Applying Mininet for Network Education

James T. Yu
DePaul University
Chicago, Illinois USA
jyu@cdm.depaul.edu

Abstract

This paper presents the use of Mininet to teach basic networking concepts for introductory courses. Mininet is a network *emulator* and it supports Linux administration functions for network management. The scope of network education covers (a) network design and configuration, (b) protocol analysis, and (c) performance measurement. We demonstrate that Mininet is effective and scalable in supporting these three functions, and its advantages to network *simulators* (e.g., ns2/ns3) and other network emulators (e.g., Open Networking Lab). We also identify some constraints and limitations of Mininet and our recommendation is to monitor host CPU occupancy while doing performance evaluation.

KEY WORDS: Mininet**,** On-Line Learning, Networking Education, Emulation, Simulation.

1. Introduction

With the growing importance of networking, the need to establish network labs to support student learning has been recognized for years [1][2]. A network lab could provide hands-on lab exercises where students could enforce their learning of networking theories through practical experiments. In general, there are three approaches to developing networking labs:

1 Physical labs. This is the most *practical* approach but it is also the most expensive one. A physical lab requires significant capital investment and on-going technical support for the labs. With the advancement of networking technology, lab equipment needs to be updated/upgraded within 3-5 years. Another critical issue of building physical labs is *scalability* as it is impossible to build a lab with hundreds of hosts, switches, routers.

2 Network Simulation. This approach is to use software to model the network behavior by *calculating* the interaction between different network entities (routers, switches, nodes, access points, links etc.). A popular network simulation tool is ns2 (www.isi.edu/nsnam/ns) which is evolved to ns3. Network simulation does not generate real *traffic* and users are required to develop their own tools to capture and analyze *simulated* traffic. It is most useful for performance analysis, especially for large scale network with hundreds or thousands of nodes. This approach is mostly used for *research* rather than *teaching*.

3 Network Emulation. This approach is to build a virtual network from which users can experiment with its functions, performance, and characteristics. An example is Open

Networking Lab (ONL) [3] which is evolved from Cisco PacketTracer. ONL supports the creation of a large virtual network through a friendly web interface. It is effective in learning and practicing network design and configuration, but it is not effective in conducting protocol analysis and performance evaluation. Although ONL is more scalable than physical labs, its lack of programming capability makes it difficult to create a large network with hundreds of nodes.

With the introduction of Software Defined Networking (SDN), Mininet becomes a popular tool to experiment with SDN concepts[4]. According to its own web site (www.mininet.org), Mininet is a network emulation orchestration system. *It runs a collection of end-hosts, switches, routers, and links on a single Linux kernel. It uses lightweight virtualization to make a single system look like a complete network.*

This paper is not about SDN; instead, it explores the use of Mininet to teach an introductory networking course. Similar to network simulation or emulation, Mininet supports the building of a large virtual network on a single physical computer [4]. This virtual environment is efficient for on-line learning (OL) where students could use their own computing equipment for hands-on lab exercises. An important advantage of Mininet over other approaches is its ability to use open-source software tools to generate *real* network traffic from which students can capture and analyze the traffic. The Mininet environment is also highly scalable to support hundreds of nodes.

Mininet is a programmable environment, and it often requires a controller to implement networking functions. As an introductory networking course, we do not have programming as a prerequisite and scripts are provided for the students to create various networks. The scripts are *parameterized* to allow variation and sizing of a network. The controller, if needed, is embedded in the script and hidden from the students.

2. Objectives of Networking Labs

The networking curriculum teaches students to understand the complexity of today's networks, and to build their expertise to design and manage these networks. From the theoretical perspective, students need to understand and appreciate communication protocols. From the practicum perspective, students need to design and manage the networks. Combining both, students need to predict (or estimate), measure, and evaluate network performance [5]. These three learning objectives are further expanded in the following subsections.

2.1 Protocol Analysis

Standards are the foundation of computer networks, and the Open System Interconnect (OSI) 7-layer model is usually introduced in the introductory networking course. Network protocols are specified in the standard documents and are mostly from ITU-T, IETF, and IEEE. Protocols define the *rules* of communication, and the rules are about the *syntax* and *semantics* of data being communicated. In addition to reading textbooks about protocol specifications, students would appreciate the purposes and functions of a protocol by *observing* the traffic. Therefore, hands-on exercises are important for students to *generate* and *capture* the traffic of the protocol under study. The availability of wireshark

(www.wireshark.org) is a blessing for network education because it is not only a traffic capturing tool but also a protocol analyzer. The challenge, however, is to generate the traffic to be captured by wireshark.

2.2 Network Design and Configuration

Network Design includes the physical design and logical design. The physical design includes (a) selection of network devices, and (b) physical connection between network devices. The logical design includes (a) logical connection between devices, (b) Virtual LAN (VLAN), (c) IP subnets, (d) static IP address assignments, (e) dynamic IP address schemes, and (f) communication protocols between devices. Hands-on lab exercises should support students to design a relatively large network with multiple subnets, and then implement the network to observe traffic flows.

2.3 Performance Analysis

As any *engineering* discipline, the use of data (i.e., numbers in the context of networking) is essential in network application. Students are required to use *numbers* to size the network, and to describe the quality, performance, and limitations of the network under construction. From their understanding of network theory, students could estimate the network performance. After building and experimenting with the network, students should measure network performance and compare the empirical results with their original estimates. In many cases, lab measurements would not conform to the original estimates. As a result, students need to determine the cause of difference (i.e., causes of variations). Students may need to reevaluate their understanding of the theory (i.e., protocols), to redo the lab work and verify the device configuration, to study the results which may be due to unexpected causes of variation, or any combination of the above.

3. Building Network Components

A basic network includes hosts, hubs (layer-1), switches (layer-2), and routers (layer-3). We can further distinguish a host as a workstation and a server where a server runs Linux daemon processes to provide services, such as web service, DNS service, proxy service, etc. This section shows the capability of Mininet to build these basic networking components.

3.1 Host

Mininet is a virtual machine (VM) built on the Oracle VirtualBox (www.virtualbox.org) which is a hypervisor for virtualization on various operating systems, including Windows, Linux, MacOS, and Solaris. A Mininet host is a virtual entity on a Mininet VM as illustrated in Figure 1.

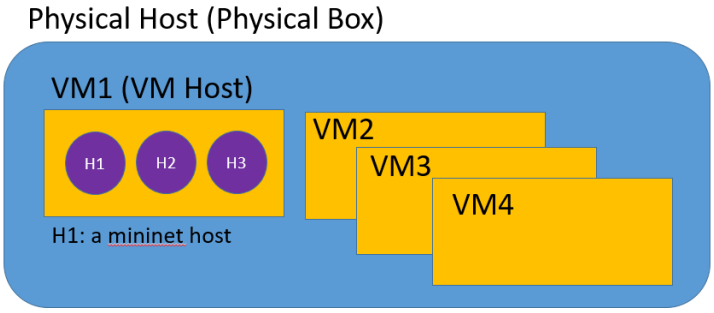


Figure 1. Mininet Hosts Environment

A Mininet host shares its environment with its hosted VM, and all Mininet hosts share the same file system as their VM as illustrated in Figure 2.

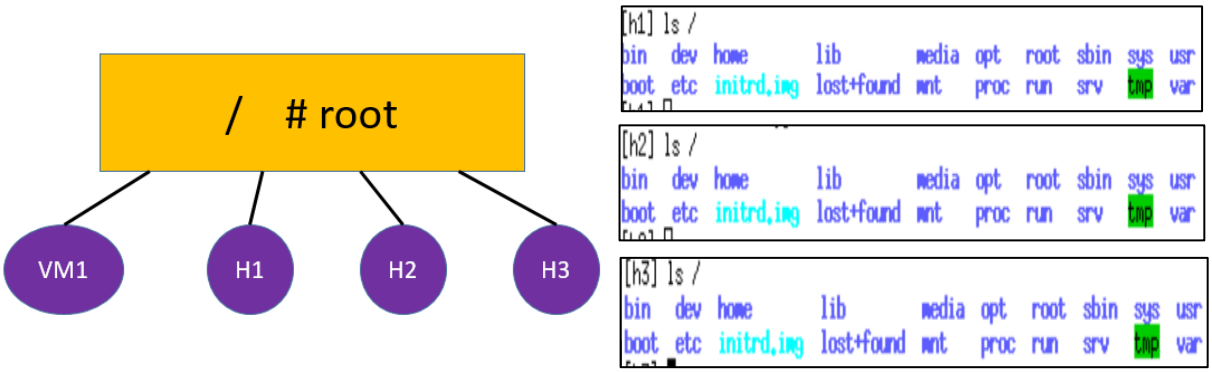


Figure 2. Mininet Host File System

A user can access a Mininet host via the Mininet console, but a better way to access a Mininet host is via **xterm** (X11 terminal application). All Mininet hosts are running the same operating system as their VM. The commands that are available to the VM are also available to the Mininet hosts. Although Mininet hosts share the same file system, they have their own interfaces, IP addresses, and MAC addresses:

```
[h1] ifconfig
h1-eth0   Link encap:Ethernet  HWaddr ce:8c:98:7a:77:89
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:932 (932.0 B)  TX bytes:1196 (1.1 KB)
```

```
[h2] ifconfig
h2-eth0   Link encap:Ethernet  HWaddr 4a:8c:a9:a6:0e:98
          inet addr:10.0.0.2  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1196 (1.1 KB)  TX bytes:932 (932.0 B)
```

3.2 Ethernet Hub

Ethernet hub is a basic device to build networks, and its practical use has been replaced by Ethernet switch. Although it is unlikely to find an Ethernet hub on the market, it is still useful to study its function for network education. If a hub has only two ports, it is also called repeater. Mininet does not have a built-in device called hub, but it provides software code to program a forwarding switch to function as a hub. With that, we can build a simple hub network as illustrated in Figure 3.

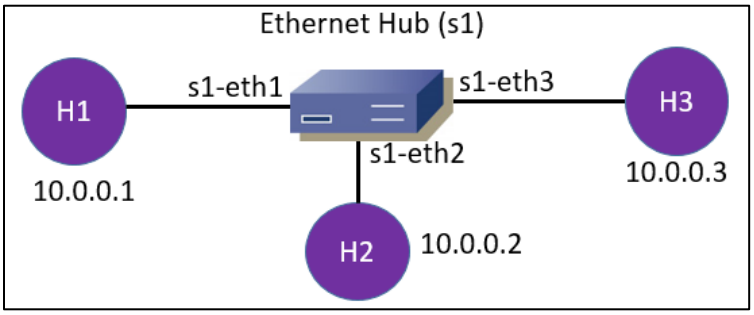


Figure 3. Network of Single Ethernet Hub

An Ethernet hub receives traffic from an incoming port and forwards the traffic to all other ports on the hub. It does not use any information in the packet header (no address), and simply forwards the traffic. To validate and learn the hub behavior, we send the traffic between h1 and h2, and use the **tcpdump** command to monitor traffic coming to h3 as illustrated in Figure 4.

```
mininet> h1 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.208 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.095 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.072 ms
```

```
[tdc411s50] sudo tcpdump -i s1-eth3 -p icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol
listening on s1-eth3, link-type EN10MB (Ethernet), capture size 262
09:14:49.244549 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 1594,
09:14:49.244624 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 1594, s
09:14:50.244903 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 1594,
09:14:50.244945 IP 10.0.0.2 > 10.0.0.1: IC
09:14:51.246040 IP 10.0.0.1 > 10.0.0.2: IC
09:14:51.246073 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 1594, s
```

ICMP traffic is observed on h3.

Figure 4. Validation of Ethernet Hub

3.3 Ethernet Switches

A Mininet switch is also a virtual entity and its behavior can be specified by the users. By default, a Mininet switch follows the standard (IEEE 802.1D) to forward an incoming frame based on its destination MAC address. For example, we create a switch with three hosts (h1, h2, and h3) connected to a switch as illustrated in Figure 5.

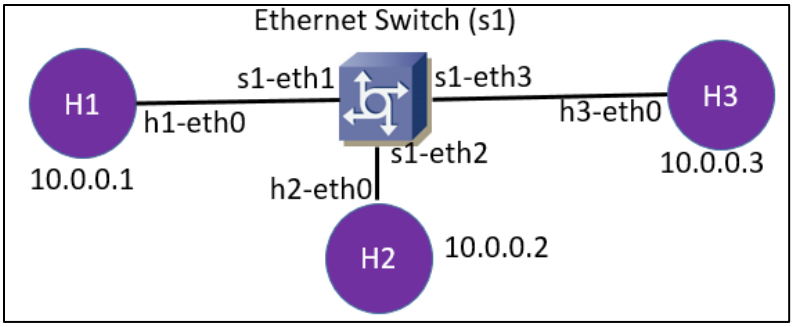


Figure 5. Mininet Network with Single Switch

If the network device is a switch, unicast traffic between h1 and h2 would not be observed on h3. We used the **ping** command to generate the traffic from h1 to h2 and used the utility **tcpdump** to monitor the traffic to h2 (live traffic) and h3 (no traffic) as illustrated in Figure 6.

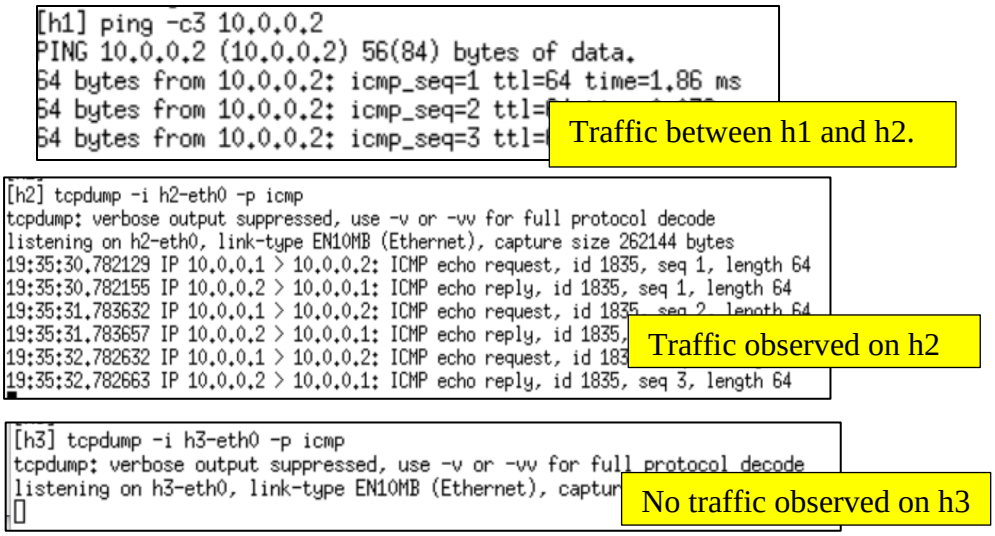


Figure 6. Validation of Ethernet Switch

In addition to tcpdump, Mininet also supports the use of wireshark to capture the traffic as illustrated in Figure 7.

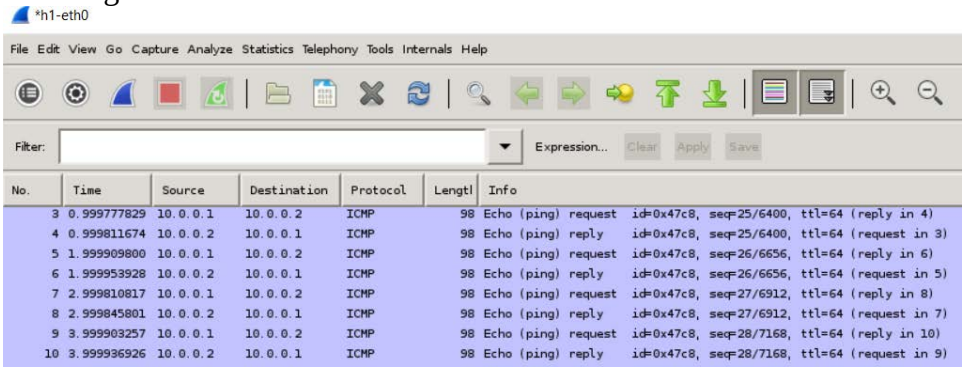


Figure 7. Network Traffic Captured in Wireshark

The default Mininet switch does not support Spanning Tree Protocol (STP), or other layer-2 control protocols. The support of STP is discussed Section 4.2.

3.4 IP Router

One approach to implement a router on Mininet is to enable forwarding on a host as illustrated in Figure 8.

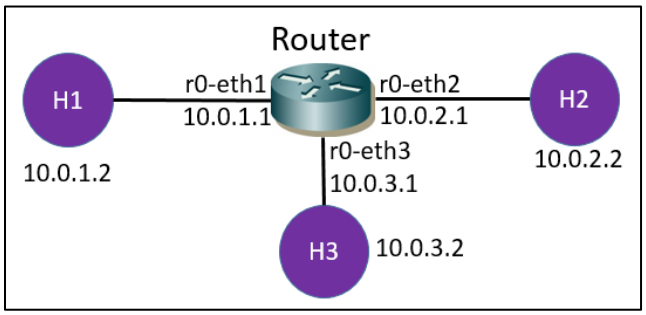


Figure 8. Network with Single Router

To prove traffic going through the router, we also used **tcpdump** on the router to monitor the traffic on its interfaces. The following screenshot shows the ICMP traffic between h1 and h2 is observed on the router interfaces (r0-eth1 and r0-eth2).

```
[r0] tcpdump -i r0-eth1 -p icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on r0-eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
21:06:42.451070 IP 10.0.1.2 > 10.0.2.2: ICMP echo request, id 2359, seq 1, length 64
21:06:42.451087 IP 10.0.2.2 > 10.0.1.2: ICMP echo reply, id 2359, seq 1, length 64
21:06:43.450072 IP 10.0.1.2 > 10.0.2.2: ICMP echo request, id 2359, seq 2, length 64
21:06:43.450091 IP 10.0.2.2 > 10.0.1.2: ICMP echo reply, id 2359, se
21:06:44.449074 IP 10.0.1.2 > 10.0.2.2: ICMP echo request, id 2359,
21:06:44.449091 IP 10.0.2.2 > 10.0.1.2: ICMP echo reply, id 2359, se
^C
```

Traffic on the left side of the router

```
[r0] tcpdump -i r0-eth2 -p icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on r0-eth2, link-type EN10MB (Ethernet), capture size 262144 bytes
21:06:59.486776 IP 10.0.1.2 > 10.0.2.2: ICMP echo request, id 2365, seq 1, length 64
21:06:59.486789 IP 10.0.2.2 > 10.0.1.2: ICMP echo reply, id 2365, seq 1, length 64
21:07:00.485777 IP 10.0.1.2 > 10.0.2.2: ICMP echo request, id 2365, seq 2, length 64
21:07:00.485788 IP 10.0.2.2 > 10.0.1.2: ICMP echo reply, id 2365, se
21:07:01.484779 IP 10.0.1.2 > 10.0.2.2: ICMP echo request, id 2365,
21:07:01.484790 IP 10.0.2.2 > 10.0.1.2: ICMP echo reply, id 2365, se
^C
```

Traffic on the right side of the router

Figure 9. Validation of IP Router

In the router test, we observed a deficiency in Mininet that it does not support the "ping –R" option as illustrated in Figure 10.

```
[h1] ping -R 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(124) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=63 time=0.031 ms
RR:     10.0.2.2
        10.0.2.2
        10.0.2.2
        10.0.2.2
        10.0.2.2
        10.0.2.2

64 bytes from 10.0.2.2: icmp_seq=2 ttl=63 time=0.114 ms (same route)
^C
```

Expected Output:
10.0.1.2
10.0.2.1
10.0.2.2
10.0.2.2
10.0.1.1
10.0.1.2

Figure 10. Deficiency of "ping –R" on Mininet

The "ping –R" option uses the Option field in the IP packet to trace the IP addresses on the route, and Mininet correctly shows the number of hops from the source to the

destination, but it does not provide the IP addresses on the route as shown in the shaded box in Figure 10.

4. Networking Protocols

4.1 Layer-2 Protocols (STP)

Mininet supports the Spanning-Tree Protocol (STP) [6] in both LinuxBridge and OVSBridge. To test the bridged configuration with STP, we create a network with loop topology as illustrated Figure 11. The **brctl** utility is a tool to manage and check the STP configuration, such as the identification of the root switch and the blocked port(s).

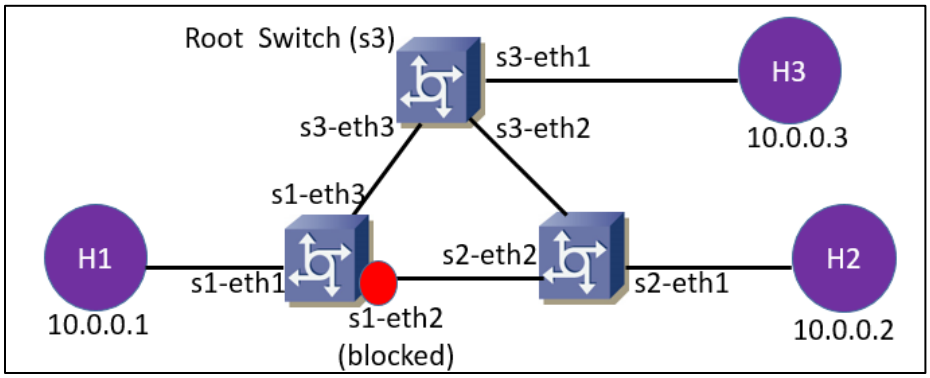


Figure 11. Switched Network with Loop Topology

To validate the STP behavior, we compared the **ping** performance before and after the STP configuration. If STP is not enabled, there would be *broadcast storm* where a single broadcast message could replicate itself hundreds or thousands of times and saturate the network in a few seconds. When broadcast storm occurs, we would observe significant packet loss due to the saturation of broadcast messages. If STP is enabled, there would be no broadcast storm and no packet loss. The packet loss data before and after the STP configuration is illustrated in Figures 12(a) and 12(b).

```
mininet> h1 ping -f -c100 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
.........................................
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 55 received, 45% packet loss, time 2192ms
rtt min/avg/max/mdev = 11.880/21.637/57.011/10.942 ms, pipe 5, ipg/ewma
mininet> h1 ping -f -c100 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
.....................................
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 60 received, 40% packet loss, time 1402ms
rtt min/avg/max/mdev = 12.054/22.355/77.051/14.449 ms, pipe 6, ipg/ewma
mininet> h1 ping -f -c100 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
.....................................................................
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 19 received, 81% packet loss, time 1333ms
rtt min/avg/max/mdev = 17.041/106.937/564.807/139.437 ms, pipe 44, ipg/ew
```

Figure 12(a). Performance before STP Configuration (Broadcast Storm)

```
mininet> h1 ping -f -c1000 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
1000 packets transmitted, 1000 received, 0% packet
rtt min/avg/max/mdev = 0.003/0.007/0.227/0.014 ms,
mininet> h1 ping -f -c1000 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
1000 packets transmitted, 1000 received, 0% packet
rtt min/avg/max/mdev = 0.003/0.009/0.313/0.017 ms,
mininet> h2 ping -f -c1000 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
1000 packets transmitted, 1000 received, 0% packet
rtt min/avg/max/mdev = 0.003/0.006/0.103/0.008 ms,
```

Figure 12(b). Performance after STP Configuration (no packet loss)

Another useful learning experiment of STP is to measure its failover time and fall-back time. STP has three timers:

- Max Age Time (default=20 sec)
- Forward Delay Timer (default=15 sec)
- Hello Timer (default=2 sec)

In the case of a simple loop topology as Figure 11, the failover time is 2×"forward delay" = 30 sec. To measure the failover time, we can simply count the number of lost ICMP packets during the failover test which to shut down an interface on the root switch. The results of failover time and fall-back measurements are illustrated in Figures 13(a) and 13(b). Note that the observed failover time and fall-back time match almost perfectly with the STP timers.

```
[h1] ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64
64 bytes from 10.0.0.2: icmp_seq=33 ttl=64
64 bytes from 10.0.0.2: icmp_seq=34 ttl=64 time=0.071 ms
64 bytes from 10.0.0.2: icmp_seq=35 ttl=64 time=0.070 ms
```

Packet loss = 30 packets
Failover time = 30 sec

Figure 13(a). STP Failover Time

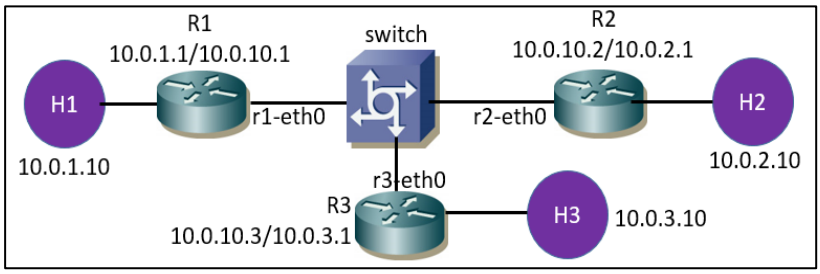
```
64 bytes from 10.0.0.2: icmp_seq=35 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=36 ttl=6
64 bytes from 10.0.0.2: icmp_seq=66 ttl=6
64 bytes from 10.0.0.2: icmp_seq=67 ttl=64
64 bytes from 10.0.0.2: icmp_seq=68 ttl=64 time=0.162 ms
64 bytes from 10.0.0.2: icmp_seq=69 ttl=64 time=0.079 ms
```

Packet loss = 29 packets
Fall-back time = 29 sec

Figure 13(b). STP Fall-Back Time

4.2 Static IP Routing

For simple network with only a few routers, static routing is preferred to establish routes among the nodes. An example of using static routes is illustrated in Figure 14, along with the routing tables on the routers.




```
mininet> r1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.1.0        *               255.255.255.0   U     0      0        0 r1-eth1
10.0.2.0        10.0.10.2       255.255.255.0   UG    0      0        0 r1-eth0
10.0.3.0        10.0.10.3       255.255.255.0   UG    0      0        0 r1-eth0
10.0.10.0       *               255.255.255.0   U     0      0        0 r1-eth0
mininet> r2 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.1.0        10.0.10.1       255.255.255.0   UG    0      0        0 r2-eth0
10.0.2.0        *               255.255.255.0   U     0      0        0 r2-eth1
10.0.3.0        10.0.10.3       255.255.255.0   UG    0      0        0 r2-eth0
10.0.10.0       *               255.255.255.0   U     0      0        0 r2-eth0
mininet> r3 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.1.0        10.0.10.1       255.255.255.0   UG    0      0        0 r3-eth0
10.0.2.0        10.0.10.2       255.255.255.0   UG    0      0        0 r3-eth0
10.0.3.0        *               255.255.255.0   U     0      0        0 r3-eth1
10.0.10.0       *               255.255.255.0   U     0      0        0 r3-eth0
```

Figure 14. Simple Network with Static Routing

The validation of static routing is the same as the single router test, and we ran **tcpdump** on the router interfaces to monitor traffic. As discussed earlier, the lack of support "ping –R" is a major deficiency of Mininet; otherwise, the route verification would be a lot easier.

5 Performance Measurements

5.1 Bandwidth and CPU Occupancy

Research has shown that the CPU occupancy of the physical host could have significant impact on Mininet performance [7]. Therefore, we conducted a baseline experiment to understand the constraint of the Mininet environment. The performance data of this paper is collected on a laptop computer of dual CPU processors (Intel M-5Y71, 1.2GHz and 1.4GHz) with 8G RAM. A general guideline of performance evaluation is that CPU occupancy should be <60%; otherwise, congestion and long waiting would happen. The baseline experiment is to run a performance test between two hosts on the following network:

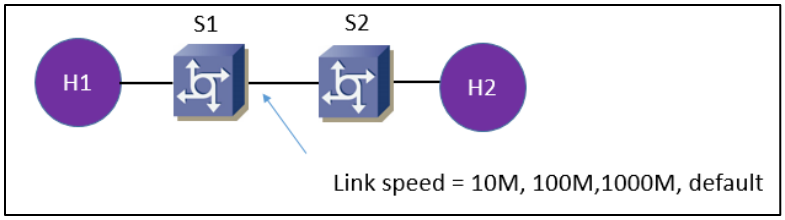


Figure 15. Network for Baseline Testing

In the case of default link setting, we observed throughput of 27.6G bps. However, the CPU occupancy of VM is close to 100%, and the CPU occupancy of the physical host is close to 60%. Therefore, the observed throughput value has no significance as it is constrained by the CPU time allocated to the VM. We then configured the link bandwidth at 10M, 100M, and 1,000M, and ran the test again. CPU occupancies under different link speeds are given in Table 1.

Table 1. CPU Occupancy of Performance Test

Link Speed	VM CPU Occupancy	Physical Host Occupancy
10M	15%	4%
100M	25%	4%
1,000M	55%	6%

We computed RTT for different link speeds, and the results are illustrated in Figure 16.

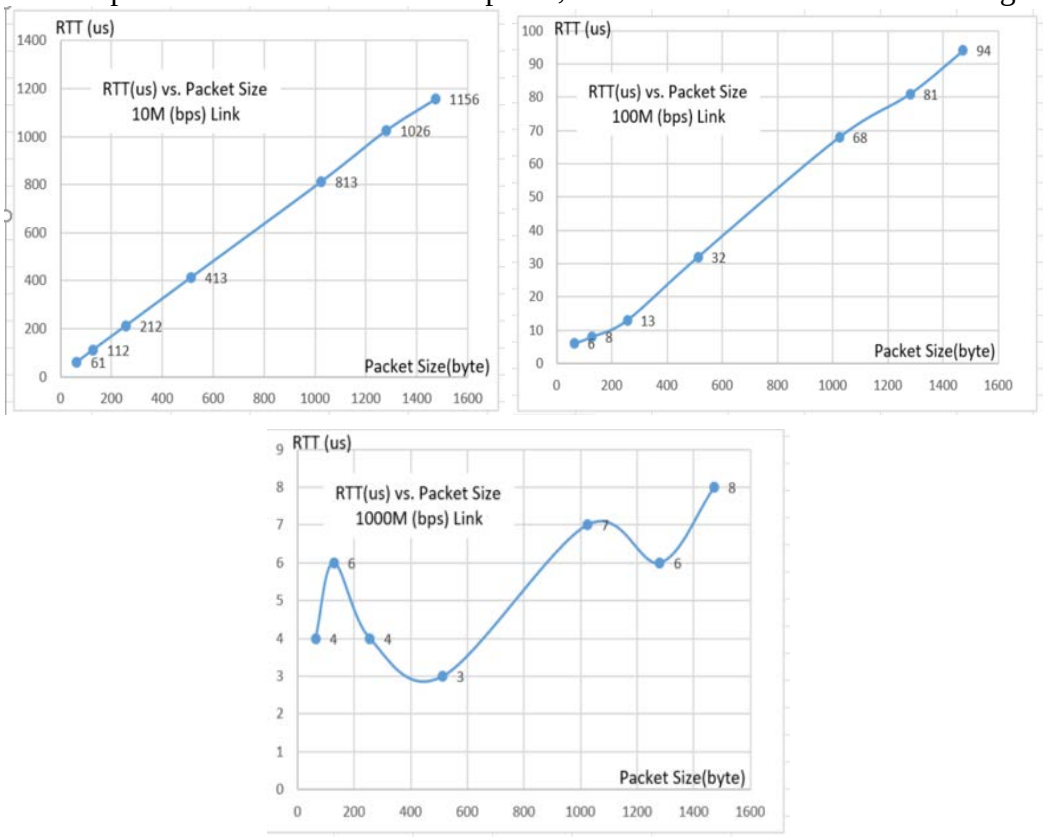


Figure 16. Round Trip Time (us) vs. Packet Size for Various Link Speeds

From the smoothness of the curve, we conclude that the effective performance evaluation would limit the link speed at 10M bps or 100M bps on the Mininet environment for this physical host. The result of 1000M link shows unknown causes of variation and makes it difficult interpret the data. Setting the link speed at 10M bps or 100M bps could avoid overloading the CPU.

5.2 Switch Operation Mode

Mininet provides several built-in modules to create a relatively large network, such as linear topology or hierarchical (tree) topology. Figure 18 illustrates a network of linear topology of 50 switches and hosts. We experimented with the link speed of both 10Mbps and 100Mbps. The data of 10Mbps is more stable (the curve is more smooth) than100Mbps, and is presented in this paper.

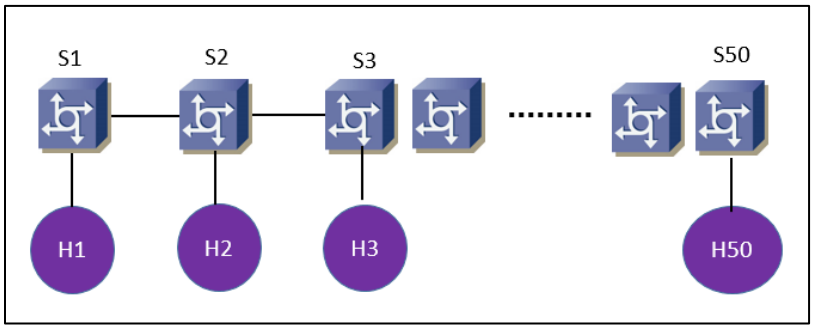


Figure 17. Linear Topology of Switched Network

The standard, RFC2544 [8], recommends the use of packet sizes of 64, 128, 256, 512, 1024, 1280, and 1500 (MTU) bytes to compute Round Trip Time (RTT). We tested all packet sizes, and the data of four packet sizes and their relationship to the number of switch hops are illustrated in Figure 18.

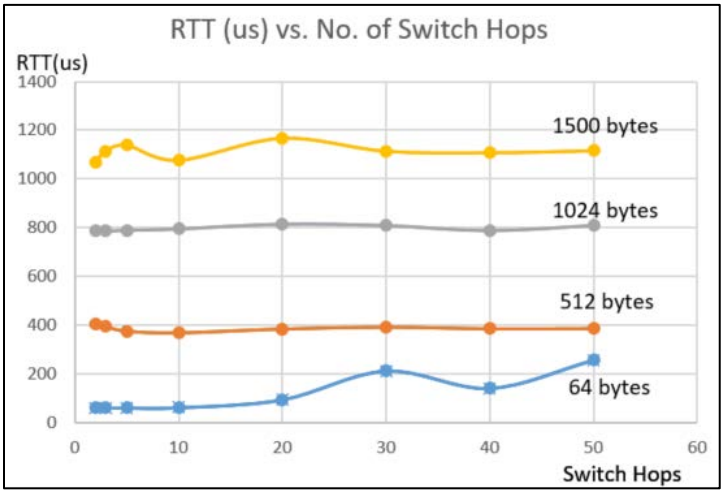


Figure 18. Round Trip Time vs. Number of Switch Hops

It is interesting to note that *RTT is independent of the number of hops* in Mininet. For example, RTT of packet size 1,024 byte is almost the same for two hops and 50 hops (786 us vs 810 us). Our observation is consistent with the data published in [9]. In their study, they consider this effect a deficiency of Mininet in performance analysis and conclude

OPNET (www.opnetprojects.com) is more desirable for stability and reliability tests. However, we consider it a difference between the two operation modes of Ethernet switch: cut-through and store-and-forward [10]. In the case of cut-through operation, the switch reads only the first six bytes of Ethernet frame (i.e., the destination MAC address) and then makes a forwarding decision. As a result, it is able to achieve wire-speed performance, i.e., going through a switch is the same as going through the wire. In the case of store-and-forward operation, the switch reads the entire Ethernet frame, performs the checksum (a.k.a. frame check sequence), and then makes the forwarding decision. RTT of store-and-forward operation would linearly increase with the number of hops (as observed in OPNET). The data (Figure 18) confirms that Mininet switch is operating in the cut-through mode. The throughput data also shows that the number of switch hops is irrelevant to the performance as illustrated in Figure 19.

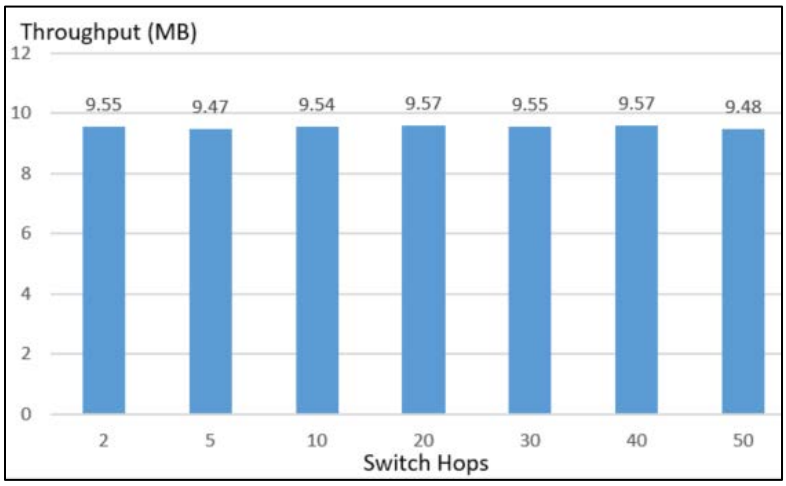


Figure 19. Throughput vs. Switch Hops

However, the reason for the throughput data (independent of switch hops) is different from the RTT data. We also observed the same results for throughput (independent of switch hops) on the *store-and-forward* switch. This seemingly counter-intuitive results can be easily explained to the students with the basics of queuing theory that traffic through multiple hops would not affect throughput.

6 Conclusions

This paper presents the use of Mininet for introductory networking courses, and we demonstrate the scalability of a relatively large network with a variety of network devices. The advantage of Mininet is its ease of use on a virtual computing environment. It has the support of many open-source tools on the Linux environment for traffic generation, capture, and analysis. The lab experiments show the efficiency and effectiveness of using **tcpdump** and **wiresahrk** to capture and analyze network traffic for student learning, and this is a major advantage of Mininet over other simulation or emulation approaches to network education. We identify a constraint of conducting performance evaluation on Mininet where host and VM CPU occupancy could be a limiting factor and should be monitored during the experiment. A disadvantage of Mininet is a lack of support of many networking protocols. However, the use of those advanced protocols could be replaced by Software Defined Network (SDN) which is a topic in more advanced networking courses.

References

[1] S. A. Mengel and C. D. Bowling, “Supporting Networking Courses with a Hands-on Laboratory,” Frontiers in Education Conference, Atlanta, GA, November 1995.

[2] N. I. Sarkar, “Teaching computer networking fundamentals using practical laboratory exercises,” IEEE Transactions on Education, Volume 49, Issue 2, May 2006, pp. 285 – 291

[3] A. Mikroyannidis, *et. al*. “The Open Networking Lab: Hands-on Vocational Learning in Computer Networking,” IEEE Frontiers in Education Conference, San Jose, CA, October 2018.

[4] B. Lantz, B. Heller, N. McKeown, “A network in a laptop: rapid prototyping for software-defined networks,” the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Monterey, CA, October 2010.

[5] James T. Yu, "Supporting Hands-on Networking Lab Exercises for On-Line Students," Proceedings of 2016 International Conference on E-Learning, E-Business, E-Government (EEE'16), Las Vegas, NV, July 2016, pp. 48-53.

[6] IEEE 802.1D-2004. IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges.

[7] J. Ortiz, J. Londoño, and F. Novillo, “Evaluation of performance and scalability of Mininet in scenarios with large data centers,” 2016 IEEE Ecuador Technical Chapters Meeting (ETCM), Guayaquil, Ecuador, October, 2016.

[8] RFC2544, Benchmarking Methodology for Network Interconnect Devices, March 1999.

[9] Seungwoon Lee, Jehad Ali, Byeong-hee Roh, “Performance Comparison of Software Defined Networking Simulators for Tactical Network: Mininet vs. OPNET, 2019 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI USA, February 2019, pp. 192-202

[10] Cisco whitepaper. “Cut-Through and Store-and-Forward Ethernet Switching for Low Latency Environments,” https://www.cisco.com/c/en/us/products/collateral/switches/nexus-5020-switch/white_paper_c11-465436.pdf