Validating the Accuracy of Analytical Modelling in Software Defined Networks

Jordan Ansell Bryan Ng* Winston K.G. Seah*
School of Engineering and Computer Science, Victoria University of Wellington
Wellington, New Zealand
*{bryan.ng,winston.seah}@ecs.vuw.ac.nz

***Abstract*—Analytical modelling and experimental measurements are both useful techniques for evaluating the performance of networks. Models provide insight while measurement provides realism. Mathematical analysis can provide valuable insights for the management and configuration of a network in order to optimise network performance. Queueing analysis has been used extensively to study the performance of telecommunications and computer networks. However, software-defined networking (SDN) has introduced greater variability within the network environment. For software defined networks, it is unknown how well the existing queueing models represent the performance of a real SDN network. This leads to uncertainty between what can be predicted and the actual behaviour of a software defined network. This work investigates the accuracy of software defined network queueing models. This is done through comparing the performance results of analytical models to experimental performance results. The outcome of this is a better understanding of how reliable the existing queueing models are for analysing SDN performance and areas where the queueing models can be improved.**

***Index Terms*—Analytical performance modelling, queueing theory, software defined networks, experimental measurements.**

I. Introduction

In a communication network, network devices such as switches and routers forward data based on a set of rules defined by protocols. In traditional networking, a switch/router has tightly coupled control and data planes [1]. The control plane performs control functions such as routing protocols and middlebox configuration while the data plane forwards packets based on decisions made by the control plane. This makes the control function hardware-dependent in traditional networks resulting in network configuration and management that is complex and inflexible. To simplify the configuration complexity that is prevalent in traditional networks, a new networking paradigm called Software-Defined Networking (SDN) has emerged, which is envisioned to be a key enabling technology for 5G networks.

SDN simplifies the configuration complexity in networks by physically separating the decision making (control plane) and packet forwarding (data plane). Consequently, there is greater variability with the network environment, making SDN networks behave differently from traditional networks. Packet forwarding in traditional networks is well defined and deeply seated in hardware, with established standards on how packets are structured and the protocols for handling them. This facilitates performance analysis using mathematical tools and models which help designers assess a network's performance before it is actually implemented, giving the designer the flexibility to adjust various network parameters during the planning phase [2]. SDN networks are flexible in the logic used to arrive at forwarding decisions. A network administrator can decide exactly how the network devices (i.e. switches) treat packets and have these decisions change in response to network and traffic behaviour. The decisions are made in the SDN controller and communicated to the switches using a standardized protocol, such as OpenFlow [3]. This greater variability implies less certainty in the way that the network will perform and encourages the need for understanding the system's behaviour and what affects the resulting performance.

In this paper, we investigate the accuracy of the analytical queueing models that have been developed to date by comparing the numerical results obtained from these models to the results obtained from measurements of typical commercial off the shelf SDN devices. Section II discusses related work on performance evaluation of SDN, focusing on cross-validation of analytical models using measurements and experimentation.

II. Related Work

A unique feature of SDN networks as compared to traditional networks is the additional control traffic generated by a switch when it receives a packet for which it cannot find a matching flow table entry. This triggers a *packet_in* event which sends a message containing the packet information to the controller to handle. The controller processes this message and responds to the switch with a *packet_out* message and also *flow_mod* messages to install new flow rules in relevant switches. Depending on the granularity of flows, the frequency of flow table misses will change. When granularity is fine (e.g. based on ports) there will be more controller path traffic than with coarse granularity (e.g. based on a range of IP addresses). The proportion of new flow rules or probability of there not being an existing flow rule has been considered in the existing models at the finest of granularity, but can be used at any level of granularity. Jarschel *et al.* [4] and Mahmood *et al.* [5] call this P_{nf}, the probability of $\boldsymbol{n}$o $\boldsymbol{f}$low entry existing.

The performance analysis of an SDN network has been done in three different ways: Experimental testbeds, Software-based tools, and Mathematical models [6]. A hundred papers were surveyed which focused on SDN using the following keywords in Google Scholar, ACM Digital Library, IEEE Xplore, and

	Identify	ProbDist	MaxRate	Obtainable	PrepTime	Online information
XenaCompact	○	×	○	○	○	xenanetworks.com/test-chassis/xenacompact-chassis-solution
STG-10G	○	○	○	×	○	www.ecdata.com/stg-10g.html
NetFPGA	E	E	○	○	×	netfpga.org
Nping	×	×	×	○	○	nmap.org/nping/
Scapy	○	E	×	○	○	scapy.net
D-ITG	○	○	×	○	○	www.grid.unina.it/software/ITG/
DPDK	E	E	○	○	×	www.dpdk.org/
PF_Ring	E	E	○	○	○	www.ntop.org/products/packet-capture/pf_ring/

○ : Passes requirement × : Fails requirement E : extra work required

TABLE I: Comparison of Traffic Generators

	Reliable	Format	Obtainable	PrepTime	Online information
TCPDump	×	○	○	○	www.tcpdump.org
PF_Ring	○	E	○	○	www.ntop.org/products/packet-capture/pf_ring/
DAG Card	○	○	×	○	www.endace.com/endace-high-speed-packet-capture-solutions/oem/dag/

○ : Passes requirement × : Fails requirement E : extra work required

TABLE II: Comparison of Data Capture Tools

SpringerLink: experimental testbeds, simulation, emulation, measurement tools, analytical models, performance analysis. The survey revealed that 75% used software-based tools, while experimental testbeds and mathematical modelling made up 12% and 13% respectively.

The two basic mathematical methodologies used to model an SDN-based network are queueing theory and network calculus. Queueing theory aims to model the performance of a system in average quantities at equilibrium state, while the Network Calculus shows the performance of a system in a probabilistic bound curve with the worst case scenario. In this paper, we focus on the queueing theory approach. The first analytical modelling of an SDN system was by Jarschel *et al.* [4]. After which, there has been a few other efforts to analytically model an SDN network, e.g. [5], [7], [8], etc. However, there has not been any explicit effort to validate these analytical model against real systems. Conversely, stochastic models have been developed for specific performance measure, e.g. end-to-end delay as a whole [9].

III. Measurement Approach

The measurement environment, tools and methods used in this study are critical aspects that determine the accuracy and validity of the results. Careful considerations were undertaken to select the appropriate tools to accomplish three critical tasks, viz. generate packets, send packets and record packet data. The requirements that need to be met include: (1) **Identify** each packet uniquely when sent and received; (2) **Probability Distribution (ProbDist)** can be specified for the packet interarrival time variable, e.g. Exponential Distribution; (3) **Maximum Rate (MaxRate)** of sending packets up to the link's capacity; (4) **Reliable** measurements can be made; (5) **Obtainable** hardware (measurement devices) at reasonable to low cost; (6) **Preparation Time (PrepTime)** needed to set up the measurement experiments is not excessive; and (7) **Format** of the recorded data is of an easily accessible form. Note that requirement (4) needs to account for any effect that the measuring equipment has on the measurements made, such as, additional delay of a host's network stack and network interface card when sending or receiving Ethernet frames.

A. Packet Generators

Tools for traffic (packet) generation can be divided into two types: hardware and software. The hardware category contains specialised devices which create and release packets directly into the network cables. The software category comprises programs that run on a computer's operating system, and generate packets via the hardware available in the host machine.

In general, the results obtained from a proprietary hardware traffic generator will have accurate measurement results. The drawbacks are generally a high cost and lack of flexibility. Software tools are more flexible but constrained by the platforms they are executed on. We focus only on tools that are available online as open-source software or are free with open APIs, viz. Nping, Scapy, Distributed Internet Traffic Generator (D-ITG), Data Plane Development Kit (DPDK) and PF_Ring. Table I summarizes the packet generation candidates under the different requirements. As requirements (4) and (7) are related to the measurement process, not the generation process, they are therefore not include in Table I. PF_Ring is able to meet all the requirements, with extra effort to implement the missing features.

B. Delay Measurement Tools

There are two important components to obtaining measurement results. One is the act of capturing the packet data and the other is recording the times when a packet is sent and received. The tools and approaches evaluated contribute to the Requirements (4) and (7).

Table II summarizes how the three tools are able to meet the requirements. In order to make time measurements across multiple hosts, the clocks on these host must be kept synchronised. Existing networking protocols exist, including Network Time Protocol (NTP) and IEEE 1588 Precision Time Protocol (PTP) [10]. The latter was chosen as it uses hardware timestamps to estimate the delay between hosts and provide sub-microsecond synchronisation.

C. Measurement Network Setup

The key elements of the measurement network, as shown in Figure 1, include an Allied Telesis AT-510x SDN switch (OF Switch), two Dell Optiplex 7040 hosts (Host A and Host B) and an Intel NUC as the SDN Controller. The Access Switch connects Host A, Host B and the SDN Controller to a host (Manager Host) that manages and automates the measurement process. All the measurements were done using the following two methods: *End-to-end Delay Method* and *Network Events Method*.

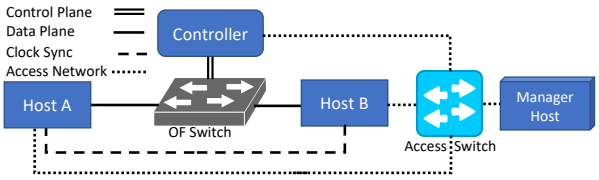


Fig. 1: Layout of measurement testbed

1) End-to-end Delay Method: Two hosts have their clocks synchronised and the times of packets leaving the sending host and arriving at the receiving host are recorded by each respective host. The difference between these times is then calculated as the delay between the hosts.

The difference between the delay when the hosts are directly connected and delay where there is a network device in between can be compared to calculate the delay across just the network device. This removes the processing delay of the source and sink hosts from the device delay measurements.

2) Network Events Method: Only one host's clock is used. Packets are intercepted in the network and copies are sent to the timekeeping host. The difference between the arrival time of the network events is used to calculate the time taken across sections of the traffic's path in the network.

Using hardware timestamps, measurements of this delay were taken, and the delay is consistently equal to $922ns$. The hardware timestamps have a resolution of $8ns$, meaning this delay is between $918ns$ and $926ns$. The only other measured delay of exactly $1000ns$ was shown in 1.3% of the results.

The delay calculated from these results is a simple difference. The difference between original and cloned packet are reliably consistent. The times reported by the arrival timestamp of network events at the sink can be used to calculate the difference between two packets. Thus the delay of a network device traversed between two consecutive network events will be equal to the difference in time between the arrival of the cloned packets at the sink.

IV. Service Times Measurements

Data on the traffic patterns of packets arriving at a network node can be collected using various measurement techniques and tools to determine the arrival rate λ. The service rate μ represents the processing rate of the network node's hardware and system software, and to align the queueing model's predicted performance results with the real world, an appropriate (if not accurate) estimation of the service rate needs to be determined. Often, a network node is very much a black box with various internal components and processes that contribute to the service rate or time. In the SDN/OpenFlow system, there are several different service times to be considered:

1) **Switch:** Receiving a data packet on the data plane, which then leaves the switch on the data plane
2) **Switch:** Receiving a data packet on the data plane, which then goes to the controller as a $packet\_in$ message
3) **Switch:** Receiving a $packet\_out$ message from the controller, which then sends a data packet out the switch on the data plane
4) **Controller:** Receiving a $packet\_in$ message and respond with $packet\_out$

In existing SDN analytical modelling work (cf: Section II), only the *controller's service rate* and the *switch's data-to-data plane service rate* are considered.

A. Component of Network Delay

Each device, along a packet's path from origin to destination, experiences transmission, processing, and queueing delays. Propagation delay is experienced between each device as the signal passes through the network cable. The *processing delay* of the network devices is the value which is intended to be measured. The *queueing delay* is what queueing models determine when calculating average performance.

Propagation times along a few meters length of Ethernet cable and below are about five nanoseconds, thus negligible. Queueing delay can be removed by having only a single packet in the system at a time during measurements. Transmission time, which is the time it takes for the packet to be sent by the network hardware, is the only time that is not trivial to remove, and will be included depending on the point where packet times are recorded. In order to eliminate the effect of queueing delay packets are sent at a constant rate, with enough inter-packet delay to avoid packets queueing in the hardware.

B. Controller Path Service Times

The controller path service times consist of the three components: (i) controller's service time, (ii) time for data packet to be sent as a $packet\_in$ message, and time for $packet\_out$ message to be sent as a data packet. All three can be measured using the *Network Events Method* and setup shown in Figure 2.

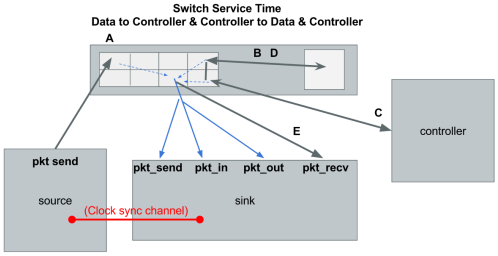
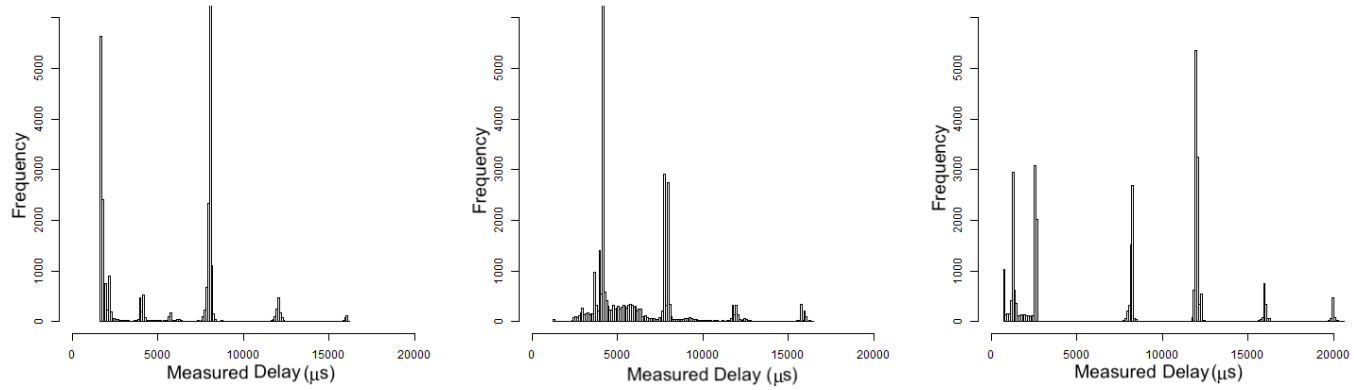


Fig. 2: Measuring service time of OpenFlow switch with messages and controller

Traffic is sent in one direction between the source and sink hosts, viz. Host A and Host B. Upon arriving at the switch,



(a) Data to Controller (b) Controller to Data (c) At Controller

Fig. 3: Distribution of Controller Path Delays

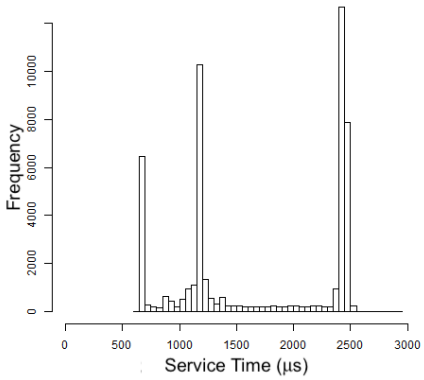


Fig. 4: Controller service time, measured using TCPDump

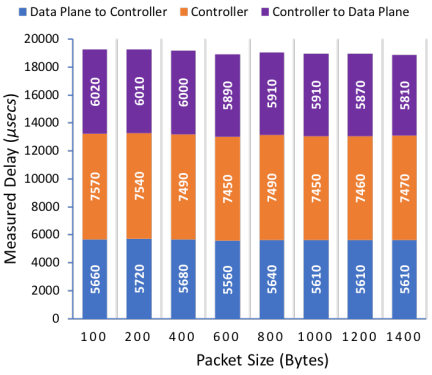


Fig. 5: Mean Delay Measured Over Controller Paths

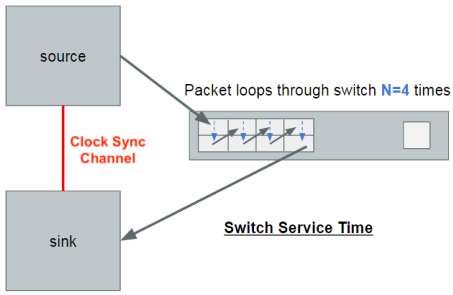


Fig. 6: Setup to loop multiple samples of switch service time

packets are cloned – one is forwarded on to the sink, the other re-enters the switch to be processed and sent to the controller as a $packet\_in$. The link to the controller also passes through the switch, where clones of each packet are sent to the sink. Messages in both directions along this link are thus captured. Once finished at the controller and returned to the switch, the packet is finally received at the sink.

Upon receiving a $packet\_in$ message, the controller application is programmed to install a new rule on the switch which matches the destination link layer address and forwards to the switch's port connected to the sink host. To obtain the service times, the priority of this rule was lower than the rule to send a $packet\_in$ event to the controller enabling all packets to be forwarded to the controller. The rule being installed is never matched and as it is always equivalent to a previously installed rule this is overwritten each time a $flow\_mod$ message arrives.

The sink records the times of network events, and enough information about the packet is captured to uniquely identify it and the event it represents. This information includes the link layer address and transport layer port number. Under the assertion that only one packet should be in the system at a time, a packet rate of 10pps is selected as experiments showed this was slow enough to avoid overlap in consecutive receive and send times when passing through the control plane.

1) Service Time Distributions: Figure 3 presents histograms of the measured service times. These describe the distribution of the delays measured for each of the desired service times. All the service times show a regular, multi-modal distribution, complimented with clusters of data at different points for each different service time. The regular modes appear at roughly four millisecond intervals, beginning at four milliseconds for Figure 3(a) and Figure 3(b), and beginning at eight milliseconds for Figure 3(c). Separating these clusters are areas of very few observed measurements.

The most interesting of these is Figure 3(c), the controller's measured service time. Figure 4 shows the service time of the controller measured at the controller using OS-bound software TCPDump. This describes a trimodal distribution of the controller's service time. When measured in the network at the sink, the same experiments show the multi-modal distribution common among all the service times on the controller path. But the trimodal trend measured on the controller is separate along the x-axis from this common pattern. On closer inspection, when we only focus on the first four millisecond interval of Figure 3(c), we notice that the trimodal distribution

closely (if not exactly) matches that shown in Figure 4.

2) Mean Service Times: Based on the service time distributions, the mean service times are computed and presented in Figure 5. The overall trend is a slight decrease in service time as the packet size increases, which is counter-intuitive. Typically, it is expected that as the packet size increases the delay across a piece of networking equipment will increase.

C. Switch Service Times

The switch (or *data path*) service time consists of only the delay over the switch when forwarding a packet from one port on the forwarding component to another port on the forwarding component.

This is measured in a more novel manner than the controller paths' service times using the *End-to-end method*. The delay measurement contains the result of a varying number of visits to the switch. This allows time for the to-be-forwarded packet to be measured accurately, despite the overhead of the hosts making the measurements. Figure 6 shows the setup for the measurement. The source host sends packets to the switch, one at a time, recording the sending time. The packet passes through the switch N times, before exiting and arriving at the sink where the receive time is recorded. The clocks of source and sink are synchronised via a dedicated and direct link using PTP. This experiment is repeated for several values of N.

As OpenFlow rules are stateless, there is no way to count the number of times a packet has passed through the switch. Instead, the switch is set up with several cables looping back into itself. OpenFlow rules are statically set so a packet arriving on port A will always leave on port B. Each arrival and departure at the switch is a service of the packet. For each N a different set of static rules are programmed to control the number of times a packet will pass through the switch.

The average delay of each number of loops is calculated. The average difference in time between the loops will be equal to the average service time of the switch. The first second of data is discarded to allow the system to start. To combine the results of several iterations, the average of each repetition is calculated, then the mean over the repetitions is calculated. Each sample is filtered to 99.999% of results recorded results to remove outliers.

1) Service Time Distribution: An example distribution of measured switch service times is shown in Figure 7(a) and similar distribution pattern is seen for all measurement samples after removing the extreme outliers. The multi-modal shape gives a clear most common value in each sample, and two other modes are spread below this.

2) Mean Service Time: As expected in experimental measurements, outliers occur. While switch's specifications are provided, it remains essentially a blackbox and when outliers like long processing times are measured, we were unable to determine the real reasons. The outliers affect the average standard deviations of the average mean, median and mode taken from all iterations, as show in Figure 7(b); this is especially evident in the mean delay. We also observed that the first loop has a higher standard deviation than other loops.

Thus, the mean is an unstable average to adopt for representing the results of each sample, shown with the high delay variations. Yet, in terms of the distribution, it is slightly lower than the median and mode as shown in Figure 7(a). In order to minimise the variance of the data used [11], and also include the average of all four loops, the median is selected as the average service time of the switch, as summarized in Figure 7(c).

V. Measured vs Modelled Performance

In this section, we study four analytical queueing models using the service rates that we measured as inputs. To assess the accuracy of these models, we compared the numerical results to experimental results of a real SDN network. We chose two of the earlier analytical models for SDN networks, viz. the M/M/1-S feedback model by Jarschel *et al.* [4] and the Single Switch Jackson networks model by Mahmood *et al.* [5]. We also picked two fundamental queueing models as *base case models*, viz. M/M/1/∞ and M/M/1/K. These four models can be divided into two categories. The first category is models with an infinite capacity and these are the M/M/1 model and Mahmood's model. The second category is models with finite capacity which are the M/M/1/K model and Jarschel's model. It is expected that the finite capacity category will offer a better comparison to the experimental results as real systems have finite and limited buffer capacity.

The difference between the measured and calculated results can be difficult to compare directly. We define a simple measure, *scaled difference*, calculated by equation Equation (1). By dividing the difference by the measured result, the "error" between the modelled results and measured results can be compared. A scaled difference of zero would indicate the model matches the experimental results at that point.

$$\frac{(X_{measured} - X_{modelled})}{X_{measured}} \tag{1}$$

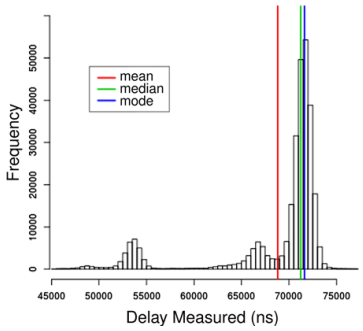
Service time parameters for the models are taken as an average of the results in Figure 5 and Figure 7(c). The service rate, μ parameter, is equal to $\frac{1}{service\_time}$. For finite capacity models the queue capacity parameter is estimated from the measured results to be roughly 200 packets.

A. Using the Base Models

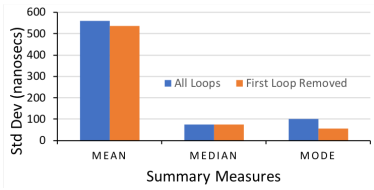
The base models used here have the facility for representing only a single queue and server. To apply the two service processes occurring in a SDN network to the single-queue base models the service time of each path is combined using a P_{nf}-weighted average. This calculation is shown in Section V-A, where the $\mu_{controller\_path}$ is the inverse of the sum of the three average controller path service times.

$$\begin{aligned} \mu_{base\_model} = \; & (1 - P_{nf}) \cdot \mu_{switch\_path} \\ & + P_{nf} \cdot \mu_{controller\_path} \end{aligned} \tag{2}$$

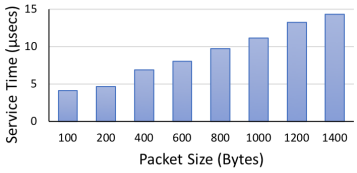
This simple process allows the results of the SDN models and the results of the base queueing models to be compared to the experimental results.



(a) Example Distribution

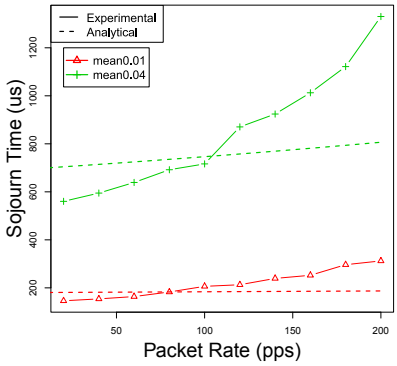


(b) Averaged Standard Deviation

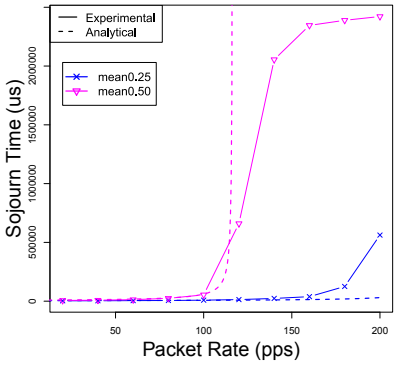


(c) Median Service Times

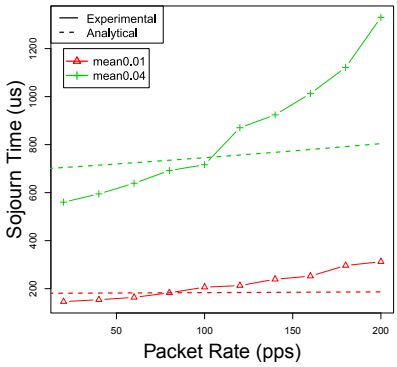
Fig. 7: Measured Service Times



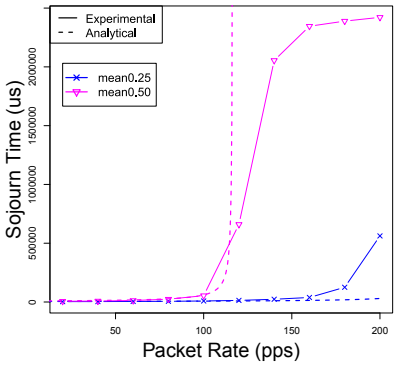
(a) M/M/1 Model P_{nf} = 0.01 and 0.04



(b) M/M/1 Model P_{nf} = 0.25 and 0.50

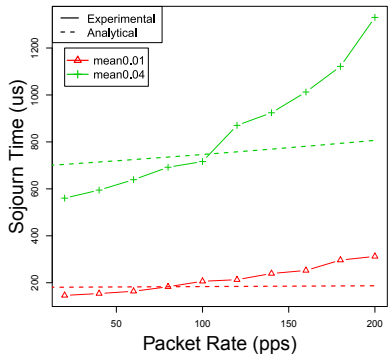


(c) Mahmood's Model P_{nf} = 0.01 and 0.04

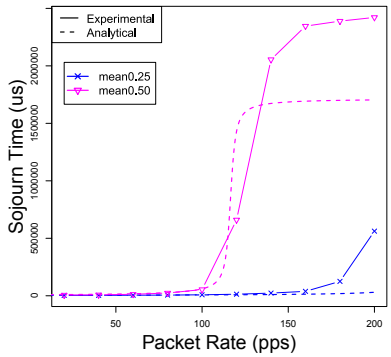


(d) Mahmood's Model P_{nf} = 0.25 and 0.50

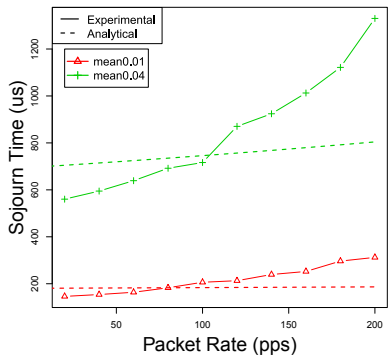
Fig. 8: Comparison of sojourn time for infinite capacity models



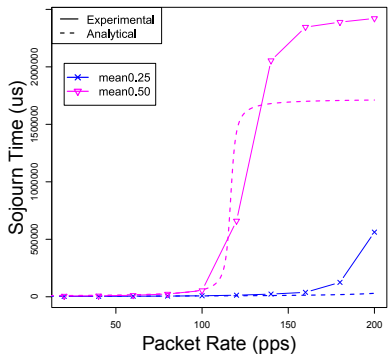
(a) M/M/1/K Model P_{nf} = 0.01 and 0.04



(b) M/M/1/K Model P_{nf} = 0.25 and 0.50

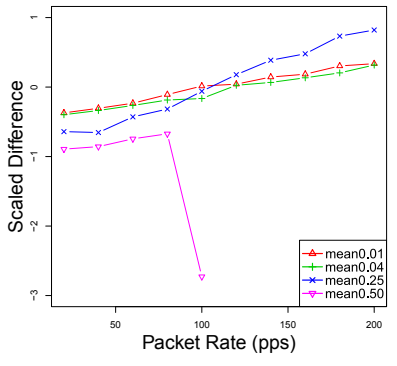


(c) Jarschel's Model P_{nf} = 0.01 and 0.04

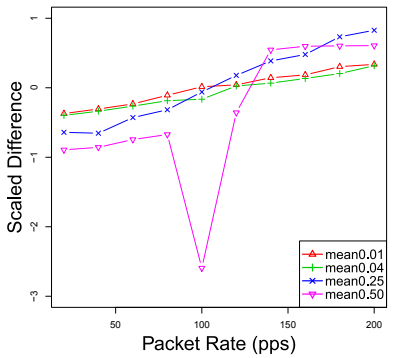


(d) Jarschel's Model P_{nf} = 0.25 and 0.50

Fig. 9: Comparison of sojourn time for finite capacity models



(a) Mahmood's Model



(b) Jarschel's Model

Fig. 10: Scaled difference sojourn results

	M/M/1	M/M/1/K	Jarschel	Mahmood
Sum	778998	775462	14.60791	14.73515
Mean	17311	17232	0.417369	0.4210042

TABLE III: Scaled difference comparison

B. Sojourn Time

The sojourn time of a packet is the time it spends in a system (e.g. switch) from the point it arrives till it leaves, which includes the time spent waiting to be processed. The graphs of $P_{nf} = \{0.01, 0.04\}$ are displayed separately from the graphs of $P_{nf} = \{0.25, 0.50\}$ for clarity due to the differences in magnitude. The sojourn times of the infinite capacity queueing models are shown in Figure 8 and the sojourn times of the finite capacity queueing models are shown in Figure 9. To compare these two models to the experimental results the scaled sojourn differences are shown in Figure 10. The P_{nf} 0.01 and 0.04 results for all models are nearly identical. However, the interesting points are those in P_{nf} 0.25 and 0.50.

C. Arrival and Processing Rates are Equal

As expected of infinite capacity models, the sojourn time increases exponentially leading up to the point where the arrival rate on the controller path is equal to the service rate of the controller path ($\rho = 1$). Similarly, at the same point the sojourn time of the finite capacity models flatten off.

Comparing the behaviour at the point of the models where $\rho = 1$ to the measured results show that the modelled results are prematurely increasing exponentially. Seen in Figure 10 this causes the biggest deviation of the modelled results for both infinite and finite capacity models.

D. General Trend

Looking at the P_{nf} = 0.01 and 0.04, their similarity between both infinite and finite capacity models hints at a key difference between the modelled and experimental results. The modelled results start at a higher sojourn, and increase at a lower rate than the measured results. The measured sojourn results intersect with the modelled results until the point where $\rho = 1$.

This difference in trend is also seen in the steady increase of the scaled difference in Figure 10. The scaling of this comparison method should eliminate the increasing trend of the data, leaving only the trend in differences between them. Note too that this increase in scaled difference is common to all the P_{nf} values. Interestingly, even when the P_{nf} = 0.50 difference increases due to the premature exponential increase it then returns to the same line followed by other P_{nf} values.

E. Scaled Difference

Comparing the mean and sum of the scaled differences, a numerical measure of how well the model represents the measured results can be calculated. The sum and mean of the absolute values of the scaled difference for the sojourn time of four models is presented in Table III. To compare Jarschel and M/M/1/K fairly, the results that are invalid in the infinite category models are also removed from the finite models. As the scaled difference represents the distance from the modelled results, a smaller number is better. Jarschel's model shows the best accuracy using this method having the lowest average and total scaled difference. Visually, however, all the models are very similar. Without better mathematical tools and more data it would be difficult to decide which is the best.

VI. Conclusion

In this work, we aim to quantify and compare the accuracy of existing queueing models with experimental measurements. We selected two of the earlier analytical models for SDN networks, viz. the M/M/1-S feedback model by Jarschel *et al.* [4] and the Single Switch Jackson networks model by Mahmood *et al.* [5], and two fundamental queueing models, the M/M/1 and M/M/1/K, as candidates for our study. Based on the measure of scaled difference, it was found that Jarschel *et at.*'s model is slightly more accurate than the rest at modelling the traffic of the network measured for sojourn time. The accuracy of the packet loss measure is left as future work because the size of the node's buffer need to be determined as accurately as possible. For our work, we used an approximate value estimated from the measurements for use in the finite buffer models.

Acknowledgement

Bryan Ng and Winston K.G. Seah are supported by VUW's Huawei NZ Research Programme, Software-Defined Green Internet of Things (project #E2881).

References

[1] D. Kreutz, F. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[2] M. N. O. Sadiku and S. M. Musa, "Performance measures," in *Performance Analysis of Computer Networks*. Springer International Publishing, 2013, pp. 1–4.

[3] ONF, "OpenFlow Switch Specification," Open Networking Foundation, Tech. Rep., Oct. 2013.

[4] 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 IEEE International Teletraffic Congress (ITC)*, San Francisco, California, 2011, pp. 1–7.

[5] K. Mahmood, A. Chilwan, O. N. Østerbø, and M. Jarschel, "On the Modeling of OpenFlow-Based SDNs: The Single Node Case," in *Proceedings of the Sixth International Conference on Networks & Communications (NeCoM)*, Dubai, UAE, Jul. 2014.

[6] F. Benamrane, M. B. Mamoun, and R. Benaini, "Performances of OpenFlow-Based Software-Defined Networks: An overview," *Journal of Networks*, vol. 10, no. 6, pp. 329–337, 2015.

[7] 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*, IEEE, vol. 3, 2014, pp. 1728–1731.

[8] K. Sood, S. Yu, and Y. Xiang, "Performance analysis of software-defined network switch using $M/Geo/1$ model," *IEEE Communications Letters*, vol. 20, no. 12, pp. 2522–2525, 2016.

[9] A. Iqbal, U. Javed, S. Saleh, J. Kim, J. S. Alowibdi, and M. U. Ilyas, "Analytical Modeling of End-to-End Delay in OpenFlow Based Networks," *IEEE Access*, vol. 5, pp. 6859–6871, 2017.

[10] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–269, Jul. 2008.

[11] G. R. Stavig and J. D. Gibbons, "Comparing the mean and the median as measures of centrality," *International Statistical Review*, vol. 45, no. 1, pp. 63–70, Apr. 1977.