92	25395.83	25553.48	27866.57	27482.20	27347.89	28068.74
Comparison based on BIC												
Proactive	24539.90	15355.49	13450.95	20798.85	23843.92	14896.33	13228.35	19652.71	26312.28	17549.45	15382.27	23400.16
All to Controller	19130.82	19038.24	19058.11	19014.28	19400.62	19283.7	19299.1	19269.59	19141.83	19010.03	19026.25	18993.15
Reactive[Switch]	19042.28	12371.34	10887.96	16611.26	20227.42	12907.02	11263.18	17578.21	18429.17	11958.38	10664.46	15653.64
Reactive[Controller]	27006.27	26724.53	26696.04	27081.89	25440.77	25462.32	25407.23	25564.87	27878.25	27493.88	27359.57	28080.42

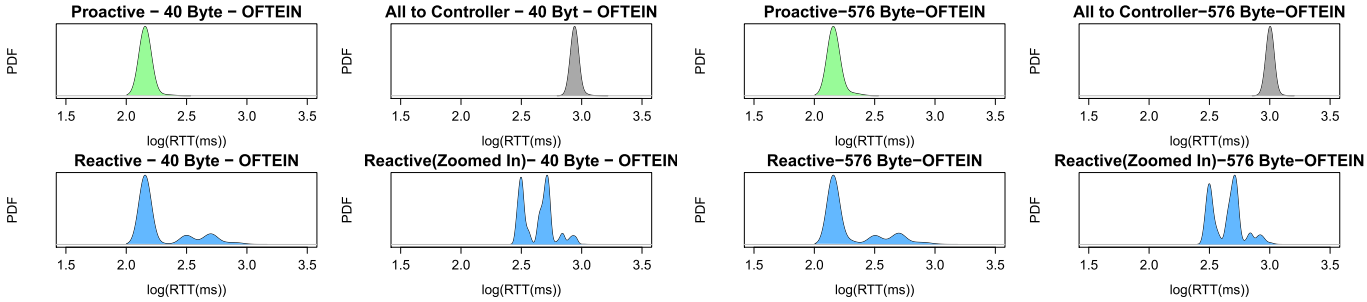
sizes the end-to-end delay follows a Normal distribution. For the all hits mode of the reactive case, AIC and BIC are lowest for the log-normal distribution for all three packets sizes, which suggests that the end-to-end delay in this case follows a log-normal distribution. For the all-or-more mode of the reactive case, AIC and BIC are lowest for the log-normal distribution, which suggests that for all three packets sizes the end-to-end delay in this case follows a log-normal distribution. Figure 12 shows the empirical and estimated PDF plots for all cases in OF@TEIN.

V. Stochastic Modeling

We model the end-to-end delay in an OpenFlow SDN, as the sum of two components: Deterministic delay (D_D) and stochastic delay (D_S), i.e.,

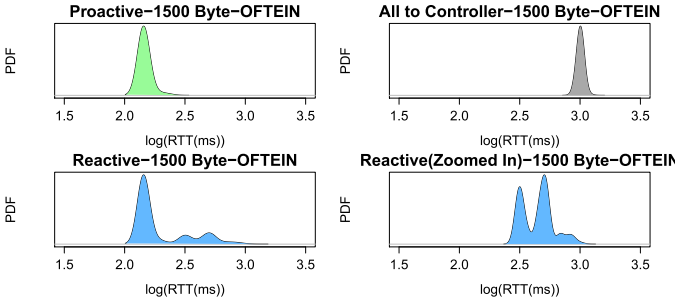
$$D_{E2E} = D_D + D_S. \tag{3}$$

These two terms are further decomposed in terms of the



(a) OFTEIN - PDF for 40 bytes

(b) OFTEIN - PDF for 576 bytes



(c) OFTEIN - PDF for $1,500$ bytes

Fig. 9: PDFs for end-to-end delay in OF@TEIN.

TABLE III: PDF estimation for OFTEIN.

OFTEIN	40 Byte				576 Byte				1500 Byte			
	Weibull	Gamma	Lognormal	Normal	Weibull	Gamma	Lognormal	Normal	Weibull	Gamma	Lognormal	Normal
Comparison based on AIC												
Proactive	52598.93	40778.92	40216.81	41924.08	36246.37	31064.05	30718.17	31768.93	46363.99	38522.75	38021.60	39440.25
All to Controller	18914.32	18828.94	18842.16	18801.01	19662.66	18925.57	18945.92	18886.99	15464.08	14790.32	14782.24	14507.56
Reactive[Switch]	41617.03	32741.79	32293.84	33650.09	38130.23	33544.40	33163.25	34320.94	43439.00	35106.31	34791.09	35744.27
Reactive[Controller]	26470.65	26124.00	26041.68	26456.13	22922.38	22663.10	22615.78	22938.80	50292.76	49683.99	49514.51	50630.28
Comparison based on BIC												
Proactive	52612.70	40792.70	40230.59	41937.86	36259.26	31076.95	30731.06	31781.82	46377.49	38536.24	38035.09	39453.74
All to Controller	18925.46	18840.09	18853.30	18810.15	19673.83	18936.74	18957.09	18898.16	15474.86	14801.10	14793.01	14518.33
Reactive[Switch]	41630.32	32755.09	32307.14	33663.39	38143.14	33557.31	33176.16	34333.85	43452.48	35119.79	34804.57	35757.75
Reactive[Controller]	26481.92	26135.26	26052.94	26467.39	22933.35	22674.06	22626.74	22949.76	50305.18	49696.41	49526.94	50642.70

following equation:

$$D_{E2E} = \sum_{i=1}^{l} (D_{trans,i} + D_{prop,i}) + \left(\sum_{i=1}^{n_S} S_{s,i} + \sum_{j=1}^{n_C} I_j \times S_{c,j} \right) \quad (4)$$

In Equation 4, $D_{prop,i}$ is the propagation delays of the i^{th} link on the path between sender and receiver, each of which is calculated as $D_{prop,i} = \frac{Distance_i}{Speed_i}$.

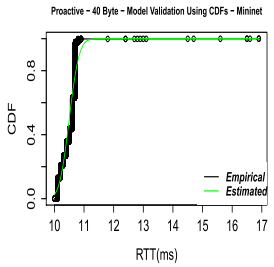
$D_{trans,i}$ is the transmission delay over the i^{th} link between sender and receiver, $D_{trans,i} = \frac{\text{Number of bits}}{\text{Link transmission rate}}$. This way, the deterministic delay D in Equation 3 is the sum of transmission delays $D_{trans,i}$ and propagation delays $D_{prop,i}$ of links on the path.

The terms $S_{s,i}$ and $S_{c,j}$ in Equation 4 are the stochastic delays associated with the i^{th} switch and j^{th} controller, respectively. Here, n_S denotes the total number of switches on the path under consideration and n_C is the number of controllers that may be queried by switches to forward a packet to its destination. I_j are Bernoulli random variables that take on value 1 with probability α_j and value 0 with probability $(1 - \alpha_j)$, also called an *indicator function*. The values of α_j depend on a variety of factors including the timeout value of flow table entries in switches and input traffic rate. The stochastic delay S in Equation 3 is the sum of all switch delays $S_{s,i}$ and all controller delays $S_{c,i}$,

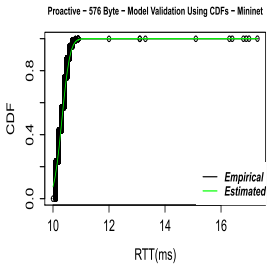
Due to the stochastic nature of the end-to-end delay (D_{E2E}) we measured and modeled its PDF. We found the PDF of transit latency in OpenFlow switch SDNs to be multi-modal, which is a departure from the unimodal distributions of transit latencies found in traditional networks with distributed control planes.

PDFs of the measurements taken from the Mininet emulator show a log-normal + Weibull mixture distribution for the end-to-end delay in SDNs. The GENI and OF@TEIN testbeds, which are overlayed on production / research networks, show that the PDFs of end-to-end delay in SDNs is a log-normal mixture. Based on the fact that these two testbeds also contain the actual traffic (Mininet contains measurements of only emulated traffic), we concluded that log-normal mixture model will be more accurate representation for end-to-end delay in SDNs. When PDFs are plotted on log-log scales we can see that the PDFs exhibit multiple modes. We model these modes

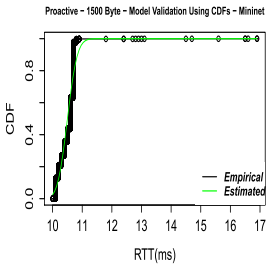
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2016.2636247, IEEE Access



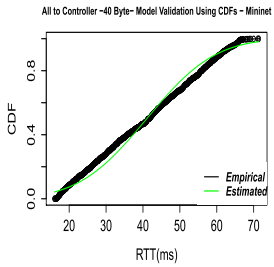
(a) Proactive-40 bytes, $R^2 = 0.97$



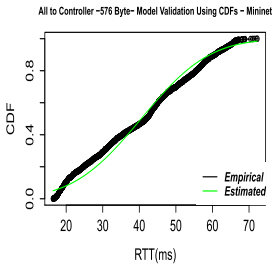
(b) Proactive-576 bytes, $R^2 = 0.97$



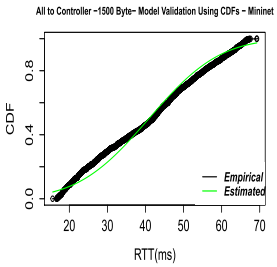
(c) Proactive-$1,500$ bytes, $R^2 = 0.97$



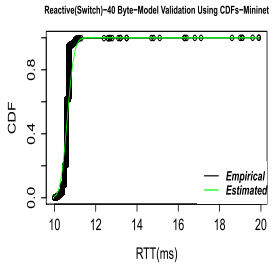
(d) All-to-controller-40 bytes, $R^2 = 0.98$



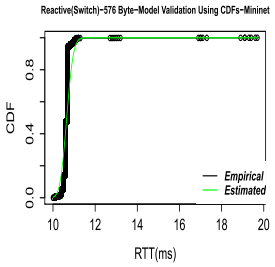
(e) All-to-controller-576 bytes, $R^2 = 0.98$



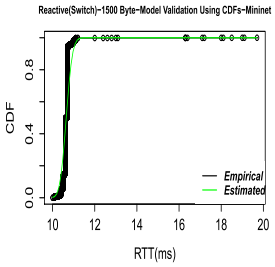
(f) All-to-controller-$1,500$ bytes, $R^2 = 0.98$



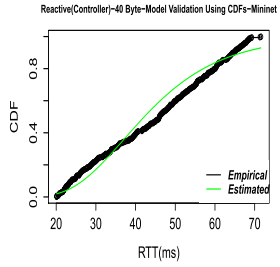
(g) Reactive (all hits)-40 bytes, $R^2 = 0.97$



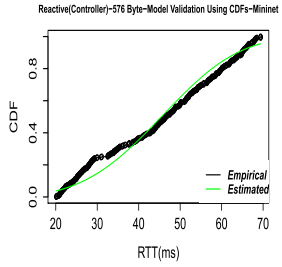
(h) Reactive (all hits)-576 bytes, $R^2 = 0.97$



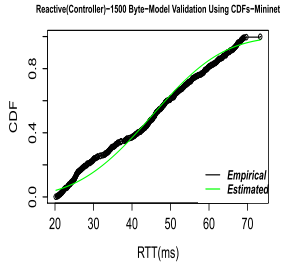
(i) Reactive (all hits)-$1,500$ bytes, $R^2 = 0.97$



(j) Reactive (one-or-more misses)-40 bytes, $R^2 = 0.93$



(k) Reactive (one-or-more misses)-576 bytes, $R^2 = 0.95$



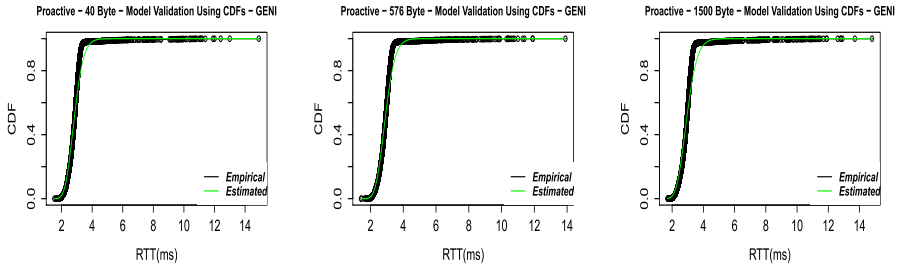
(l) Reactive (one-or-more misses)-$1,500$ bytes, $R^2 = 0.95$

Fig. 10: Mininet - Estimated vs. empirical PDFs comparison.

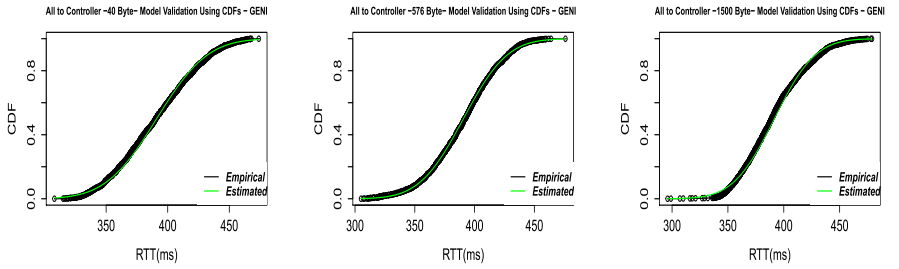
as log-normal distributions. We model the PDF as a log-normal mixture model (LNMM) of random variable X in Equation 5:

$$f_X(x) = \sum_{i=1}^{K} \lambda_i \mathcal{N}(\log x; \mu_i, \sigma_i^2)$$
$$\text{where, } 0 \leq \lambda_i \leq 1 \text{ and } \sum_{i=1}^{K} \lambda_i = 1, \tag{5}$$

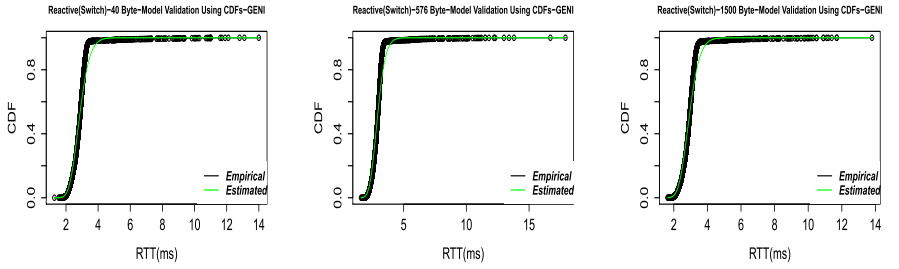
and $\mathcal{N}(x; \mu_i, \sigma_i^2)$ for $1 \leq i \leq K$ are K log-normal PDFs



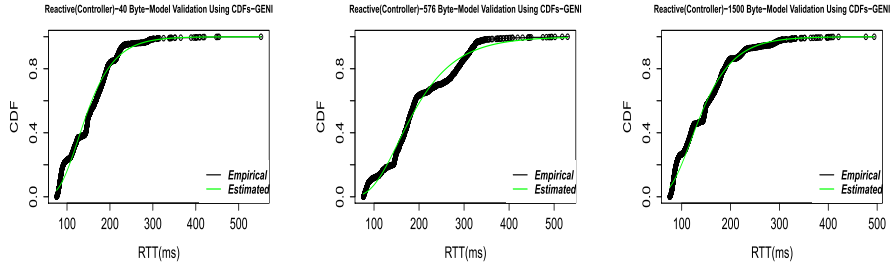
(a) Proactive-40 bytes, $R^2 = 0.99$ (b) Proactive-576 bytes, $R^2 = 0.99$ (c) Proactive-$1,500$ bytes, $R^2 = 0.99$



(d) All-to-controller-40 bytes, $R^2 = 0.99$ (e) All-to-controller-576 bytes, $R^2 = 0.99$ (f) All-to-controller-$1,500$ bytes, $R^2 = 0.99$



(g) Reactive (all hits)-40 bytes, $R^2 = 0.99$ (h) Reactive (all hits)-576 bytes, $R^2 = 0.99$ (i) Reactive (all hits)-$1,500$ bytes, $R^2 = 0.99$



(j) Reactive (one-or-more misses)-40 bytes, $R^2 = 0.98$ (k) Reactive (one-or-more misses)-576 bytes, $R^2 = 0.97$ (l) Reactive (one-or-more misses)-$1,500$ bytes, $R^2 = 0.98$

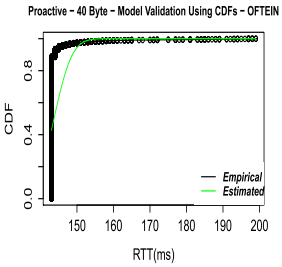
Fig. 11: GENI - Estimated vs. empirical PDFs comparison.

of the form,

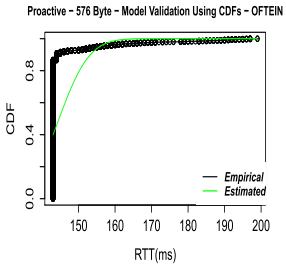
$$\mathcal{N}(\log x; \mu, \sigma^2) = \frac{1}{x\sigma\sqrt{2\pi}} e^{\left(-\frac{1}{2}\left(\frac{\log x - \mu}{\sigma}\right)^2\right)}. \quad (6)$$

In our case we set $K = 2$ to produce a bi-modal PDF of the LNMM. One mode represents the transit delay when the controller is consulted by the switch in making a forwarding decision and the other one for the case when the switch finds a matching flow table entry. The parameters λ_1 and $\lambda_2 = 1 - \lambda_1$ are weights that determine the probability with which a packet is forwarded proactively (λ_1, without controller involvement) or reactively (λ_2, with controller involvement).

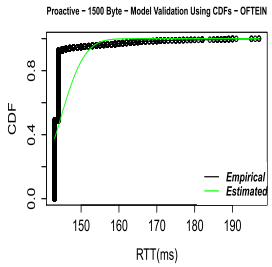
$$Mean = \mu e^{\frac{\sigma^2}{2}} \quad (7)$$



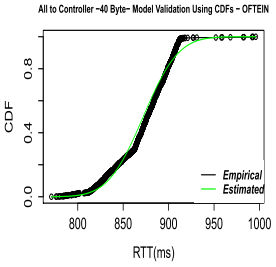
(a) Proactive-40 bytes, $R^2 = 0.93$



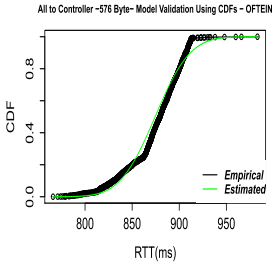
(b) Proactive-576 bytes, $R^2 = 0.90$



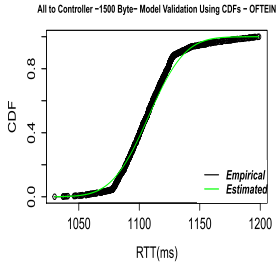
(c) Proactive-$1,500$ bytes, $R^2 = 0.90$



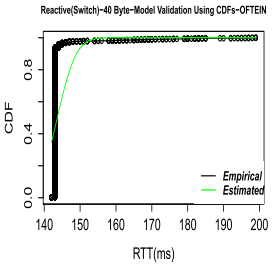
(d) All-to-controller-40 bytes, $R^2 = 0.95$



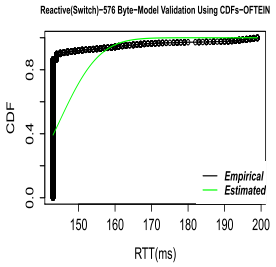
(e) All-to-controller-576 bytes, $R^2 = 0.95$



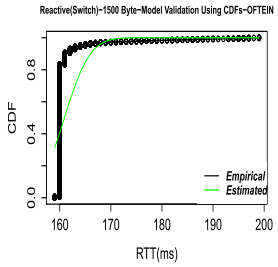
(f) All-to-controller-$1,500$ bytes, $R^2 = 0.95$



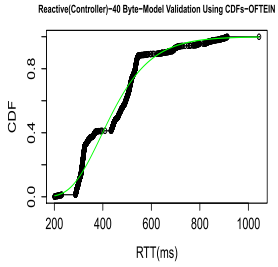
(g) Reactive (all hits)-40 bytes, $R^2 = 0.93$



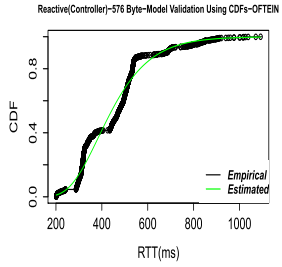
(h) Reactive(all hits)-576 bytes, $R^2 = 0.90$



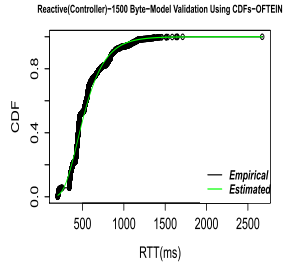
(i) Reactive(all hits)-$1,500$ bytes, $R^2 = 0.90$



(j) Reactive (one or more misses)-40 bytes, $R^2 = 0.94$



(k) Reactive (one or more misses)-576 bytes, $R^2 = 0.94$



(l) Reactive (one or more misses)-$1,500$ bytes, $R^2 = 0.98$

Fig. 12: OF@TEIN - Estimated vs. empirical PDFs comparison.

$$Variance = (e^{\sigma^2} - 1)e^{2\mu+\sigma^2} \tag{8}$$

$$\overline{x} = \exp\left(\frac{1}{n}\sum_{i=1}^{n}\log(x_i)\right), \tag{9}$$

The MLEs of the mean and variance parameters of a log-normal distribution using n samples x_i, where $1 \leq i \leq n$, are calculated as:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\log\frac{x_i}{\mu}\right)^2}. \tag{10}$$

To validate the proposed stochastic model, we compared its cumulative distribution functions (CDFs) against that of the empirical data for all platforms under considerations. We quantify the degree of similarity between model and data CDF using the coefficient of determination, denoted R^2, and defined in Equation 11,

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(x_i - f_X(x_i)\right)^2}{\sum_{i=1}^{n} \left(x_i - \overline{x}\right)^2}, \tag{11}$$

where $\overline{x}$ is the sample mean of the n samples.

VI. Conclusion

In this paper, we presented a stochastic model for end-to-end delay for networks with OVS-based SDN switches derived from empirical measurements. We performed experiments on four OpenFlow SDN switches on three different platforms (i.e., Mininet emulator, and the GENI and OF@TEIN testbeds) together with POX controllers to study the end-to-end delay characteristics in OpenFlow-enabled networks. We proposed a log-normal mixture model for end-to-end delay in SDN and validated it with our experimental data and found out it be a good fit to the empirical measurements. Previous studies proposed models for end-to-end delay across SDN switches that were rooted in queuing theory and were M/M/1 models. Our results show that an M/G/1 model with a log-normal mixture model will model end-to-end delay in OpenFlow-enabled networks more accurately.

Acknowledgment

The authors would like to thank Aris Cahyadi Risdianto at GIST, Gwangju, South Korea for his technical assistance for issues related to the OF@TEIN testbed. We would like to acknowledge the travel grants that enabled us to attend the OF@TEIN collaborative training workshops in Thailand (2013) and Vietnam (2014) and the first author's summer internship at the Networked Computing System Laboratory, also provided by GIST, South Korea.

References

[1] T. Benson, A. Akella, and D. A. Maltz, "Unraveling the complexity of network management." in *NSDI*, 2009, pp. 335–348.

[2] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

[3] M. Pham and D. B. Hoang, "Sdn applications: The intent-based northbound interface realisation for extended applications," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. IEEE, 2016, pp. 372–377.

[4] "ONF, Open Networking Foundation,," https://www.opennetworking.org/, 2014.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[6] K. He, J. Khalid, A. Gember-Jacobson, S. Das, C. Prakash, A. Akella, L. E. Li, and M. Thottan, "Measuring control plane latency in sdn-enabled switches," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. ACM, 2015, p. 25.

[7] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an openflow architecture," in *Proceedings of the 23rd international teletraffic congress*. International Teletraffic Congress, 2011, pp. 1–7.

[8] A. Chilwan, K. Mahmood, O. N. Østerbø, and M. Jarschel, "On modeling controller-switch interaction in openflow based sdns," *International Journal of Computer Networks & Communications*, vol. 6, no. 6, p. 135, 2014.

[9] T.-C. Yen and C.-S. Su, "An sdn-based cloud computing architecture and its mathematical model," in *Information Science, Electronics and Electrical Engineering (ISEEE), 2014 International Conference on*, vol. 3. IEEE, 2014, pp. 1728–1731.

[10] R. Sherwood and Y. KOK-KIONG, "Cbench: an open-flow controller benchmarker," 2010.

[11] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic," in *ACM SIGCOMM Computer Communication Review*, vol. 23, no. 4. ACM, 1993, pp. 183–193.

[12] F. Ciucu and J. Schmitt, "Perspectives on network calculus: no free lunch, but still good value," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. ACM, 2012, pp. 311–322.

[13] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "Openflow switching: Data plane performance," in *Communications (ICC), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1–5.

[14] D. Y. Huang, K. Yocum, and A. C. Snoeren, "High-fidelity switch models for software-defined network emulation," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 43–48.

[15] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker, "Extending networking into the virtualization layer." in *Hotnets*, 2009.

[16] D. Sunnen, "Performance evaluation of openflow switches." 2011.

[17] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: state distribution trade-offs in software defined networks," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 1–6.

[18] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 7–12.

[19] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore, "Oflops: An open framework for openflow switch evaluation," in *Passive and Active Measurement*. Springer, 2012, pp. 85–95.

[20] S. Azodolmolky, R. Nejabati, M. Pazouki, P. Wieder, R. Yahyapour, and D. Simeonidou, "An analytical model for software defined networking: A network calculus-based approach," in *Global Communications Conference (GLOBECOM), 2013 IEEE*. IEEE, 2013, pp. 1397–1402.

[21] Z. Bozakov and A. Rizk, "Taming sdn controllers in heterogeneous hardware environments," in *Software Defined Networks (EWSDN), 2013 Second European Workshop on*. IEEE, 2013, pp. 50–55.

[22] M. Samavati, "An analytical and performance study of random topologies in an openflow network," 2013.

[23] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic," in *ACM SIGCOMM Computer Communication Review*, vol. 23, no. 4. ACM, 1993, pp. 183–193.

[24] F. Ciucu and J. Schmitt, "Perspectives on network calculus: no free lunch, but still good value," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. ACM, 2012, pp. 311–322.

[25] C. Bovy, H. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, and P. Van Mieghem, "Analysis of end-to-end delay measurements in internet," in *Proc. of the Passive and Active Measurement Workshop-PAM*, vol. 2002, 2002.

[26] K. Mahmood, A. Chilwan, O. Østerbø, and M. Jarschel, "Modelling of openflow-based software-defined networks: the multiple node case," *IET Networks*, vol. 4, no. 5, pp. 278–284, 2015.

[27] C. Lin, C. Wu, M. Huang, Z. Wen, and Q. Zheng, "Performance evaluation for sdn deployment: an approach based on stochastic network calculus," *China Communications*, vol. 13, no. Supplement, pp. 98–106, 2016.

[28] H. Yang, J. Zhang, Y. Zhao, Y. Ji, J. Han, Y. Lin, and Y. Lee, "Cso: cross stratum optimization for optical as a service," *IEEE Communications Magazine*, vol. 53, no. 8, pp. 130–139, 2015.

[29] H. Yang, J. Zhang, Y. Ji, Y. He, and Y. Lee, "Experimental demonstration of multi-dimensional resources integration for service provisioning in cloud radio over fiber network," *Scientific Reports*, vol. 6, 2016.

[30] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.

[31] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "GENI: A federated testbed for innovative

network experiments," *Elsevier Computer Networks*, vol. 61, pp. 5–23, 2014.
[32] A. C. Risdianto and J. Kim, "Prototyping media distribution experiments over of@ tein sdn-enabled testbed," *Proceedings of the Asia-Pacific Advanced Network*, vol. 38, pp. 12–18, 2014.
[33] J. Mccauley, "Pox: A python-based openflow controller," 2014.
[34] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, 2012.
[35] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *2014 IEEE 15th International Symposium on*. IEEE, 2014, pp. 1–6.
[36] M. L. Delignette-Muller and C. Dutang, "fitdistrplus: An r package for fitting distributions," 2014.

Azeem Iqbal received the B.S. degree from University of Management and Technology (UMT), Pakistan and M.S. degree from National University of Sciences and Technology (NUST), Pakistan, in 2013 and 2016, respectively. He is also a member of Applied Network and Data Science Research (AN-DASH) Group at SEECS, NUST. His research interests include software defined networking, data analytics and cloud computing.

Uzzam Javed received the B.S. degree from the National University of Computer and Emerging Sciences (FAST-NUCES) Islamabad, Pakistan and M.S. degree from the National University of Sciences and Technology (NUST), Pakistan, in 2013 and 2016, respectively. He is also a member of the Applied Network and Data Science Research (AN-DASH) Group at SEECS, NUST. His research interests includes software defined networking, data science and cloud computing.

Saad Saleh received the B.S. and M.S. degrees from the National University of Sciences and Technology (NUST), Pakistan, in 2010 and 2012, respectively. He is currently Team Lead at Applied Network and Data Science Research (AN-DASH) Group at SEECS, NUST. His research interests includes machine learning, social networks, software defined networking and cloud computing.

JongWon Kim received the B.S., M.S. and Ph.D. degrees from Seoul National University (Seoul, Korea), in 1987, 1989 and 1994, respectively, all in Control and Instrumentation Engineering. In 1994 2001, he was a faculty member of KongJu National University (KongJu, Korea) and University of Southern California (Los Angeles, USA). From September 2001, he has joined Gwangju Institute of Science and Technology (Gwangju, Korea), where he is now a full Professor. Since April 2008, he has been leading GIST SCENT (Super Computing and Collaboration Environment Technology) Center as a director. He is also leading Networked Computing Systems Lab. (recently renamed from Networked Media Lab.) which focuses on Dynamic and Resource-aware Composition of Media-centric Services employing Programmable/Virtualized Computing/Networking Resources. His recent research interests cover topics such as software defined networking (SDN)/cloud computing for Future Internet testbed and smart media-centric services employing heterogeneous SmartX nodes.

Jalal S. Alowibdi is currently appointed Assistant Professor of Computer Science at the University of Jeddah, Saudi Arabia.

Muhammad Usman Ilyas is currently appointed Assistant Professor of Computer Science at the University of Jeddah, Saudi Arabia. Prior to this he was Assistant Professor of Electrical Engineering at the School of Electrical Engineering and Computer Science (SEECS) of the National University of Sciences and Technology (NUST), Islamabad, Pakistan. Prior to that he was a Post-doctoral Research Associate appointed jointly by the Electrical and Computer Engineering (ECE)department and the Computer Science and Engineering (CSE) department at Michigan State University (MSU). He worked under the joint supervision of Dr. Hayder Radha (IEEE Fellow) and Dr. Alex Liu at East Lansing, MI. Dr. Ilyas received his Ph.D. and MS degrees in Electrical Engineering from Michigan State University in 2009 and 2007, an MS in Computer Engineering from the Lahore University of Management Sciences (LUMS) at Lahore, Pakistan in 2004, and a BE (Honors) in Electrical Engineering from NUST, Pakistan in 1999. He is interested in network science, future Internet architectures and social networks.