

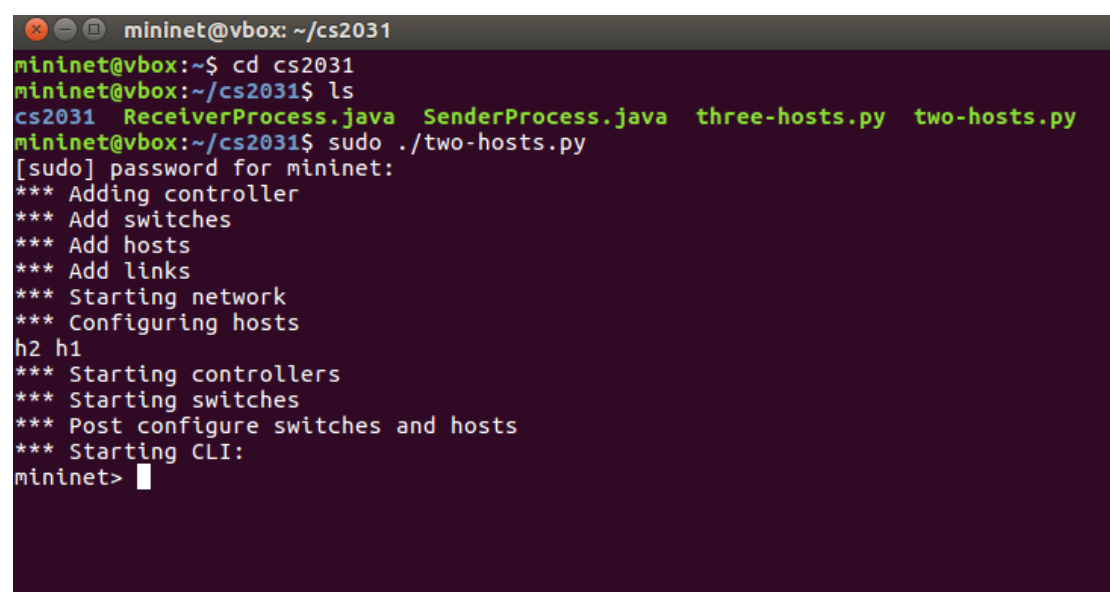
Mininet

Disclaimer: The use of Mininet for the assignments is optional – All assignments can be implemented and run between personal laptops or computers in the labs as well. No one will be penalised for not using Mininet for the assignments.

Mininet [1] provides a simple emulator for networks based on containers, a light alternative to virtual machines. The virtual machine image “CS2031.ova” consists of a Ubuntu 16.04 distribution, Mininet 2.3.0, two small topology examples and two short Java programs. The login for the only user on the system is “mininet” and the password for this account is “mininet”.

Walkthrough:

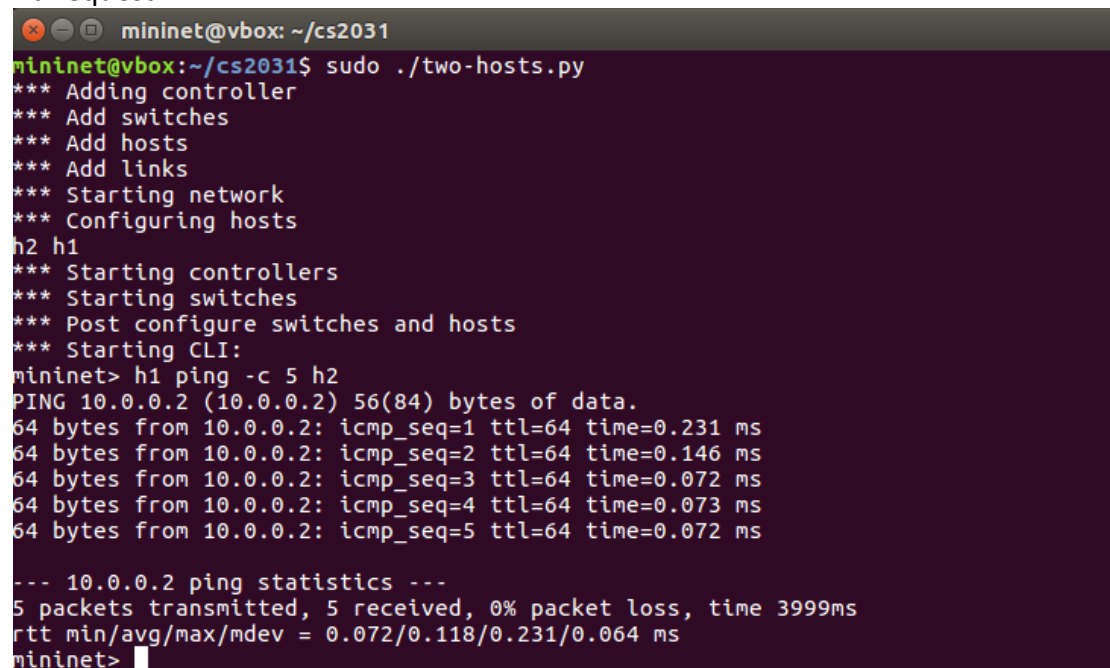
1. Install Oracle VirtualBox [2] – and possibly its extension – or VMware Fusion [3]. In the following, I will assume that VirtualBox is being used.
2. Copy the virtual machine image “CS2031.ova” to your disk drive.
3. Double click the virtual machine image – This should start VirtualBox and allow you to “Import Virtual Appliance”. After this, VirtualBox should show a new virtual machine called “Mininet CS2031”.
4. Double click/start virtual machine called “Mininet CS2031”. The system may report that it has an issue with one of the host-only adapters. If no “host-only” adapter exists in the setup of VirtualBox, you will need to enable an additional adapter under the Network tab and declare it as “host-only”
5. This should start Ubuntu 16.04 version and you should be able to login as “mininet” with the password “mininet”.
6. Open a terminal (e.g. icon on left-hand Launcher bar).
7. Go into the directory “cs2031” i.e. “cd cs2031”.
8. Execute the python script “two-hosts.py” as root i.e. “sudo ./two-hosts.py”.



```
mininet@vbox: ~/cs2031
mininet@vbox:~$ cd cs2031
mininet@vbox:~/cs2031$ ls
cs2031  ReceiverProcess.java  SenderProcess.java  three-hosts.py  two-hosts.py
mininet@vbox:~/cs2031$ sudo ./two-hosts.py
[sudo] password for mininet:
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h2 h1
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet>
```

Figure 1: Starting mininet with a simple topology of 2 hosts connected to each other.

9. Execute “h1 ping -5 h2” – This will send 5 request packets from host h1 to host 2 and h2 will respond to each packet with a reply. The time that is shown for each sequence number is how long it took for the reply packet to arrive at h1 after it sent a request.



```

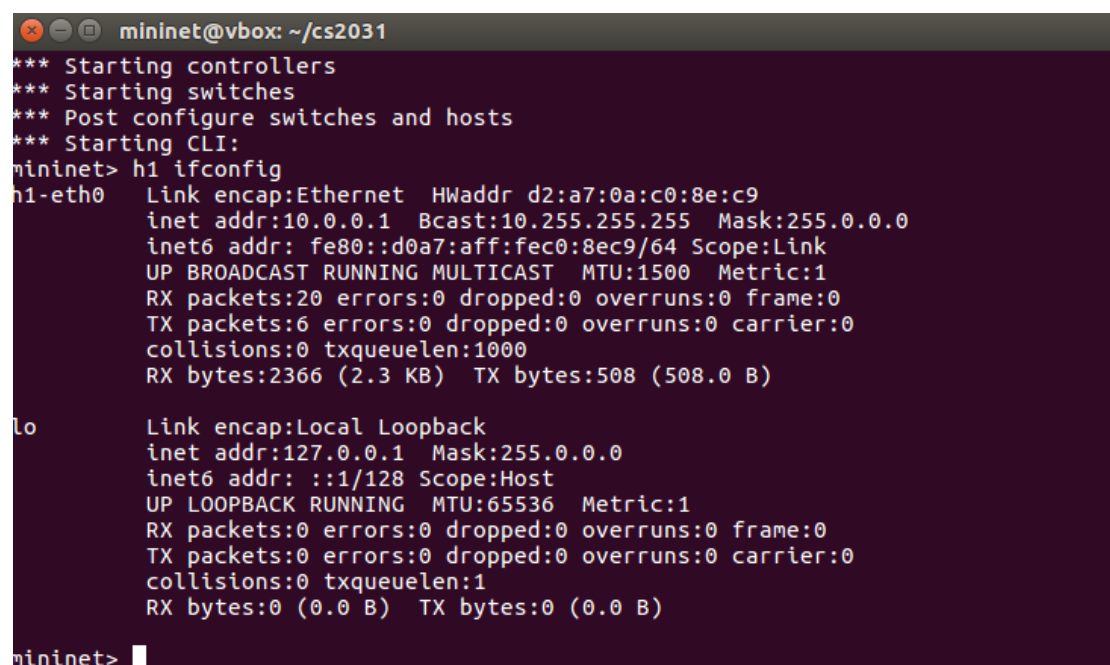
mininet@vbox: ~/cs2031
mininet@vbox:~/cs2031$ sudo ./two-hosts.py
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h2 h1
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet> h1 ping -c 5 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=0.231 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.146 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.072 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.073 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.072 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.072/0.118/0.231/0.064 ms
mininet>

```

Figure 2: Simple ping between host h1 and host 2.

10. Execute “h1 ifconfig” – This will show you information for the interfaces at host h1, including its IP addresses e.g. in the case in figure 3 “10.0.0.1”.



```

mininet@vbox: ~/cs2031
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet> h1 ifconfig
h1-eth0  Link encap:Ethernet  HWaddr d2:a7:0a:c0:8e:c9
         inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
         inet6 addr: fe80::d0a7:aff:fec0:8ec9/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:20 errors:0 dropped:0 overruns:0 frame:0
         TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:2366 (2.3 KB)  TX bytes:508 (508.0 B)

lo       Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.0.0.0
         inet6 addr: ::1/128 Scope:Host
         UP LOOPBACK RUNNING  MTU:65536  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

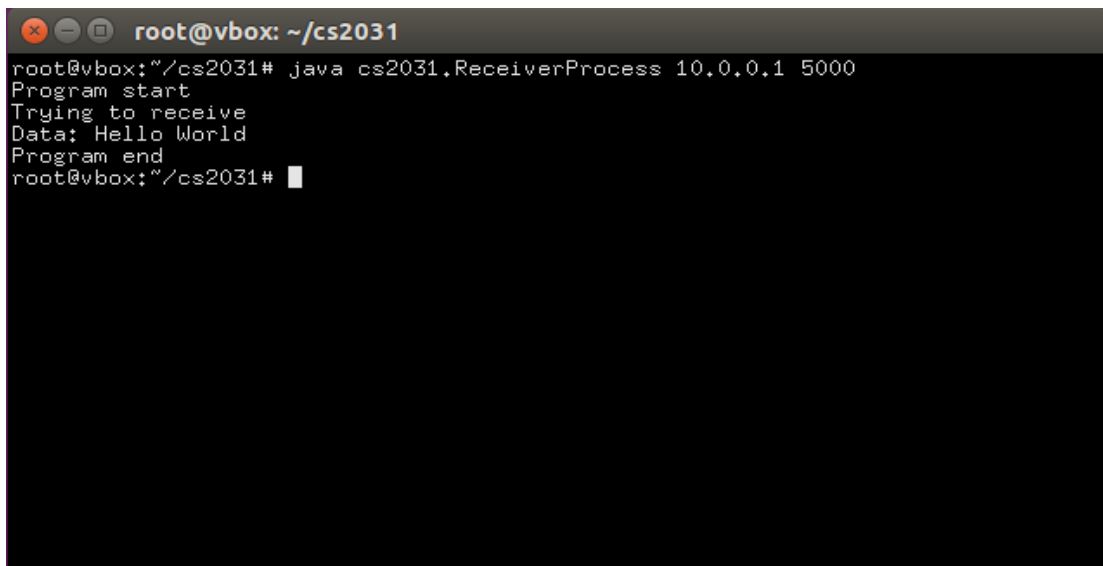
mininet>

```

Figure 3: Interface information for host h1.

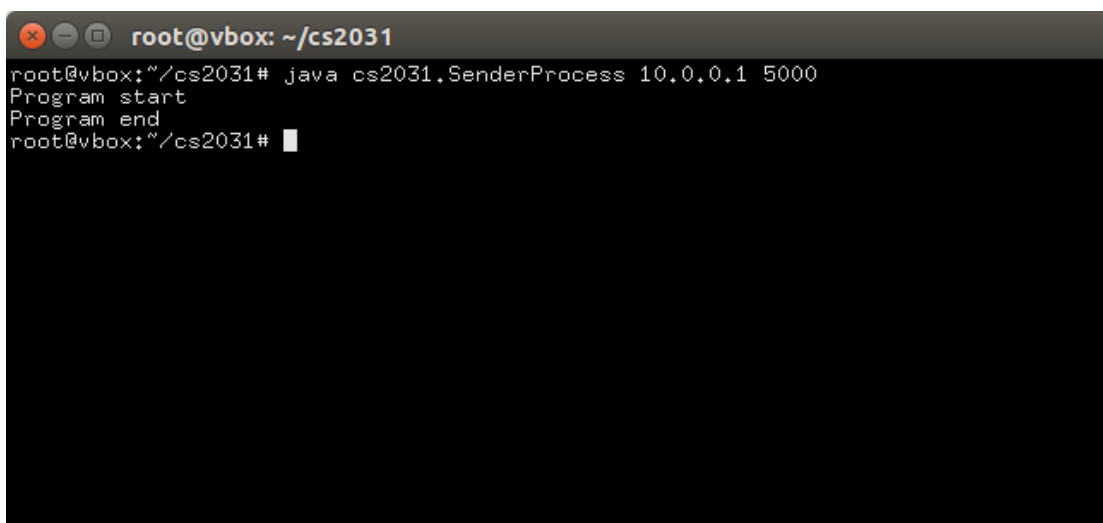
11. In Mininet execute the following commands:

- a. "mininet> h1 wireshark &" <Enter> - This will start a capture program called "wireshark" [4] as an independent process on host h1 (Ignore the warning "Lua: Error during loading" by clicking <Ok>).
- b. Click "h1-eth0" – This will start capturing packets on the device eth0.
- c. Go back to the terminal by using <alt-tab> or Launcher bar.
- d. Start two xterminals, one for each host:
 "mininet> h1 xterm &" and
 "mininet> h2 xterm &".
(Ctrl-Right click will allow you to change the font size of the terminal.)
- e. In the xterminal for host h1 start a the program ReceiverProcess
 "java cs2031.ReceiverProcess 10.0.0.1 5000 &" <Enter>
 and in the xterminal for host h2 start the program SenderProcess
 "h2 java cs2031.SenderProcess 10.0.0.1 5000" <Enter>



```
root@vbox: ~/cs2031
root@vbox:~/cs2031# java cs2031.ReceiverProcess 10.0.0.1 5000
Program start
Trying to receive
Data: Hello World
Program end
root@vbox:~/cs2031#
```

Figure 4: Output of the xterminal of host h1, showing that ReceiverProcess received the string "Hello World" from the SenderProcess.



```
root@vbox: ~/cs2031
root@vbox:~/cs2031# java cs2031.SenderProcess 10.0.0.1 5000
Program start
Program end
root@vbox:~/cs2031#
```

Figure 5: Output of the xterminal for host h2 after the SenderProcess has completed.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.000067930	10.0.0.2	10.0.0.1	UDP	61	41162 → 5000 Len=19

▶ Frame 3: 61 bytes on wire (488 bits), 61 bytes captured (488 bits) on interface 0
 ▶ Ethernet II, Src: 4a:9c:3d:6e:8c:4d (4a:9c:3d:6e:8c:4d), Dst: d2:a7:0a:c0:8e:c9 (d2:a7:0a:c0:8e:c9)
 ▶ Internet Protocol Version 4, Src: 10.0.0.2, Dst: 10.0.0.1
 ▼ User Datagram Protocol, Src Port: 41162 (41162), Dst Port: 5000 (5000)
 Source Port: 41162
 Destination Port: 5000
 Length: 27
 ▶ Checksum: 0xc189 [validation disabled]
 [Stream index: 0]
 ▼ Data (19 bytes)
 Data: aced0005770d000b48656c6c6f20576f726c64
 [Length: 19]

0000	d2 a7 0a c0 8e c9 4a 9c	3d 6e 8c 4d 08 00 45 00J. =n.M..E.
0010	00 2f 20 a3 40 00 40 11	06 19 0a 00 00 02 0a 00	./ .@.
0020	00 01 a0 ca 13 88 00 1b	c1 89 ac ed 00 05 77 0dw.
0030	00 0b 48 65 6c 6c 6f 20	57 6f 72 6c 64	..Hello World

Figure 6: Output of Wireshark, showing the packet that Wireshark captured from the transmission between host h1 and host h2. The string "Hello World" is visible at the end of the payload. The payload is longer than the string because it contains additional information about the representation of the string in Java e.g. its length, etc.

- f. To close everything you can type "mininet> quit" <Enter> which will close wireshark and mininet.

The original example of SenderProcess sends the string "Hello World". The modification below fills an array partially with numbers from 1 to 6 and sends these instead of the string.

```

mininet@vbox: ~/cs2031

// extract destination from arguments^M
address= InetAddress.getByName(args[0]);^M
port= Integer.parseInt(args[1]);^M
^M
buffer = new byte[10];
buffer[0]= 1;
buffer[1]= 2;
buffer[2]= 3;
buffer[3]= 4;
buffer[4]= 5;
buffer[5]= 6;
^M
// create packet addressed to destination^M
packet= new DatagramPacket(buffer, buffer.length, ^M
                                address, port);^M
^M
// create socket and send packet^M
socket= new DatagramSocket();^M
socket.send(packet);^M
^M
System.out.println("Program end");^M
}^M

```

Figure 7: Modification to SenderProcess.java which fills a byte array that is 10 bytes long with numbers from 1 to 6 and sends them to the ReceiverProcess.

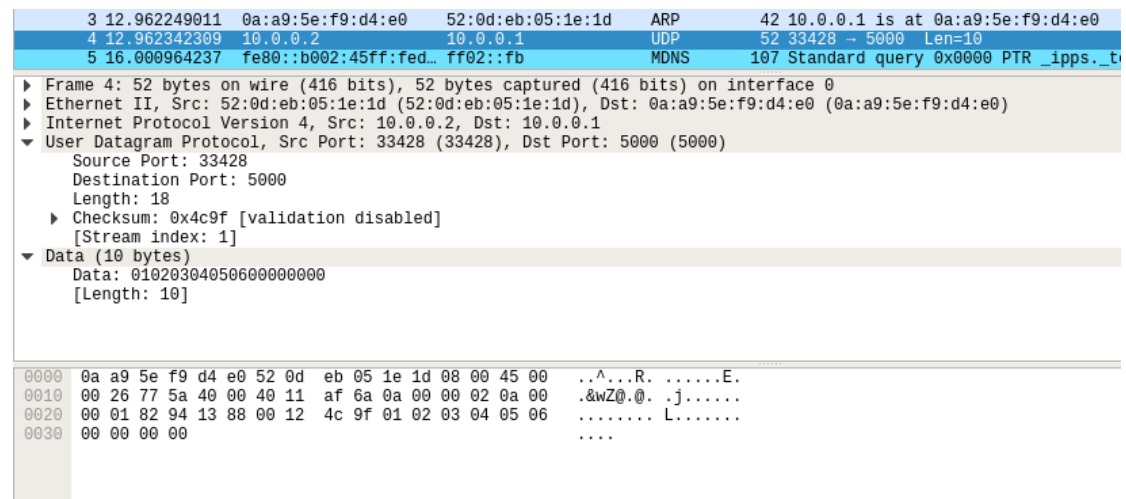


Figure 8: The captured packet in Wireshark shows the payload of the packet as being 10 bytes long and the numbers 1 to 6, followed by 4 0s.

- [1] <http://mininet.org>
- [2] <https://www.virtualbox.org/>
- [3] <http://www.vmware.com/>
- [4] <https://www.wireshark.org/>