Comparison of SDN OpenFlow Network Simulator and Emulators: EstiNet vs. Mininet

Shie-Yuan Wang
Department of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
shieyuan@cs.nctu.edu.tw

***Abstract*—In this paper, we compare and evaluate the correctness, performance, and scalability of EstiNet OpenFlow simulator, EstiNet OpenFlow emulator, and Mininet OpenFlow emulator over a set of grid networks. The popular Floodlight OpenFlow controller is used without any modification to control the simulated/emulated OpenFlow switches created in these tools. We performed experiments over a set of N x N grid networks, where N = 5, 6, ..., 31 and used the real-world ping program to observe whether the average RTTs reported by these ping packets are correct or not over these tools. We found that in EstiNet simulation, the simulated results of the average RTT are always correct and repeatable, but EstiNet simulator needs more time to finish the simulation when the network size becomes larger. As for emulation, we found that Mininet emulator generated strange results that cannot be explained over some network sizes. In addition, Mininet emulator spends a huge amount of time on its program launch, network setup, and resource releasing when the network size is large. As for EstiNet emulator, we found that it generated good performance and scalability and it used less time to obtain results.**

I. Introduction

In recent years, software-defined networks (SDNs) [1] have come into existence as a new type of network to solve the limitations of legacy networks. The main difference between legacy networks and SDNs is the way they manage the entire network. The former uses a distributed method to manage the network while the latter uses a centralized method to help network administrators more conveniently and flexibly configure the settings of a network. The OpenFlow protocol [2], which has been supported by network providers such as Google, Microsoft, Amazon, and equipment vendors such as NEC, Cisco, is one of the mechanisms that fit into the concept of SDN. An OpenFlow controller sends control messages to OpenFlow switches to instruct them how they should process incoming packets. At present, SDN OpenFlow networks are more and more prevalent because of the benefits of improved network efficiency.

To evaluate the performance of a network efficiently, instead of building a large experimental testbed (e.g., Emulab [3] and PlanetLab [4]), there are two common methods — simulation and emulation, to mimic the environment of an actual network without real deployment. The former approach uses a software program to execute the operations of real devices and the interactions between them. Running a simulation is inexpensive, flexible, controllable, and scalable than using real devices to run real operating systems and applications. Nevertheless, if the model used in the simulator is not correct enough, the results of the simulator will deviate from the results of the actual experiment. To overcome this problem, the latter approach, emulation, uses some real devices running real applications to interact with some simulated devices. Using an emulator is like performing experiments. Therefore, the clock used by the emulator must be the real time clock while the simulation clock can be faster or slower than the real time. Because an emulator does not have its own clock to precisely control the execution order of emulated components, but instead must rely on the kernel's CPU scheduler to schedule the execution order, every time when an emulator generates results, the results may be different due to some activities occurring on the system. In contrast, the results generated by a simulator are always the same because its simulation world is closed without uncontrollable events.

In this study, we used two tools that can simulate or emulate an OpenFlow network. One of them is EstiNet [5], which can be used as a simulator or an emulator, and the other is Mininet emulator [6], [7]. One good property of EstiNet is that it uses the "kernel re-entering" methodology [8], [9] to enable unmodified real application programs to run on simulated hosts. Because of this capability, the simulation results of EstiNet simulator are as accurate as the results obtained from an emulator. Thus, EstiNet simulator not only has many good advantages of simulation, which uses its own simulation clock to control the execution order of simulation events, but also generates very accurate results. In an OpenFlow network simulated by EstiNet, real-life OpenFlow controller programs such as NOX/POX [10], Ryu [11], and Floodlight [12] can directly run on a simulated host to control simulated OpenFlow switches without any modification. As for EstiNet emulator, like any emulator must do, it needs to perform the emulation in real time and can allow the controller application program to run on an external machine to control emulated OpenFlow switches. Due to the use of the kernel re-entering methodology, however, EstiNet emulator can also allow the controller program and emulated OpenFlow switches to run on the same machine.

Mininet emulator is an inexpensive and quickly configurable network testbed. So far, it is the most well-known tool supporting SDN OpenFlow network research, as observed from

the ONS 2013 conference [13]. Mininet uses virtual hosts, switches, and links to create a network on a single OS kernel, and uses the real network stack to process packets and connect to real networks. In addition, Unix/Linux-based network applications are also allowed to run on virtual hosts. In an OpenFlow network emulated by Mininet, a real OpenFlow controller application can run on an external machine or on the same machine where virtual hosts are emulated.

In this paper, we compared the correctness, performance, and scalability of EstiNet simulator, EstiNet emulator, and Mininet emulator. As will be reported in this paper, we found that EstiNet simulator shows its correctness and scalability and EstiNet emulator performs better than Mininet in several performance aspects.

II. Architecture of EstiNet and Mininet

A. EstiNet

"Kernel re-entering (KR) [8], [9]," the unique methodology used by EstiNet, is implemented by using tunnel network interfaces to catch the packets sent down from the IP layer in the Linux kernel and send them to the simulation engine. If an application program running on host 1 sends a packet to another application program running on host 2, the packet will go through the real socket/TCP/IP layers in the Linux kernel and go into a tunnel network interface that connects to the simulation engine. Within the simulation engine, each host has its own simulated protocol stack that contains the Medium Access Control (MAC) layer, the physical layer, and other protocol layers below the IP layer. These layers simulate many effects such as link delay, link bandwidth, link down time, etc. After experiencing these simulated effects on host 1, the packet is sent to the physical layer of host 2 and will be processed by the protocol stack of host 2 in the simulation engine. After that, the packet is sent out from the simulation engine into another tunnel network interface that connects to the IP protocol in the Linux kernel. From that point on, the packet will be processed by the IP/TCP/socket layers of the Linux kernel until it is received by the application program running on host 2.

With the KR methodology, EstiNet not only enables real applications to run on simulated hosts but also uses the real TCP/IP protocol in the Linux kernel to set up TCP connections between them. Note that in a real OpenFlow network each OpenFlow switch needs to establish a TCP connection with a real OpenFlow controller. With the above unique capabilities brought by the KR methodology, in an EstiNet simulated OpenFlow network, each simulated OpenFlow switch can establish a real TCP connection with a real OpenFlow controller and the protocol processing between the OpenFlow controller and an OpenFlow switch is exactly the same as the protocol processing for setting up a TCP connection between two real hosts. More details about the kernel re-entering methodology can be found in [8], [9].

B. Mininet

Mininet [6] creates virtual hosts by using a process-based virtualization method and the network namespace mechanism, which is a feature supported since Linux version 2.2.26, to separate network interfaces, routing tables, and ARP tables of different virtual hosts. The virtual switches in Mininet are a kind of software OpenFlow switches called "Open vSwitch" [14]. The links between virtual hosts and virtual switches are implemented by using virtual Ethernet pairs provided by the Linux kernel. If a packet is sent from one application running on host 1 to another application running on host 2, it will be processed by the real network protocol stack in the Linux kernel. In an OpenFlow network emulated by Mininet, virtual switches need to set up TCP connections to a real OpenFlow controller, which can run on a virtual host or on an external machine. However, because in Mininet the CPU cycles need to be shared by all the virtual hosts, virtual switches, and the controller running on a single OS kernel and the scheduling order of these operations cannot be precisely controlled by the CPU scheduler in the Linux kernel, the results generated by Mininet emulator cannot be repeatable and sometimes may differ from the correct results.

III. Experimental Settings

Figure 1 shows a 5 x 5 grid network among the tested 27 N x N networks, where N is 5, 6, 7, ..., 31. We used this figure to explain the experimental settings when N = 5. These grid networks are used to evaluate three different tools — EstiNet 8.0.4 simulator, EstiNet 8.0.4 emulator, and Mininet 2.0 emulator. Node 1 and node 2 are simulated hosts that link to node 5 and node 9, respectively. Node 5, 6, 7,..., 29 are simulated/emulated OpenFlow switches that support the OpenFlow 1.0 protocol. The Floodlight 0.90 controller application ran on node 3, which is a simulated host. Node 4 simulates a traditional switch, which connects the 25 OpenFlow switches with node 3 to form the control-plane network. Over this control-plane network, these OpenFlow switches will establish their TCP connections with the OpenFlow controller and then exchange messages with the controller. We set the bandwidth and delay of each link to 100 Mbps and 1 ms, respectively.

During a simulation or an emulation, a real-world ping program running on node 1 sent ping packets to node 2 100 times (each separated by 1 second) to measure the Round Trip Time (RTT) between node 1 and node 2. Because the OpenFlow switches need to communicate with the OpenFlow controller to know how to forward the first ping request/reply packets, the first three ping RTT records are higher than the rest 97 RTT records. Therefore, we removed the first three ping RTT records and averaged the rest 97 records of RTT as the average RTT between node 1 and node 2 for this ping test. To get reliable results, we repeated the ping test 10 times and obtained their respective average RTTs. Then, we averaged these ten average RTTs and used this value as the final average RTT between node 1 and node 2. Because EstiNet's simulation results were always correct and repeatable across the 10 ping tests, we did not repeat the ping test 10 times for EstiNet

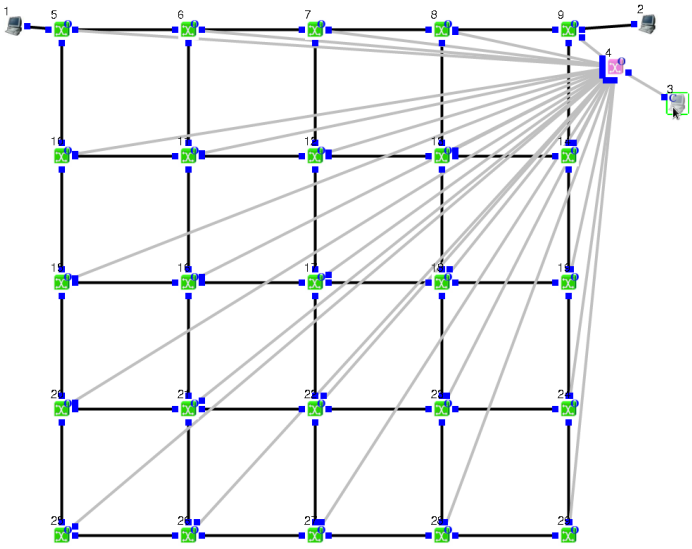


Fig. 1: A N x N grid network, where N = 5.

simulator. However, because an emulator's results cannot be repeatable. We repeated the ping test 10 times for EstiNet emulator and Mininet emulator to evaluate their accuracy and scalability performance.

We used different sizes of the N x N grid network where N is 5, 6, 7, ..., 31 to create networks with different numbers of OpenFlow switches. In these networks, a N x N grid network has N^2 OpenFlow switches in the network. The reason why we did not let N exceed 31 was that Floodlight can only control up to 1,000 OpenFlow switches at most. Therefore, we just created up to 31 * 31 = 961 OpenFlow switches in the network. For EstiNet simulator, because its results are always correct and repeatable, we show the average RTT (ms), main memory consumption (MB), and speed ratio of the elapsed time to the simulated time over these N x N networks. As for EstiNet emulator and Mininet emulator, we showed their average RTT (ms), standard deviation of RTT (ms), main memory consumption (MB), failure rate (%) of ping tests, and total execution time (s) over these N x N networks.

IV. Performance and Scalability Evaluation

First, we present the results of EstiNet simulator and compare them to the theoretical values to evaluate the correctness, scalability, and performance of this tool. Then, we present the results of EstiNet emulator and Mininet emulator and compare their correctness, accuracy, reliability, and scalability.

A. EstiNet Simulator

EstiNet simulator uses its unique methodology, "kernel reentering," to run simulations and emulations. In the simulation mode, the operations of real applications running on any host are precisely controlled by its simulation clock and the measured/reported time by real applications (such as the RTTs reported by ping packets) are all based on its simulation clock. As a result, no matter whether the simulation speed is faster or slower than the real time, the reported RTT times by the ping program are always correct and the same.

In our experiment, we ran the ping program on node 1 and let it send ping packets to node 2. The shortest path between node 1 and node 2 is composed of N OpenFlow switches. Therefore, the number of hops between node 1 and node 2 is proportional to N in a N x N grid network and actually it is (N + 1). For this reason, we used the number of hops that a ping request packet needs to pass as the X-axis variable when showing the average RTT and standard deviation of RTTs of ping packets.

Figure 2 shows the average RTT vs. the number of hops. We take N = 5 as an example. When there are N + 1 = 6 hops between node 1 and node 2, the theoretical value for the average RTT should be 12 ms (i.e., 1 ms link delay * (N + 1) hops * 2 round-trips). From this figure, one sees that this theoretical value is exactly the same as the simulated results for N = 5. Actually, the simulated average RTTs match all theoretic values for all N = 5, 6, 7, ..., 31. Therefore, this figure shows the correctness and scalability of EstiNet simulator.

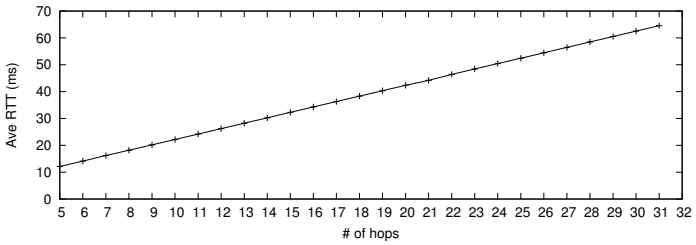
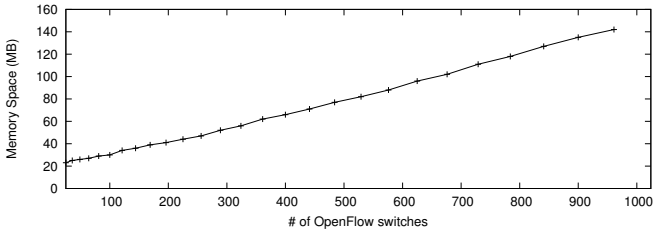


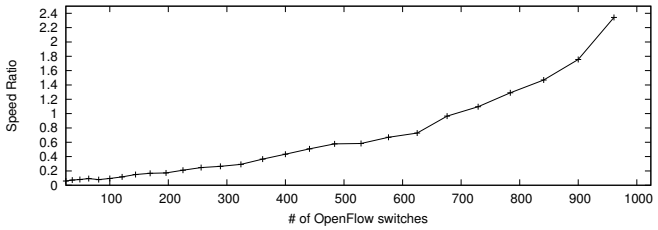
Fig. 2: Average RTT in EstiNet simulator over a N x N grid network, where N = 5, 6, 7,...,31

Figure 3(a) and Figure 3(b) show the main memory consumption and the speed ratio of the elapsed time to the simulated time of EstiNet simulator under different numbers of OpenFlow switches (N^2). The reason why we use the N^2 as the X-axis in these figures is that the number of OpenFlow switches directly affects the main memory space consumption and the simulation speed. In Figure 3(a), we used the "top" command on Linux to get the main memory space used by the EstiNet simulator process. One can see from the figure that the consumption of main memory space increases linearly as the number of OpenFlow switches increases and simulating an OpenFlow switch in EstiNet only consumes 0.15 MB of main memory. This indicates that just 1 GB of main memory is enough to simulate up to 6,666 OpenFlow switches in EstiNet and memory space is not the resource bottleneck of EstiNet.

Figure 3(b) shows the ratio of the elapsed time, which is the real time required to simulate the network, to the simulated time, which is the time that one wants to simulate the network, under different numbers of OpenFlow switches. If the ratio is 2, it means that one has to spend 2 seconds in the real world to simulate the network for 1 second. The required elapsed time is caused by the load of Link Layer Discovery Protocol (LLDP) [15] packets generated and delivered by Floodlight. Floodlight sends LLDP packets to all ports of all OpenFlow



(a) Main Memory Consumption

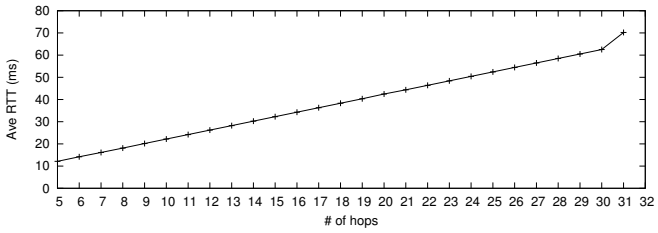


(b) Speed Ratio of the elapsed time to the simulated time

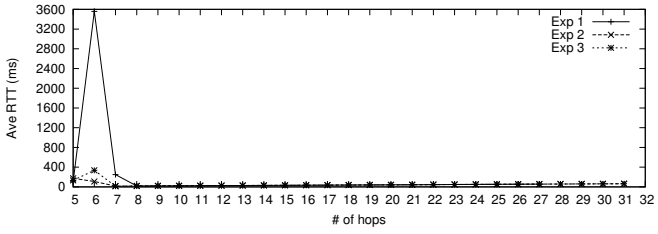
Fig. 3: Main Memory Consumption and Speed Ratio of EstiNet simulator over a N x N grid network, where N = 5, 6, 7,...,31

switches in the network every 15 seconds in order to detect and maintain the newest network topology. Since the port count of a grid network grows with the number of OpenFlow switches (the port count is roughly 4 * # of OpenFlow switches), it is easy to infer that the ratio of the elapsed time to the simulated time will increase (i.e., the simulator will run slower) when the number of OpenFlow switches increases. From Figure 3(b), one sees that although the speed ratio is not linear with the number of OpenFlow switches, EstiNet simulator can still run quite fast to simulate the LLDP packet traffic on the control-plane network.

B. EstiNet Emulator and Mininet Emulator



(a) EstiNet emulator

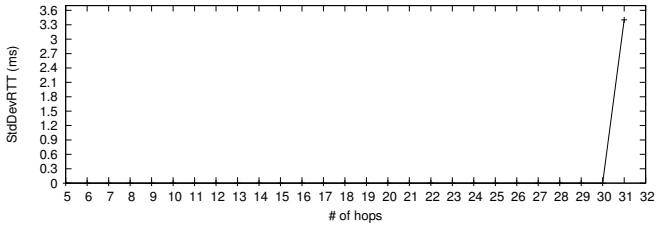


(b) Mininet emulator

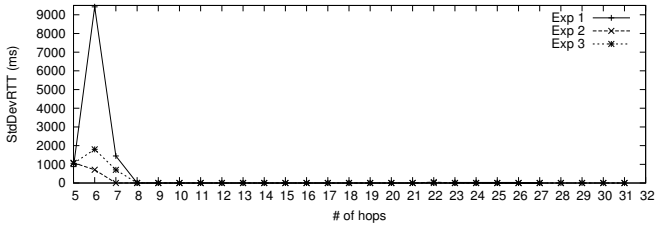
Fig. 4: Average RTT of EstiNet emulator and Mininet emulator over a N x N grid network, where N = 5, 6, ..., 31.

In Figure 4(a), the results generated by EstiNet emulator are the same as the results generated by EstiNet simulator, which are also the theoretical values. The main difference between EstiNet emulator and EstiNet simulator is that EstiNet emulator does not use a simulation clock but instead uses the real clock in the Linux kernel to trigger the operations of network applications and protocol processing and report the RTT values of ping packets. Therefore, if the CPU is not powerful enough to keep the emulation real time, the reported RTT value will become higher than the theoretic value. From Figure 4(a), one sees that EstiNet emulator is very accurate until N is 30. However, the emulated RTT value is a little bit higher than the theoretical value when N is 31.

For Mininet emulator, its average RTT values over different number of hops as shown in Figure 4(b) are hard to explain. In the first set of experiments, the measured average RTTs are 104 ms, 3,558 ms, and 250 ms when N is 5, 6, and 7, respectively. These average RTT results deviate from their theoretic values too much and show that Mininet emulator does not generate correct and reliable results for these settings. Because we cannot explain why Mininet emulator exhibited such a strange behavior, we decided to do the same set of experiments three more times to confirm whether this phenomenon is accidental or not. The three curves shown in Figure 4(b) denoted as Exp1, Exp2, and Exp3 represent the first, second, and third sets of experiments performed over Mininet emulator. One sees that across these three sets of experiments, Mininet emulator always generates incorrect results when N is 5, 6, and 7. Because we are not the developers of Mininet emulator, we do not know how to explain these strange phenomenon. However, we note that when N is not 5, 6, or 7, the results of Mininet emulator are close to the theoretic values and when N = 31 the average RTT value is still close to the theoretic value.



(a) EstiNet emulator



(b) Mininet emulator

Fig. 5: Standard deviation of the average RTTs by EstiNet emulator and Mininet emulator over a N x N grid network, where N = 5, 6, 7, ..., 31.

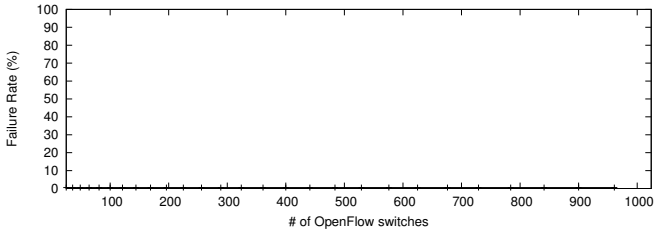
As we mentioned before, because the ping program ran 10 times for each experiment of an emulator, there were totally 10 average RTT records for each tool. Nevertheless, there were 10 average RTT records for EstiNet emulator and 30 average RTT records for Mininet emulator because we did the same experiment three times for Mininet emulator. We used these data to calculate the standard deviation of the average RTTs generated by the two tools. Figure 5(a) shows the standard deviations of average RTTs of EstiNet emulator. These standard deviations are almost equal to zero, meaning that the average RTTs are almost the same across the 10 different ping tests. This shows the consistency and correctness of EstiNet emulator.

Figure 5(b) shows the standard deviations of average RTTs of Mininet emulator. One sees that there are large standard deviations when N is 5, 6 and 7. This means that the average RTTs are spread over a very large range of values across the 30 ping tests and this shows that Mininet emulator does not produce correct and consistent results under these settings. From Figure 4(b) and Figure 5(b), one sees that Mininet emulator does not generate correct and consistent results under some settings. In contrast, the correctness and accuracy of EstiNet emulator are higher than Mininet emulator before N reaches 31.

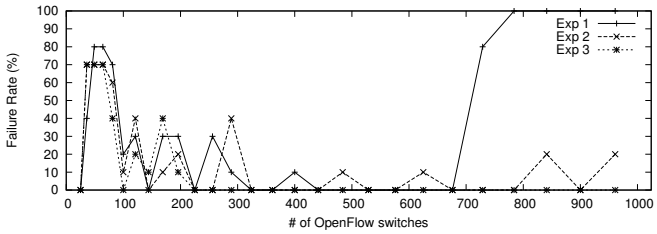
When testing Mininet emulator, we found that the ping packets may get lost due to insufficient CPU cycles to run the emulation in real time. Because an emulator has to execute all emulated OpenFlow switches and network applications in real time, the CPU workload will become heavier and heavier when the number of OpenFlow switches increases. When this situation occurs, the CPU starts to generate packet backlog and the ping packets will start to experience longer delays. If the backlog queue overflows, the ping packets will inevitably need to be dropped by the emulator.

We ran the ping program 10 times for each experiment and the failure might happen among the 10 measurements. The failure rate is defined to be the number of failures divided by 10. We defined that a measurement failed if the first 20 ping packets sent by the ping program, which sent 100 ping packets consecutively, all get dropped. Otherwise, it was considered successful. For example, if the first 10 ping packets get dropped and the eleventh ping packet successfully returned, it was still considered a successful case and we then calculated the average RTT of the rest 90 packets. Figure 6(a) shows that even though N increases up to 31, the failure rates of EstiNet emulator over different N values are always zero. This shows that EstiNet emulator is very reliable. Conversely, Figure 6(b) shows that the failure rates over Mininet emulator can be very high over different values of N and across the three sets of experiments. We do not know how to explain this unreasonable phenomenon but this shows the instability of Mininet emulator.

Figure 7(a) and Figure 7(b) show the main memory consumption by EstiNet emulator and Mininet emulator, respectively. These measurements were performed by using the "top" command in Linux. Clearly, when the number of emulated OpenFlow switches increases, the consumption of main memory increases for both tools. However, one sees that Mininet emulator consumes more main memory than EstiNet emulator under the same number of OpenFlow switches. For example, when N^2=100, EstiNet emulator uses only 32 MB of main memory but Mininet emulator uses 171 MB of main memory, a factor of 5.34! As the number of emulated OpenFlow switches increases up to 961, the difference between the main memory consumption of EstiNet emulator and that of Mininet emulator becomes larger and reaches 7.35. These results show that EstiNet emulator consumes much less main memory than Mininet emulator. The reason for the large difference is that in EstiNet an OpenFlow switch is simulated as a module compiled and linked with its simulation engine while in Mininet an OpenFlow switch is simulated by a process and a namespace. Since the memory consumption of a process and a namespace is much larger than a module on Linux, this explains the large difference in their memory consumptions.



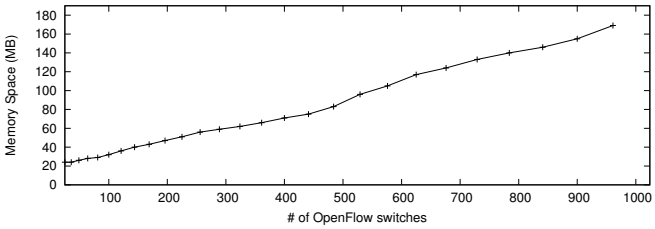
(a) EstiNet emulator



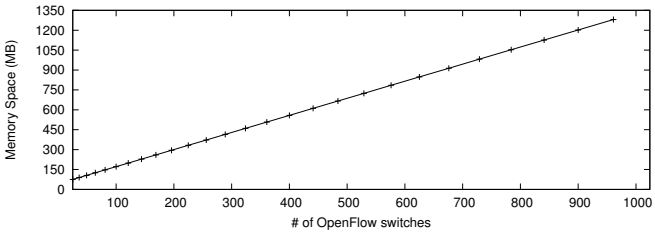
(b) Mininet emulator

Fig. 6: Failure rate of EstiNet emulator and Mininet emulator over a N x N grid network, where N = 5, 6, 7,..., 31.

Finally, we compared the total execution time between the two tools. The total execution time includes the tool launch time and network setup time, the execution time of the emulation, and the resource release time of emulation. The execution time of the emulation, which is 100 seconds in real time, is the same for both tools. Thus, the difference between the total execution time of these tools is their launch time, network setup time, and resource release time. Figure 8(a) and Figure 8(b) show the total execution time of EstiNet emulator and Mininet emulator, respectively. The total execution time of Mininet emulator is smaller than EstiNet emulator before N^2 reaches 36. When N^2 reaches 36, the total execution time of Mininet emulator, which is 120 seconds, is a bit smaller than the total execution time of EstiNet emulator, which is 132 seconds. When N^2 reaches 49, Mininet emulator begins to spend more time in launch time, network setup time, and resource release time than EstiNet emulator. The difference



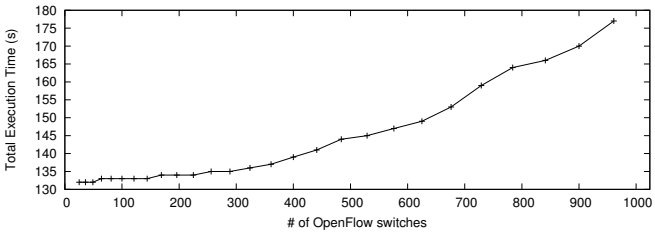
(a) EstiNet emulator



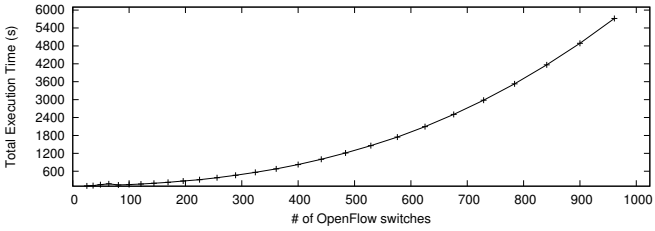
(b) Mininet emulator

Fig. 7: Main memory space consumption of EstiNet emulator and Mininet emulator over a N x N grid network, where N = 5, 6, 7,..., 31.

between the total execution time of Mininet emulator and that of EstiNet emulator becomes greater and greater when the value of N^2 keeps increasing. When N^2 reaches 961, the total execution time of Mininet emulator becomes approximately 32 times the total execution time of EstiNet emulator. According to our analysis, Mininet spends most of its total execution time in the launch time and network setup time to activate the processes used to emulate OpenFlow switches.



(a) EstiNet emulator



(b) Mininet emulator

Fig. 8: Total execution time of EstiNet emulator and Mininet emulator over a N x N grid network, where N = 5, 6, 7,..., 31.

V. Conclusion

In this paper, we compared the correctness, performance, and scalability of EstiNet OpenFlow network simulator, EstiNet OpenFlow network emulator, and Mininet OpenFlow network emulator. The EstiNet tool can operate in either the simulation mode or the emulation mode. We used the popular open source Floodlight OpenFlow controller to control the OpenFlow switches created in EstiNet simulator, EstiNet emulator, and Mininet emulator. In our experiments, we varied N from 5 to 31 to create a set of N x N grid networks with N^2 OpenFlow switches. We ran the real-world ping program to measure the RTTs experienced by ping packets, the failure rate of ping tests, and the main memory consumption of these tools when simulating/emulating N^2 OpenFlow switches. EstiNet simulator shows its correctness, accuracy, and scalability but it needs more time to simulate more OpenFlow switches. Mininet emulator generally worked well but generated inconsistent results in our tests under some network settings. In contrast, EstiNet emulator reveals correctness and accuracy before the number of OpenFlow switches reaches 961, beyond which its results started to deviate from the theoretical values bit.

References

[1] "Software-Defined Networking: The New Norm for Networks," a white paper of Open Networking Foundation, April 13, 2012.

[2] Nick Mckeown, Tom Anderson, Hari BalaKrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner, "OpenFlow: Enabling Innovation in Campus Networks," ACM SIGCOMM Computer Communication Review, Volume 38 Issue 2, April 2008.

[3] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. "An Integrated Experimental Environment for Distributed Systems and Networks," In Proc. of the Fifth Symposium on Operating Systems Design and Implementation, pages 255 - 270, Boston, MA, Dec. 2002.

[4] B. Chun, D. Culler, T. Roscoe, A. Bravier, L, Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: an Overlay Testbed for Broad-Coverage Services," ACM SIGCOMM Computer Communication Review, Volume 33 Issue 3, July 2003.

[5] S.Y. Wang, C.L. Chou, and C.M. Yang, "EstiNet 8.0 OpenFlow Network Simulator and Emulator," IEEE Communication Magazine, Vol. 51, Issue 9, September 2013.

[6] Bob Lantz, Brandon Heller, and Nick McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks" ACM Hotnets 2010, October 20-21, 2010, Monterey, CA, USA.

[7] Mininet – An Instant Virtual Network on your Laptop (or other PC), available at http://mininet.org/

[8] S.Y. Wang and H.T. Kung, "A Simple Methodology for Constructing Extensible and High-Fidelity TCP/IP Network Simulator," IEEE INFOCOM'99, March 21-25, 1999, New York, USA.

[9] S.Y. Wang, C.L. Chou, and C.C. Lin, "The Design and Implementation of the NCTUns Network Simulation Engine," Simulation Modelling Practice and Theory, 15 (2007) 57-81.

[10] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martn Casado, Nick McKeown, Scott Shenker, "NOX: towards an Operating System for Networks," ACM SIGCOMM Computer Communication Review, Volume 38 Issue 3, July 2008

[11] Ryu, a component-based software defined networking framework, available at http://osrg.github.io/ryu/.

[12] Floodlight OpenFlow controller, available at http://www.projectfloodlight.org/floodlight/

[13] Open Networking Summit 2013, Santa Clara, CA, Apr.15-17 2013

[14] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Extending Networking into the Virtualization Layer," in Prof. of HOTNETS 2009.

[15] Link Layer Discovery Protocol, IEEE 802.1AB standards.