

Reader Telecommunication Networks

Part of EE2T21

Niels van Adrichem and Fernando Kuipers

April, 2016

Telecommunication Networks course set-up

The study material for this course comprises chapters 1-7.5 of the book *Data Communications Networking* by Piet Van Mieghem.

The class schedule is as follows:

- 21/04: Explanatory class & chapter 1 - Introduction to Telecommunication Networks (*Homework to be completed before the class: Install Mininet as explained in this reader, read chapter 1*).
- 28/04: Chapter 2 – Local area networking (*Homework to be completed before the class: Complete exercise 1 and read chapter 2*).
- 12/05: Chapter 4 – Architectural principles of the Internet (*Homework to be completed before the class: Complete exercise 2 and read chapter 4*).
- 19/05: Chapter 3 – Error control and retransmission protocols (*Homework to be completed before the class: Complete exercise 3 and read chapter 3*).
- 25/05: Exam on chapters 1, 2, and 4 (possibly also chapter 3).
- 26/05: Chapter 5 – Flow control in the Internet: TCP (*Homework to be completed before the class: No exercise, read chapter 5*).
- 02/06: Chapter 6 – Routing algorithms (*Homework to be completed before the class: Complete exercise 4 and read chapter 6*).
- 09/06: Chapter 7 – Routing protocols (*Homework to be completed before the class: Complete exercise 5 and read chapter 7 up to, and including, section 7.5*).
- T.B.A.: Q & A Session
- 29/06: Exam on chapters 3, 5, 6 and 7-7.5
- 25/07: Re-exam on chapters 1-7.5

Hence, in total there will be 7 classes, 1 Q&A session and ample of opportunity to get and give feedback online via FeedbackFruits (<https://secure.feedbackfruits.com/#groups/81105>). The material, exercises and the examination have been set up in such a way that the course can be passed via self-study, although we do encourage you to come to the classes. All material will be in English so that (foreign) MSc students can participate in this course as part of their homologation program. Along the classes we will be doing exercises that require a (decent) laptop, so please bring your laptop with you.

During the course, we will be providing 5 exercises. By completing them correctly (i.e., answering the Qx.x questions), you can earn 1 point bonus for the final grade of the Telecommunication Networks part of EE2T21. You can also earn a bonus by providing me with additional exercises. However, in total, only a maximum bonus of 1 can be earned. The bonus counts for 100% for the regular exam, for 50% for the re-exam, and for 0% thereafter.

Solutions to the exercises will be discussed at the beginning of each lecture. When time permits, the lectures will close with an opportunity to work on the next exercise. At this time, as well as during the lecture breaks and anytime using FeedbackFruits, questions regarding the exercises can be asked.

The examination will take place in two parts. Both parts will consist of a written closed book examination. You should be able to explain the various networking concepts: what? why? relation

with others? The material itself is not difficult, but it is a lot, and if not properly studied, you may run the risk of mixing up the concepts or being unable to explain them.

The material covered by the examination comprises Chapters 1-7.5 from the book “Data Communications Networking, ISBN 978-94-91075-01-8”, except for the following content:

- Appendices
- footnote-sized text
- Section 2.4.2 (pp. 42 – 43)
- Section 2.6 (pp. 46 – 50)
- Section 4.1 (pp. 73 – 81)
- Section 4.6.4 (pp. 103 – 104)
- Fig. 5.7 on p. 118
- Sec. 5.6 (pp. 122 – 124)
- Sec 6.2.4 (pp. 145 – 146)
- Sec. 6.3.3, 6.3.4 and 6.3.5 (pp. 157 – 167)
- Sec. 6.5, 6.6 and 6.7 (pp. 170 – 176)
- Sec. 7.6 and further.
- This reader

Mininet installation

Mininet is a network simulator. We will use Mininet in our classroom exercises. In order to do this, we advise you to download and install Mininet VM, a Virtual Machine containing a preinstalled version of Mininet. Please do this before the first class.

Install the open-source VirtualBox and download the Mininet VM image from here:

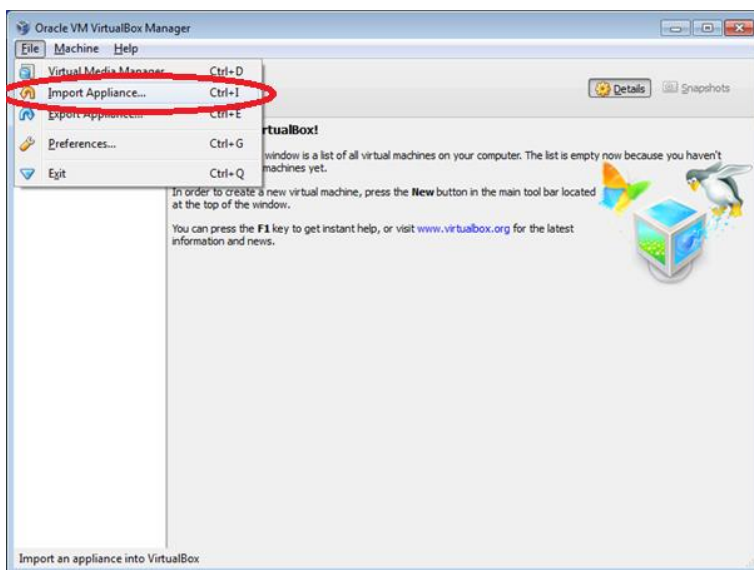
<http://nas.ewi.tudelft.nl/tmp/Mininet-VM.ova>

If you do not have access to a personal laptop or computer to run the Virtualization Server, contact one of the Teaching Assistants to arrange access to our lab.

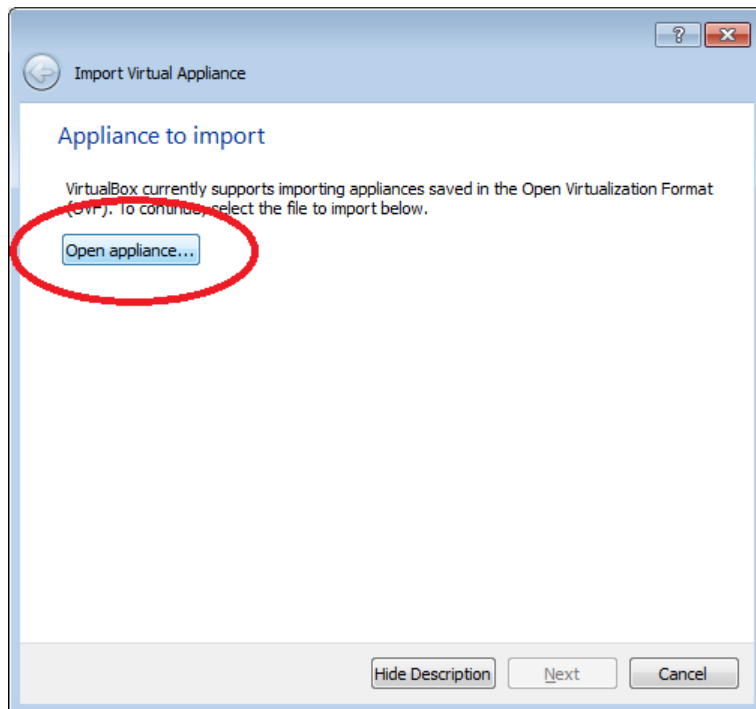
The Mininet and VirtualBox websites and communities provide ample information on installation issues. *We will not provide installation support*, only support for the exercises that are provided by us. The following installation instructions are therefore only meant as a guideline and, in case of problems, you are advised to closely follow the directions on the Mininet website (which are kept up-to-date).

After downloading the image, start VirtualBox and import the VM by executing the following steps:

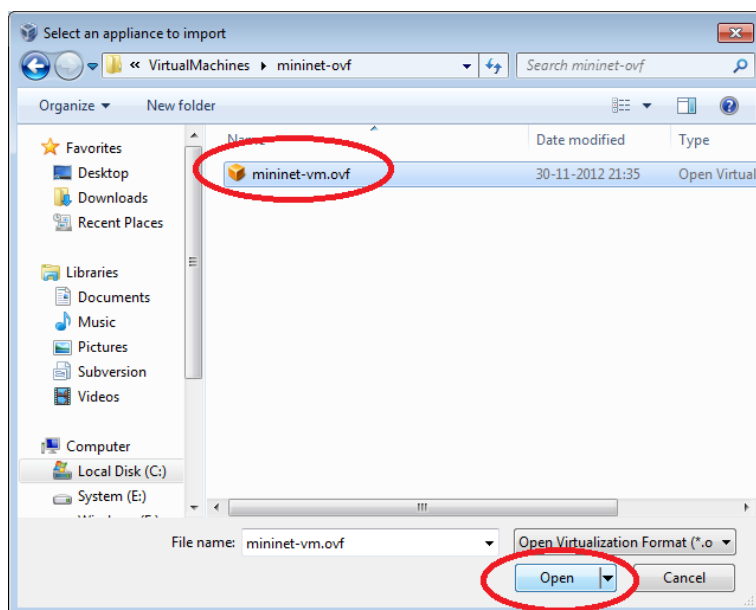
Import the VM by opening *File -> Import Appliance...* from the menu.



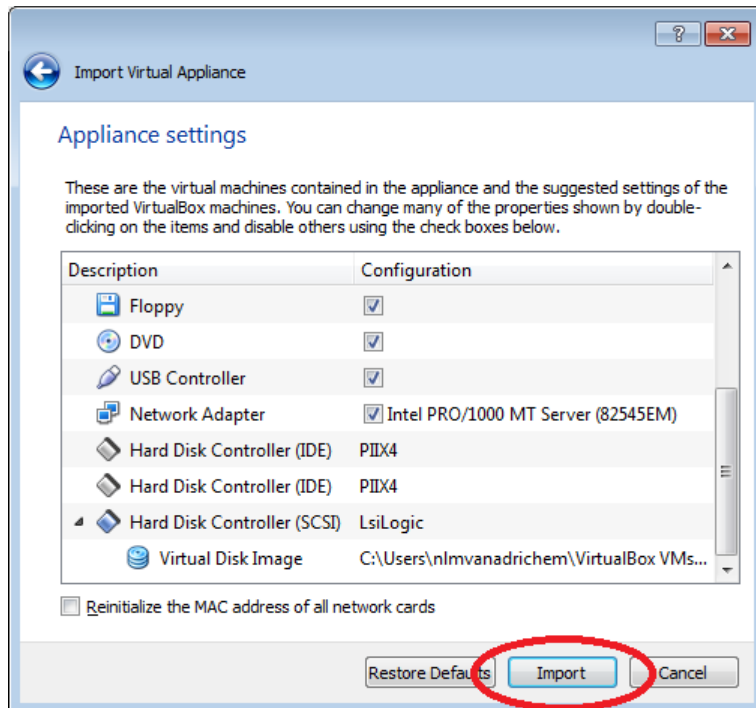
In the new opened dialog press *Open appliance...*



Select and open the file *Mininet-vm.ova* from the directory where you stored the Mininet VM.

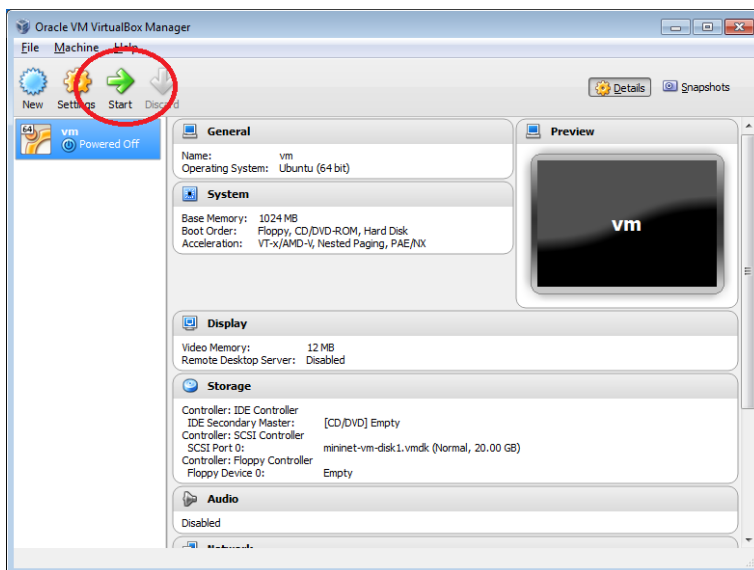


Check if your pc has sufficient RAM. If not, lower the amount of reserved RAM to 512 or 256 MB. Further, accept the default settings and press *Import*.

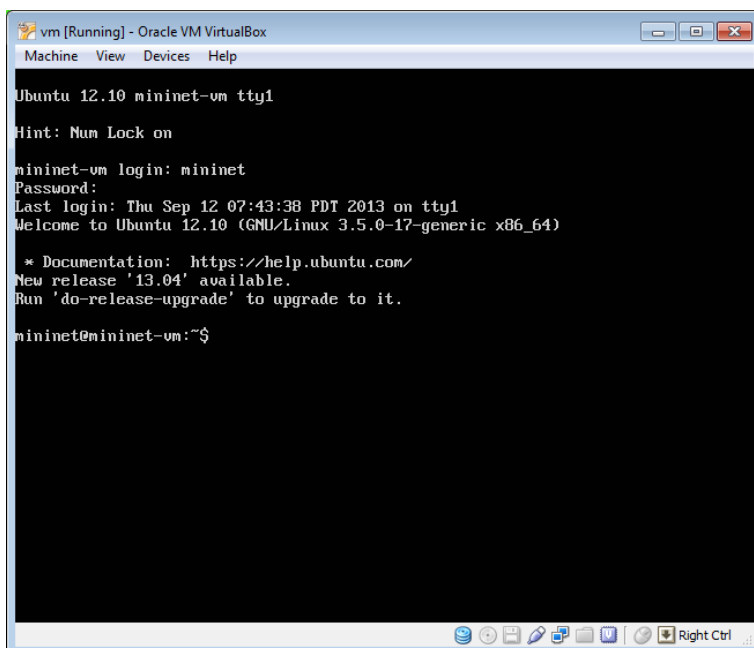


The VM is now being imported. This may take a couple of minutes, depending on the speed of your computer.

When finished, the newly imported VM is shown in VirtualBox. Start it by pressing Start. At this point, ensure that virtualization is enabled in your computer's BIOS or UEFI settings (especially if it does not start).



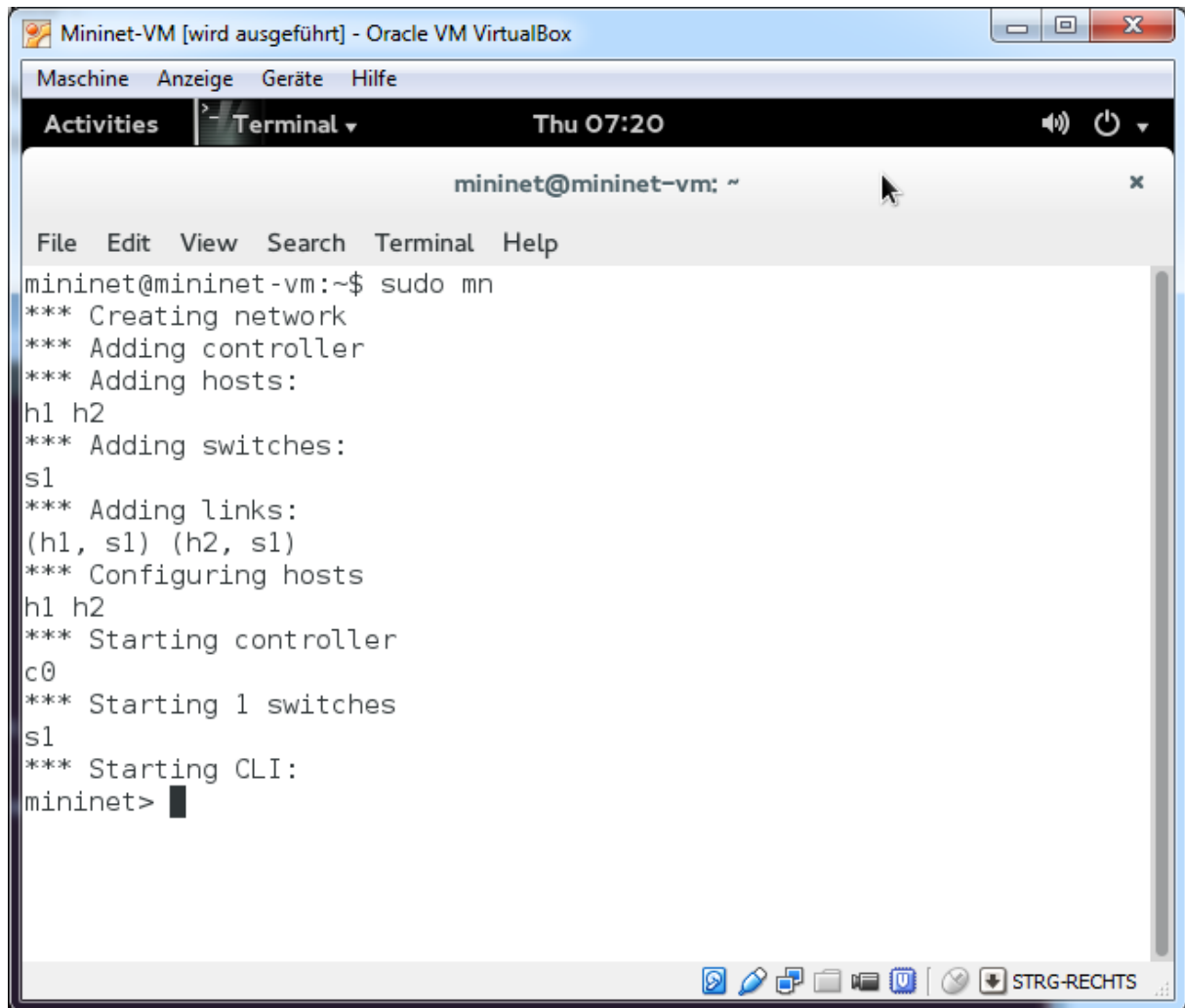
Once started, log in using the default username / password combination *mininet* / *mininet*.



Start the Desktop with the command 'startx' and open a terminal using the key combination *Ctrl+Alt+T*. The linux terminal is comparable to the Windows Command Prompt and offers a Command Line Interface in which commands can be executed.

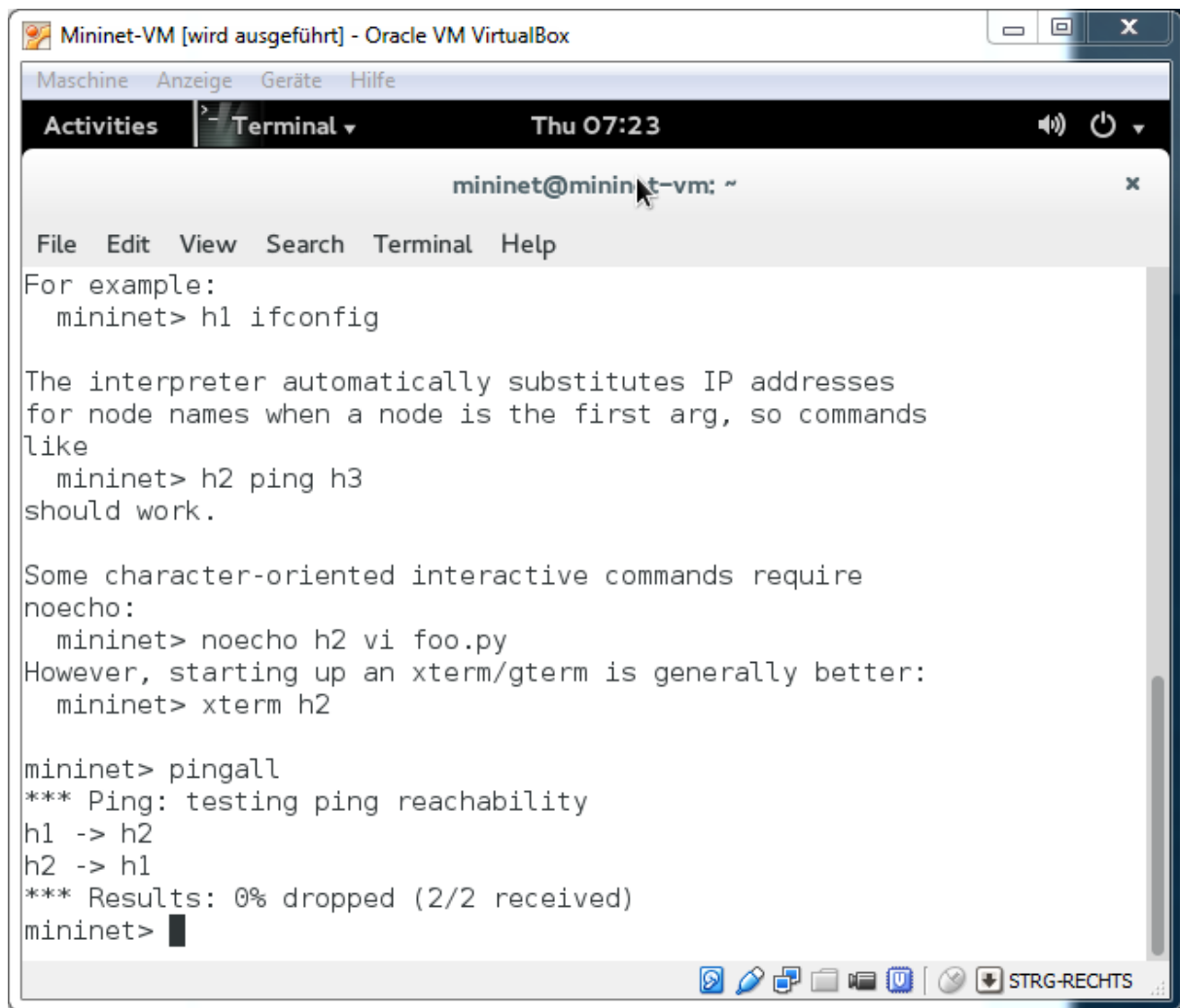
If 'startx' does not open a desktop environment, reconfirm that virtualization is enabled in your computer's BIOS or UEFI settings.

In this terminal, start mininet by running *sudo mn*. By default, Mininet will start a virtual network with 2 hosts *h1* and *h2* connected via 1 switch *s1*.

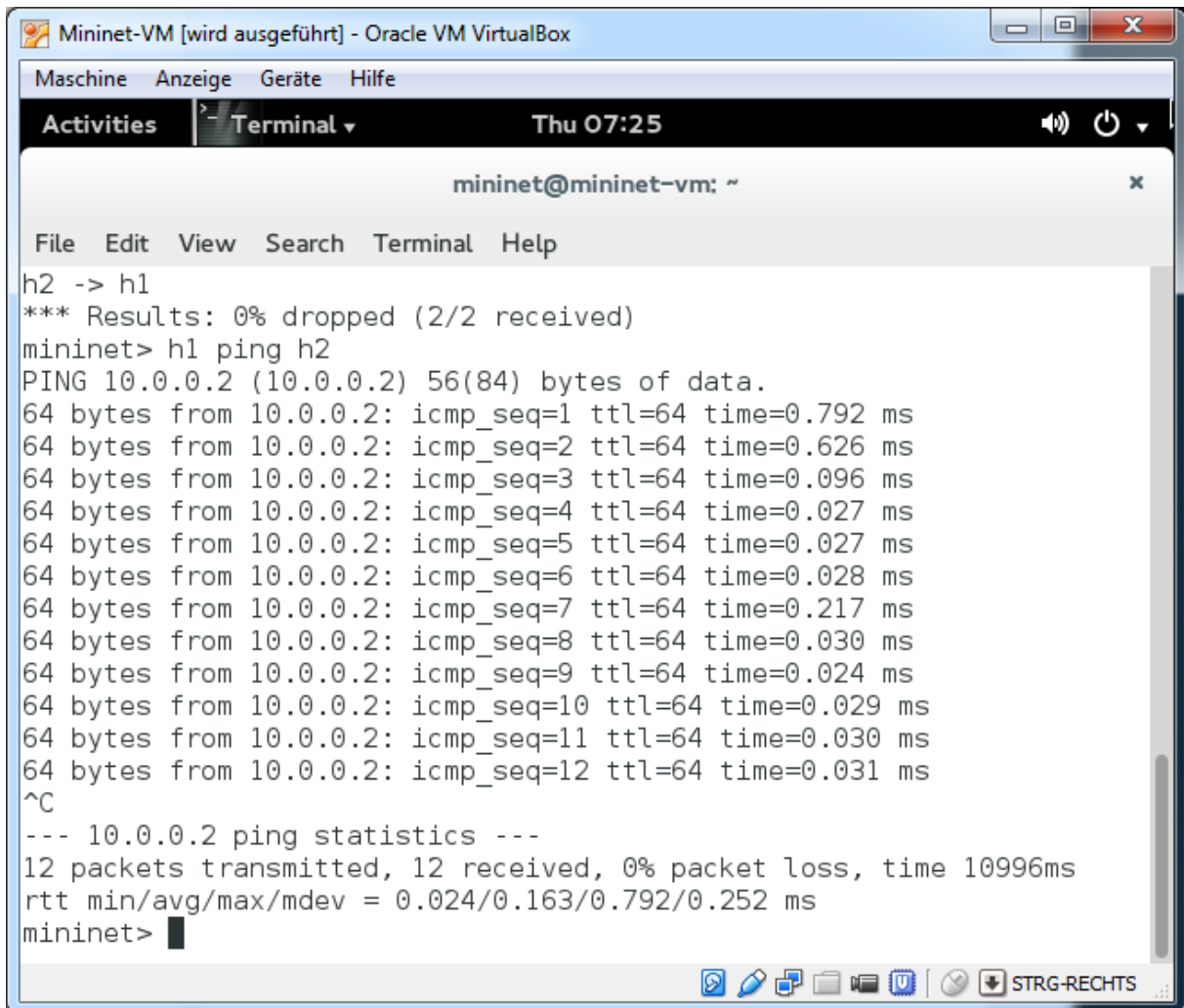


```
Mininet-VM [wird ausgeführt] - Oracle VM VirtualBox
Maschine Anzeige Geräte Hilfe
Activities Terminal Thu 07:20
mininet@mininet-vm: ~
File Edit View Search Terminal Help
mininet@mininet-vm:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1
*** Starting CLI:
mininet>
```

Run *?* to view all possible commands. Feel free to play with them, e.g. run *pingall* to confirm connectivity between all hosts.



You can run most linux commands directly on any of the virtual hosts within Mininet by prepending it by its hostname. For example, you can run ping h2 from h1 by running *h1 ping h2*. You can cancel the ping by pressing *Ctrl+C*.



```
Mininet-VM [wird ausgeführt] - Oracle VM VirtualBox
Maschine Anzeige Geräte Hilfe
Activities Terminal Thu 07:25
mininet@mininet-vm: ~
File Edit View Search Terminal Help
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> h1 ping 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=0.792 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.626 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.096 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.027 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.027 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.028 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.217 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.030 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.024 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.029 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.030 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.031 ms
^C
--- 10.0.0.2 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 10996ms
rtt min/avg/max/mdev = 0.024/0.163/0.792/0.252 ms
mininet>
```

Congratulations: you have now successfully installed Mininet!

Exercise 1: Practice with Mininet (Hand-in before 28/04)

Creating topologies

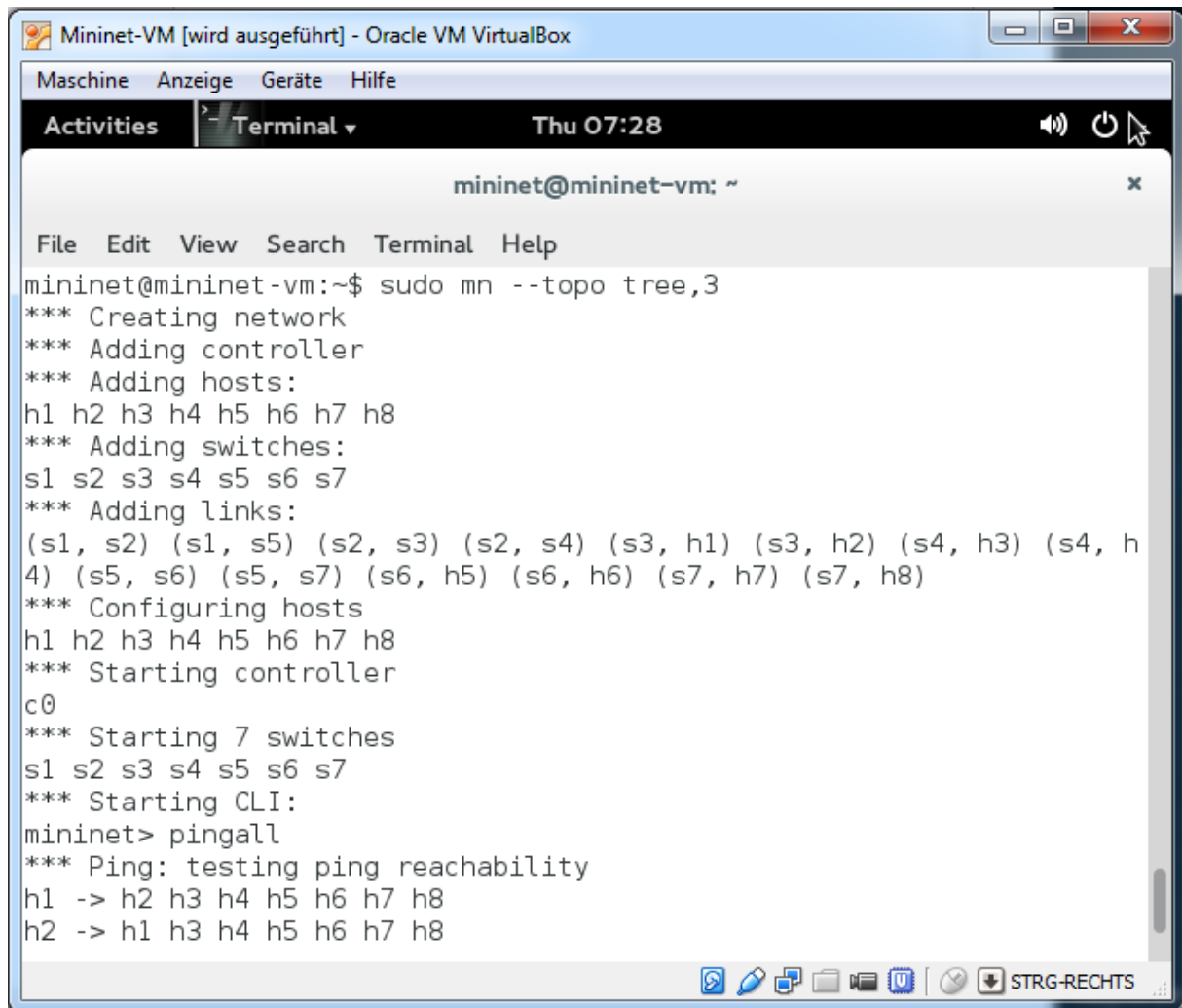
One of the advantages of Mininet is that you can create complex network topologies and run experiments on them without the need to physically create those networks. There is a GUI for creating networks, called Miniedit, which is available in the Mininet VM under `mininet/examples` and can be run with the command `python miniedit.py`. For now, we will consider a topology to consist of only hosts and switches. You may consider a host to be a computer that is generating or receiving data and a switch to be an intersection where data arriving on one entry/exit point (port) is directed to another port depending on its intended destination.

Mininet itself comes with the following built-in topologies:

- Minimal: The default topology of 1 switch with 2 hosts. No further parameters apply.
- Single: A single switch, h hosts topology. This topology can be used by appending `--topo=single,h'` to the `mn` command where h refers to the number of hosts.
- Reversed: Equal to the single switch topology, except that hosts connect to the switch in reverse order (i.e. the highest host number gets the lowest switch port).
- Linear: One switch for each host, switches are connected in a line. To use this topology, append `--topo linear,h'` to the `mn` command.
- Tree: A binary tree topology of depth d . To use this topology, append `--topo tree,d'` to the `mn` command.

For instance, to create a binary tree topology of depth 3:

- Open the terminal using `Ctrl+Alt+T` and run `sudo mn --topo tree,3`
- Wait for Mininet to start, create the hosts, switches and corresponding links and run `pingall` to confirm connectivity between all nodes

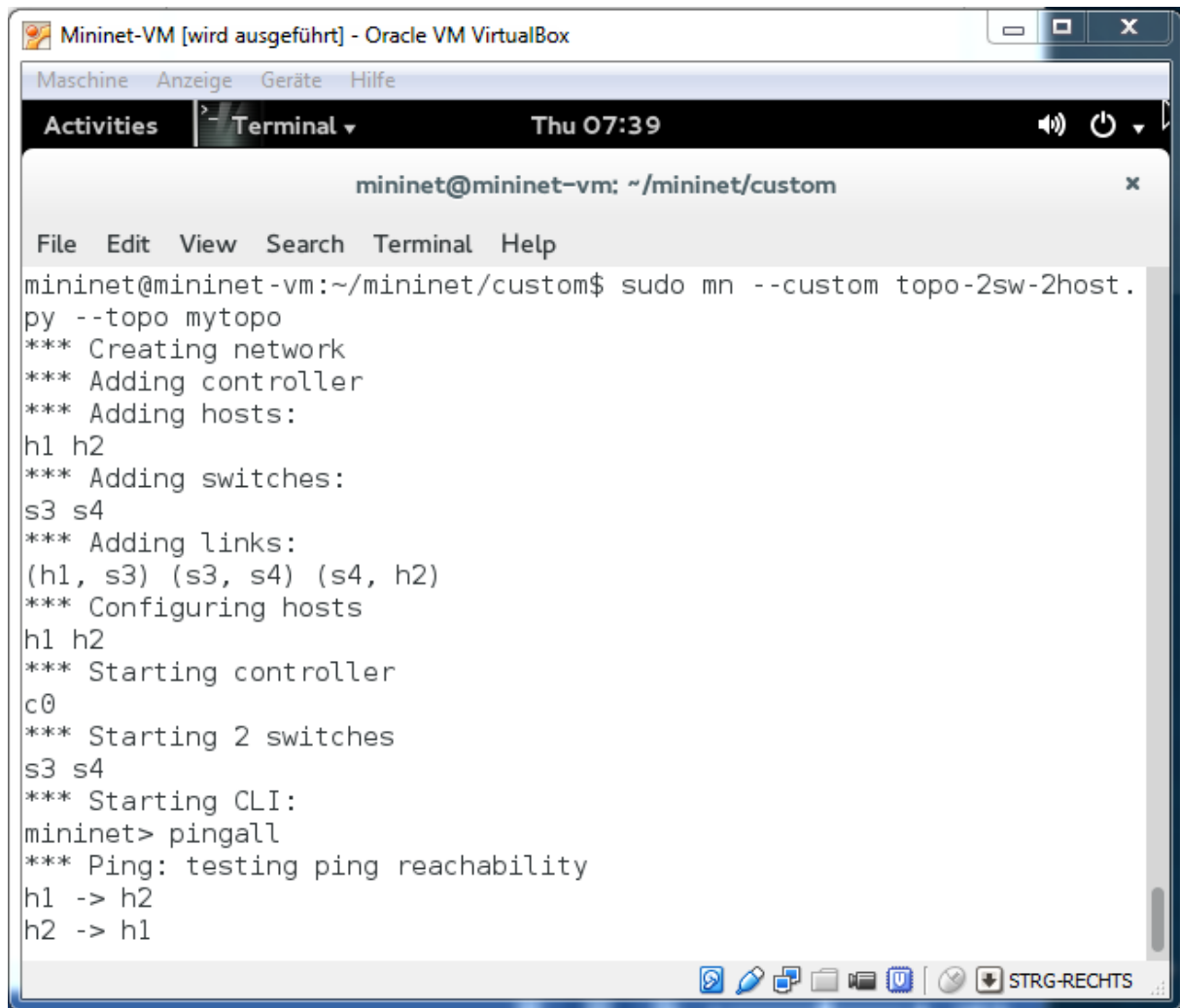


```
Mininet-VM [wird ausgeführt] - Oracle VM VirtualBox
Maschine Anzeige Geräte Hilfe
Activities Terminal Thu 07:28
mininet@mininet-vm: ~
File Edit View Search Terminal Help
mininet@mininet-vm:~$ sudo mn --topo tree,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h
4) (s5, s6) (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
```

In order to implement a custom topology, you can use Miniedit, but to understand the topology file, we will modify it by hand. Your working directory contains a folder mininet that contains a folder custom containing a sample custom topology file *topo-2sw-2host.py*. You can store your topologies there.

CD into this directory, read its README file and inspect the sample file *topo-2sw-2host.py* using either *gedit* or your favorite text editor.

- `cd ~/mininet/custom`
- `gedit README &`
- `gedit topo-2sw-2host.py &`
- `sudo mn --custom topo-2sw-2host.py --topo mytopo` (--custom indicates that a custom topology file will be used as mytopo)
- `pingall`



```
Mininet-VM [wird ausgeführt] - Oracle VM VirtualBox
Maschine Anzeige Geräte Hilfe
Activities Terminal Thu 07:39
mininet@mininet-vm: ~/mininet/custom
File Edit View Search Terminal Help
mininet@mininet-vm:~/mininet/custom$ sudo mn --custom topo-2sw-2host.
py --topo mytopo
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s3 s4
*** Adding links:
(h1, s3) (s3, s4) (s4, h2)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 2 switches
s3 s4
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
```

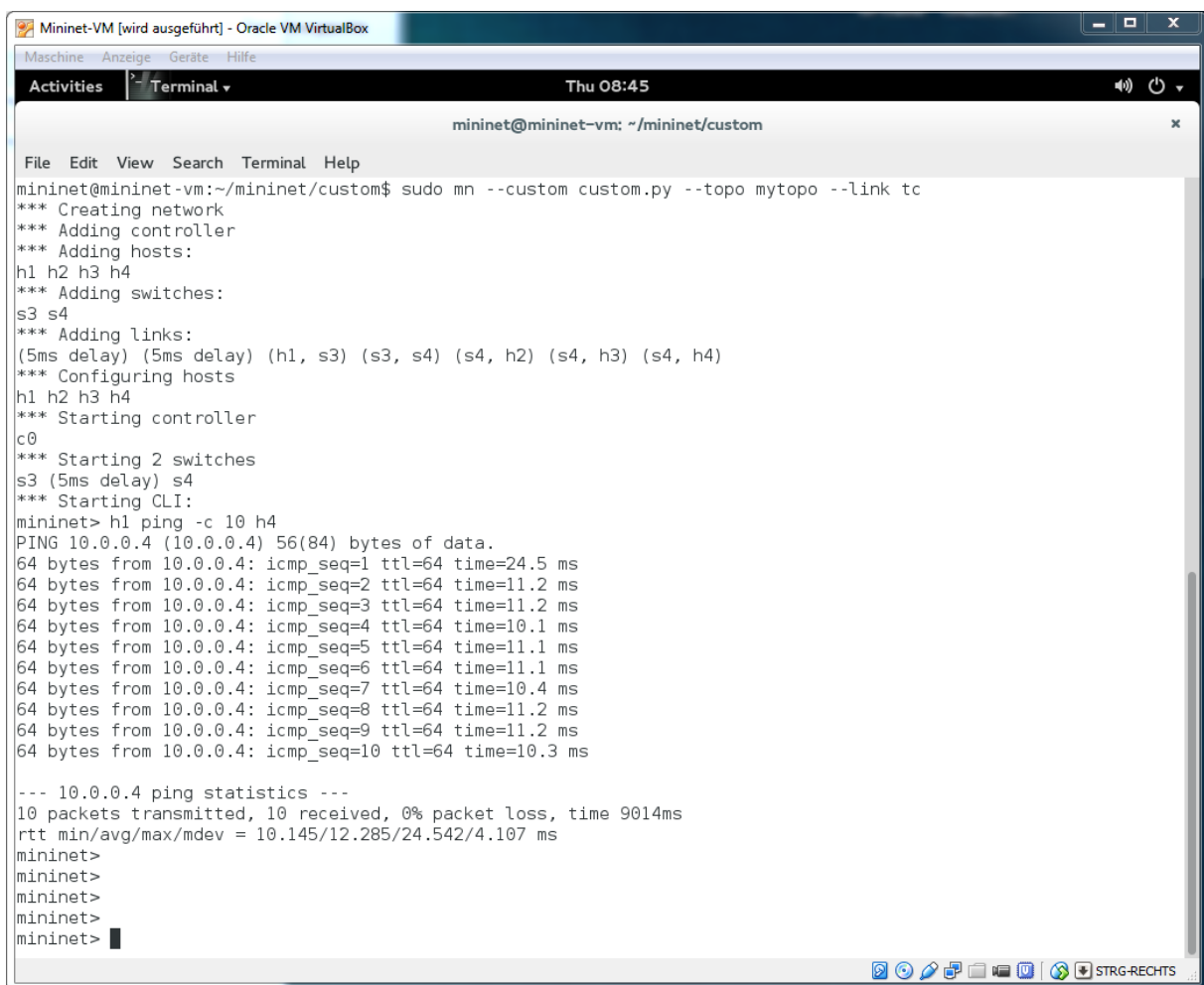
After inspecting the file topo-2sw-2host.py:

1. Identify which lines are responsible for
 - a. Adding hosts
 - b. Adding switches
 - c. Adding links
2. Copy the file topo-2sw-2host.py to a new file custom.py. Alter the topology to contain 4 hosts, connected evenly distributed over the 2 switches. Restart Mininet and confirm connectivity between all nodes.

Introducing link properties

In order to simulate real networks, one can apply link metrics such as bandwidth [*bw*], delay [*delay*] and packet loss [*loss*]. In order to simulate a long-delay connection between the two switches from the previous assignment, add the following lines to your custom topology file.

- Right before or after `from mininet.topo import Topo` add a line `from mininet.link import TCLink`. The TCLink class offers link specific performance parameters.
- Change the link between your 2 switches from `self.addLink(leftSwitch, rightSwitch)` to `self.addLink(leftSwitch, rightSwitch, delay='5ms'` to add 5 ms delay to the inter-switch link.
- Add the parameters `--link tc` to the Mininet command to inform Mininet about the usage of the TCLink class and perform 10 pings from a host on the left switch to a host on the right switch to confirm the configured delay.



```
Mininet-VM [wird ausgeführt] - Oracle VM VirtualBox
Maschine Anzeige Geräte Hilfe
Activities Terminal Thu 08:45
mininet@mininet-vm: ~/mininet/custom

File Edit View Search Terminal Help
mininet@mininet-vm:~/mininet/custom$ sudo mn --custom custom.py --topo mytopo --link tc
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s3 s4
*** Adding links:
(5ms delay) (5ms delay) (h1, s3) (s3, s4) (s4, h2) (s4, h3) (s4, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s3 (5ms delay) s4
*** Starting CLI:
mininet> h1 ping -c 10 h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=24.5 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=11.2 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=11.2 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=10.1 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=11.1 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=11.1 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=10.4 ms
64 bytes from 10.0.0.4: icmp_seq=8 ttl=64 time=11.2 ms
64 bytes from 10.0.0.4: icmp_seq=9 ttl=64 time=11.2 ms
64 bytes from 10.0.0.4: icmp_seq=10 ttl=64 time=10.3 ms

--- 10.0.0.4 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 10.145/12.285/24.542/4.107 ms
mininet>
mininet>
mininet>
mininet>
mininet>
```

1. Confirm that the delay is actually taken into account. Where the delay previously was around tenths of milliseconds, you should now see a much larger delay.

Q1.1 Comment upon the times measured by ping: Why is it not around 5 ms?

2. Include bandwidth [*bw*] limitations in the network:
 - a. Limit the links between hosts and switches to 100 Mbps
 - b. Limit the bandwidth of the inter-switch link to 10 Mbps, and add a 1 % packet loss [*loss*] (which is very high for real networks).

- c. *Q1.2: Write down how you implemented these properties.*
- d. Mininet is shipped with iperf, a popular traffic generator that is used to test network paths and tries to use as much bandwidth as it can get. Confirm the new bandwidth using iperf measurements (f.e. by running an experiment from and to hosts on the same switch, as well as from and to hosts connected to different switches).

Q1.3: write down your iperf results.

Exercise 2: Wireshark and Local Area Networking (Hand-in before 12/05)

In this exercise, we will be using Wireshark. Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible. You could think of a network packet analyzer as a measuring device used to examine what's going on inside a network cable, just like a voltmeter is used by an electrician to examine what's going on inside an electric cable (but at a higher level, of course).

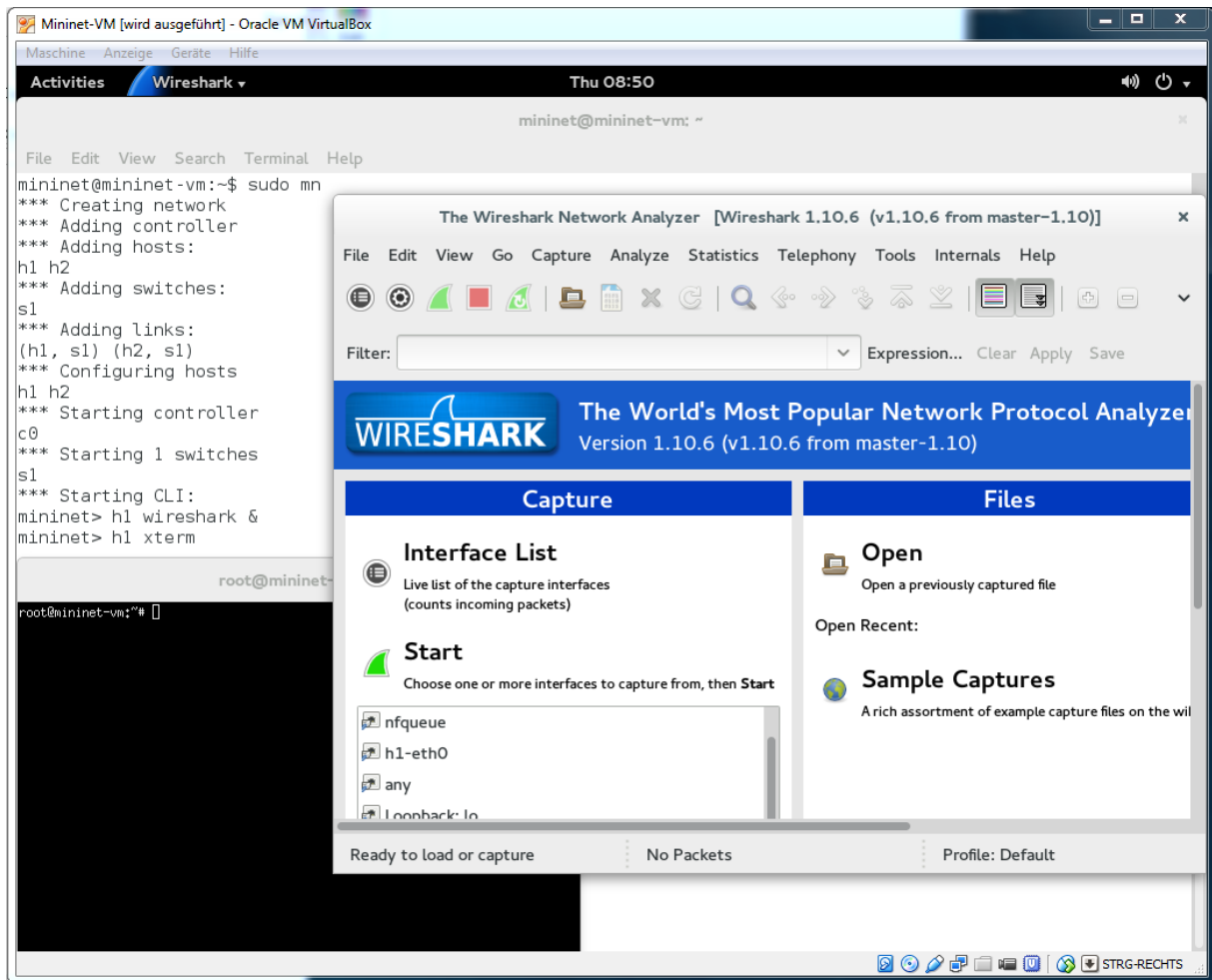
The Wireshark program and its documentation (from which the above description was retrieved) can be obtained at: <http://www.wireshark.org/>. Fortunately, in our case, the Wireshark program is already included in your Mininet installation.

To practice with Wireshark, we are going to try to capture FTP traffic. FTP stands for File Transport Protocol, and it is an application to exchange files. Essentially FTP is a client-server application that uses two ports on both the client and the server. One port is used to exchange FTP control or command information and the other port is used to transfer the data. A port is a Service Access Point (SAP) address. On the FTP server, port 20 is used for data and port 21 for control. The FTP client can use any ports larger than 1023 (the others are used for server applications). An FTP exchange of information consists of requests sent by the client and responses sent by the server.

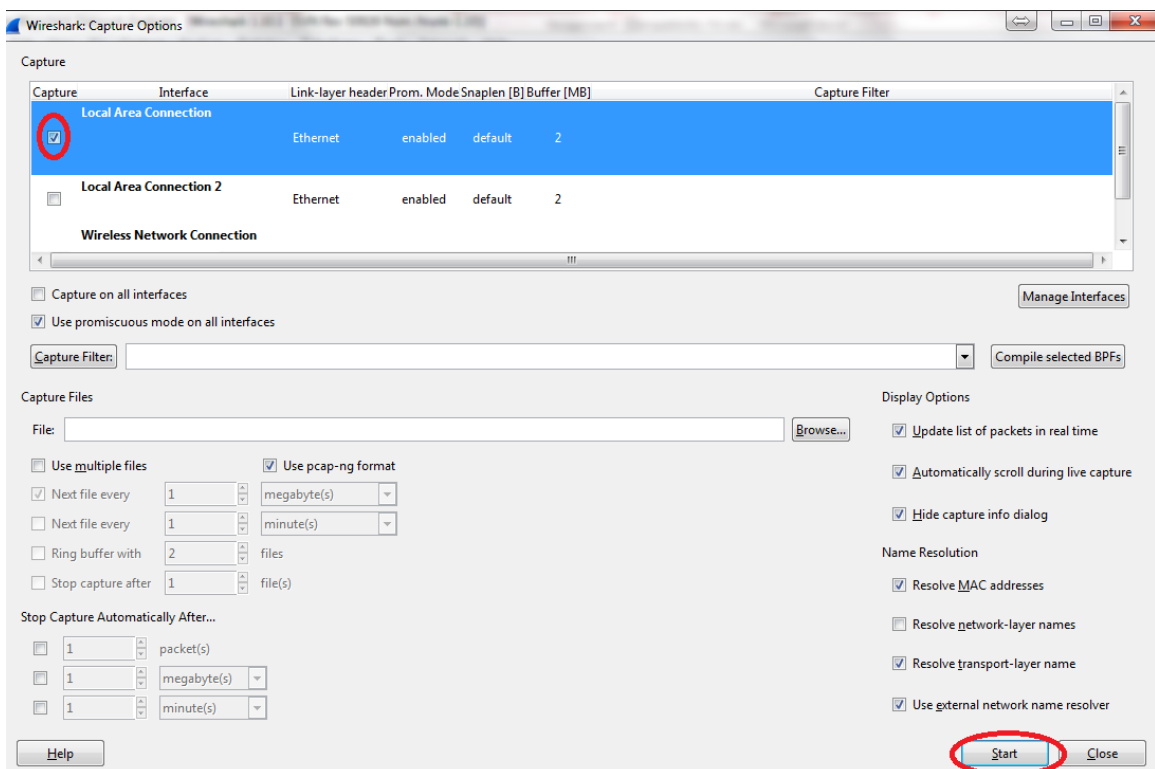
To use FTP in Mininet, we will have to do the following:

- Within the Mininet VM go to ("cd") /tmp
- Create a sample file using "fallocate -l 1MiB ftpfile"
- Install an FTP server using "sudo apt-get install ftpd"
- Start-up Mininet "sudo mn" and open a terminal on each host using "h1 xterm &" and "h2 xterm &"
- In the h2 xterm window, start the FTP server using the command "inetd"
- Run a Wireshark instance on host h2 by running *h2 wireshark &* from the Mininet console

The following screen will appear:



In Wireshark, click on "Start a capture with detailed options". The following screen will appear:



Choose the interface on which you want to capture (sometimes you will have multiple interfaces, for instance a wired and a wireless one) and press Start.

In Wireshark, type in “ftp” (without quotation marks) next to the Filter:



In the h1 xterm window start an FTP session using the command "ftp -d 10.0.0.2" as follows:

Name: mininet

Password: mininet

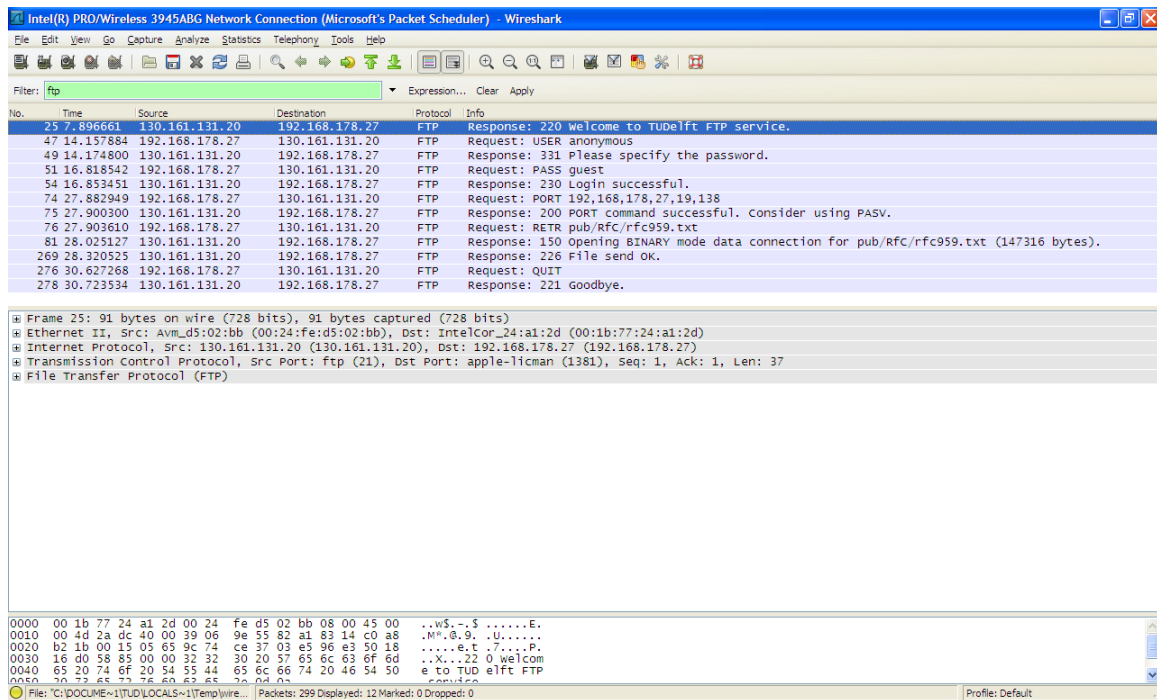
```
ftp> cd /tmp
```

```
ftp> get ftpfile
```

“help” gives commands, “get” is used to download a file, and “bye” closes and exits ftp.

FTP uses a Client - Server transaction model; the FTP client sends a request and the FTP server sends one or more responses.

Exit by typing “bye” and subsequently stop the Wireshark capture. You should see something like:



There are two filters in Wireshark, a Capture filter and a Display filter. The Capture filter is found underneath Capture, Options. The Display filter is in Wireshark's main window, in the Filter toolbar. The Capture filter and Display filter have a different syntax.

You see a list of the packets that were captured (and filtered based on “ftp” with the display filter). If you click on one of those packets you will see its details below and you will recognize a layered structure.

Q2.1: Explain what layering is and why it is useful: give at least two advantages and one disadvantage of the layering concept.

Wireshark captures data link layer frames and can look inside those frames. In the example above you see that in this case an Ethernet frame carries an IP packet inside, which on its turn has TCP information for the FTP application. By clicking on the “+” you will see more information.

Q2.2: Was the username and password visible in the trace?

We will continue to use Wireshark to examine traffic on the data link (Layer 2 in OSI).

In promiscuous mode, which is the default Wireshark setting, the trace will include all packets seen, regardless of whether they are addressed to or from your machine. On a LAN this means that you might capture many packets that are not intended for you 😊

Open Wireshark and start a capture that stops after 100 packets have been captured.

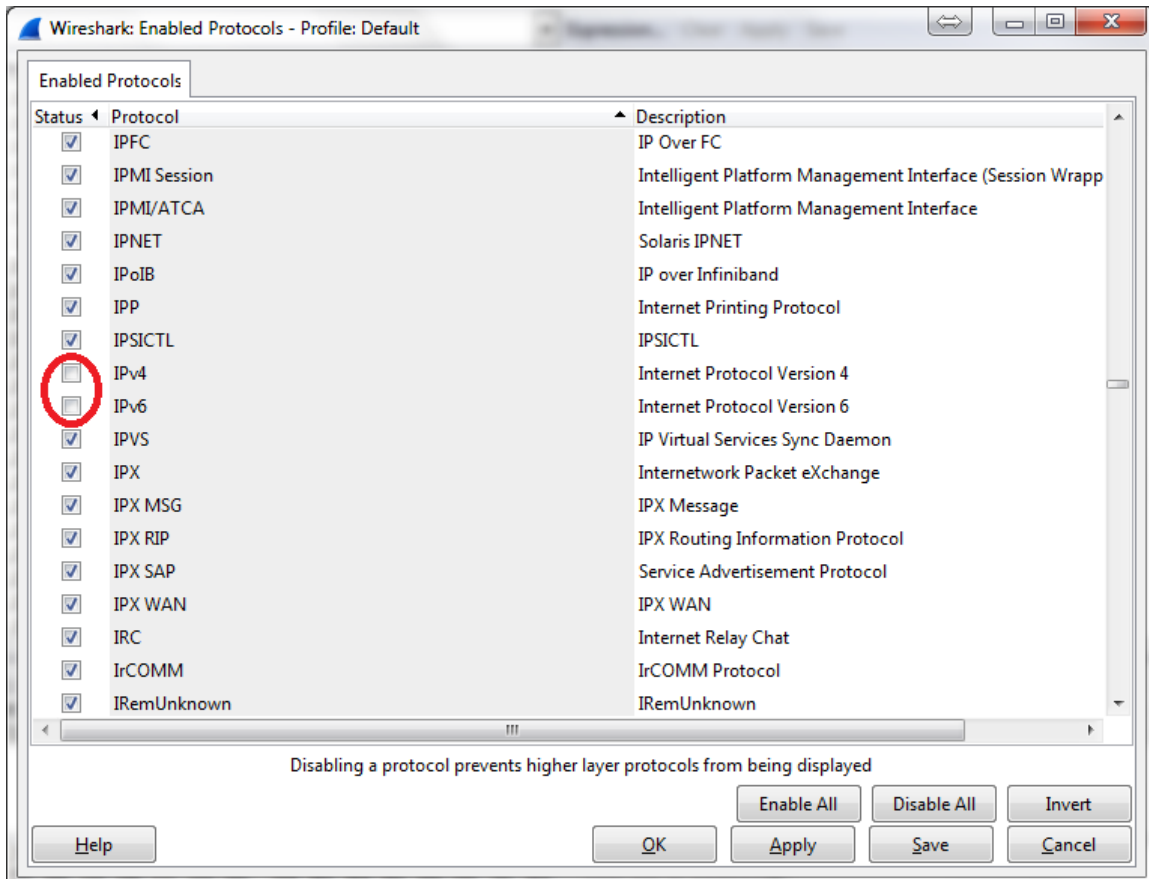
Run ‘iperf h1 h2’ and answer the following questions:

Q2.3: How long (in seconds) was the interval?

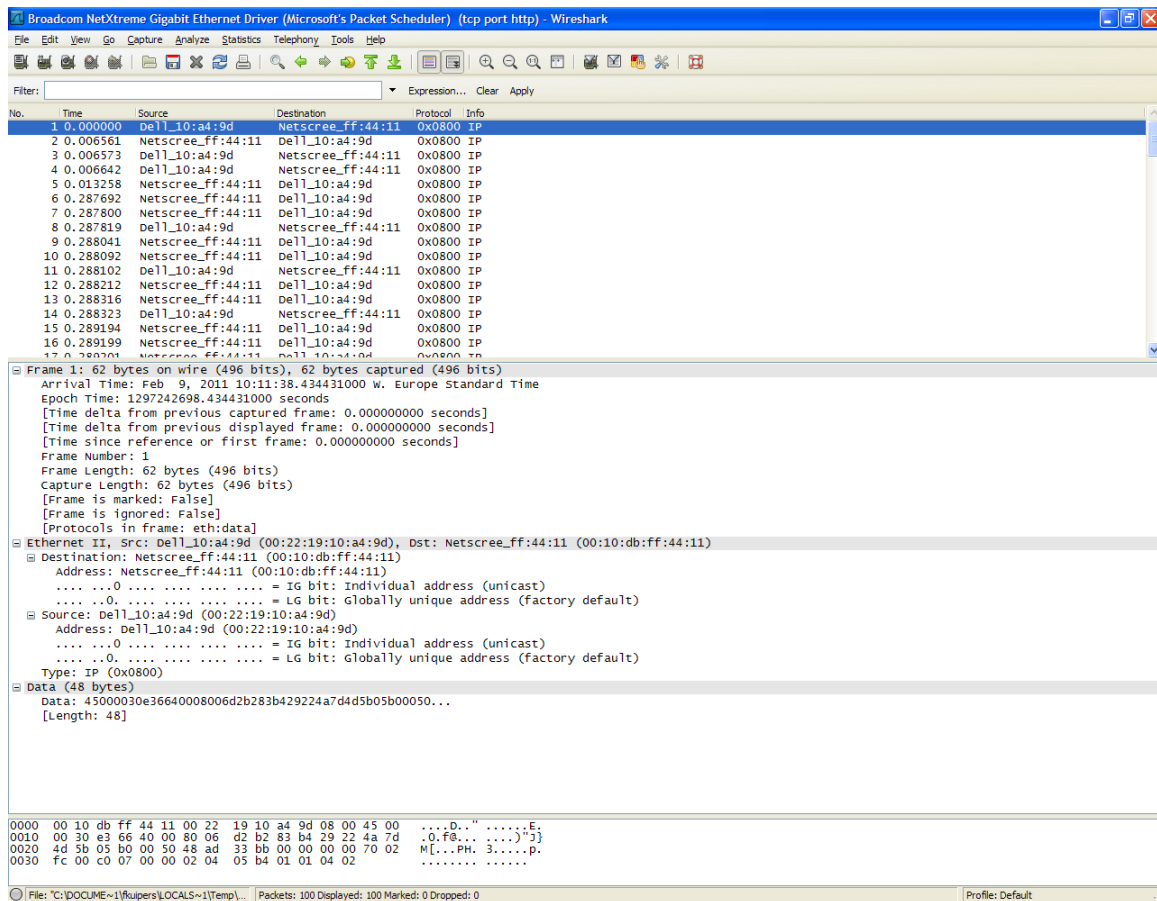
Q2.4: How many data (in bytes) have been transferred?

Q2.5: What is the average throughput between the hosts?

In the following, we are not interested in the higher layer protocols, so we can remove them from the displayed data, by deselecting IPv4 and IPv6 under “Analyze->Enabled Protocols”.



Wireshark will now look like:



An Ethernet II frame is composed of the following fields:

Field	Length (bytes)	Description
Preamble	8	Not shown in capture. This field contains synchronizing bits, processed by the NIC hardware.
Destination address	6	<p>Layer 2 addresses for the frame. Each address is 48 bits long, or 6 bytes, expressed as 12 hexadecimal digits, 0-9,A-F. A common format is 12:34:56:78:9A:BC.</p> <p>The first six hex numbers indicate the manufacturer of the network interface card (NIC). The last six hex digits are the serial number of the NIC.</p> <p>The destination address may be a broadcast, which contains all 1s or unicast. The source address is always unicast.</p>
Source address	6	
Frame type	2	For Ethernet II frames, this field contains a

		<p>hexadecimal value that is used to indicate the type of upper layer protocol in the data field. There are numerous upper layer protocols supported by Ethernet II. Two common frame types are:</p> <p>0x0800 IPv4 Protocol</p> <p>0x0806 Address Resolution Protocol (ARP)</p>
Data	46-1500	Contains the encapsulated upper level protocol. The data field is between 46 and 1500 bytes.
FCS	4	Not shown in capture. Frame Check Sequence, used by the NIC to identify errors during transmission. The value is computed by the sending machine, encompassing frame addresses, type, and data field. It is verified by the receiver.

Answer the following questions:

Q2.6: What is the Ethernet address of h1?

Q2.7: To what does the destination address in the Ethernet frame refer?

Consider the first captured frame. Click on the packet contents (the bottom window in Wireshark) and try to identify, by clicking on it, the source address, the destination address, and the data.

Exercise 3: IP Routing and Forwarding (Hand-in before 19/05)

Up till now, we have only used Mininet to simulate Local Area Networks or LANs, meaning small networks solely consisting of OSI Layer 2 / Ethernet interconnects using switches. Ultimately, Mininet can also be used to simulate networks containing routers, Layer 3 / IP interconnects in which packets are forwarded across subnetworks by inspection of the IP address rather than the Ethernet address. In this exercise re-enable IP decapsulation in the displayed data, by activating IPv4 and IPv6 under “Analyze->Enabled Protocols”.

Confirm forwarding is enabled on your host (Mininet VM) by running ``sysctl net.ipv4.ip_forward`` or ``cat /proc/sys/net/ipv4/ip_forward``, the result should equal 1. If this is not the case, enable forwarding by running the command ``sudo sysctl -w net.ipv4.ip_forward=1`` or ``echo 1 > /proc/sys/net/ipv4/ip_forward``.

Make a new copy of the sample file *topo-2sw-2host.py*, call it *custom2.py*. Next, remove the two switches and their respective links, add a new *regular host* named *r1* - which will become our main (simple) router - and connect the two existing hosts to the new host.

A screenshot of a terminal window with a dark background and light-colored text. The window title bar shows 'File Edit View Search Terminal Help' and system icons on the right. The script content is as follows:

```
"""Custom topology example

Two directly connected switches plus a host for each switch:

   host --- switch --- switch --- host

Adding the 'topos' dict with a key/value pair to generate our newly defined
topology enables one to pass in '--topo=mytopo' from the command line.
"""

from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        #remove line leftSwitch = self.addSwitch( 's3' )
        #remove line rightSwitch = self.addSwitch( 's4' )
        router = self.addHost( 'r1' )

        # Add links
        # remove line self.addLink( leftHost, leftSwitch )
        # remove line self.addLink( leftSwitch, rightSwitch )
        # remove line self.addLink( rightSwitch, rightHost )
        self.addLink( leftHost, router )
        self.addLink( router, rightHost )

topos = { 'mytopo': ( lambda: MyTopo() ) }
~
~
```

The terminal shows line numbers 34, 34-41 on the right side of the code block. The bottom status bar shows system icons and the text 'Right Ctrl'.

Now run your custom topology by running ``sudo mn --custom custom2.py --topo mytopo`` and confirm all-to-all connectivity using *pingall*.

```
@mininet-vm: ~/mininet/custom
mininet@mininet-vm:~/mininet/custom$ sudo mn --custom custom2.py --topo mytopo
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 r1
*** Adding switches:

*** Adding links:
(h1, r1) (h2, r1)
*** Configuring hosts
h1 h2 r1
*** Starting controller
*** Starting 0 switches

*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X r1
h2 -> X X
r1 -> h1 X
*** Results: 66% dropped (4/6 lost)
mininet> 
```

As you might notice, *pingall* is unsuccessful at reaching all nodes, due to the fact that these are not connected to the same Local Area Network anymore. Although Mininet configured all hosts with IP addresses from the same subnet, they, in fact, do not reside within the same broadcast medium anymore. Instead, we will configure IP addresses and subnet masks manually on the 2 LANs. Instead of the private IP address range 10.0.0.0/255.0.0.0 or 10.0.0.0/8, we will use addresses from the private address range 192.168.0.0/16.

We will configure the two (*left* and *right*) subnetworks according to the following addresses:

Left		Right	
Subnet	192.168.1.0	Subnet	192.168.2.0
Netmask	255.255.255.0	Netmask	255.255.255.0
R1	192.168.1.1	R1	192.168.2.1
H1	192.168.1.2	H2	192.168.2.2

We start by configuring the interface of host h1 and the first interface of the router, which are connected to the first/left subnet. We will use the *ifconfig* command to configure the IP address on their primary and only interfaces. Then we confirm connectivity between the host and the router via *ping*.

- r1 ifconfig r1-eth0 192.168.1.1 netmask 255.255.255.0
- h1 ifconfig h1-eth0 192.168.1.2 netmask 255.255.255.0
- h1 ping -c 1 192.168.1.1


```
File Edit View Search Terminal Help
mininet@mininet-vm:~/mininet/custom$ sudo mn --custom custom2.py --topo mytopo
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 r1
*** Adding switches:

*** Adding links:
(h1, r1) (h2, r1)
*** Configuring hosts
h1 h2 r1
*** Starting controller
*** Starting 0 switches

*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X r1
h2 -> X X
r1 -> h1 X
*** Results: 66% dropped (4/6 lost)
mininet> r1 ifconfig r1-eth0 192.168.1.1 netmask 255.255.255.0
mininet> h1 ifconfig h1-eth0 192.168.1.2 netmask 255.255.255.0
mininet> h1 ping -c 1 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=64 time=0.131 ms

--- 192.168.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.131/0.131/0.131/0.000 ms
mininet> 
```

Repeat the same procedure for the `right` subnet. We now select the second interface of the router to be configured. Note that we use an alternative notation for the subnet mask. Furthermore, confirm connectivity from the router to both hosts, and whether both hosts can connect.

- `r1 ifconfig r1-eth1 192.168.2.1/24`
- `h2 ifconfig h2-eth0 192.168.2.2/24`
- `h2 ping -c 1 192.168.2.1`
- `r1 ping -c 1 192.168.1.2`
- `r1 ping -c 1 192.168.2.2`
- `h1 ping -c 1 192.168.2.2`

```
File Edit View Search Terminal Help 7:25 AM
mininet> pingall
*** Ping: testing ping reachability
h1 -> X r1
h2 -> X X
r1 -> h1 X
*** Results: 66% dropped (4/6 lost)
mininet> r1 ifconfig r1-eth0 192.168.1.1 netmask 255.255.255.0
mininet> h1 ifconfig h1-eth0 192.168.1.2 netmask 255.255.255.0
mininet> h1 ping -c 1 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=64 time=0.150 ms

--- 192.168.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.150/0.150/0.150/0.000 ms
mininet> r1 ifconfig r1-eth1 192.168.2.1/24
mininet> h2 ifconfig h2-eth0 192.168.2.2/24
mininet> h2 ping -c 1 192.168.2.1
PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.
64 bytes from 192.168.2.1: icmp_req=1 ttl=64 time=0.133 ms

--- 192.168.2.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.133/0.133/0.133/0.000 ms
mininet> r1 ping -c 1 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_req=1 ttl=64 time=0.127 ms

--- 192.168.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.127/0.127/0.127/0.000 ms
mininet> r1 ping -c 1 192.168.2.2
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_req=1 ttl=64 time=0.121 ms

--- 192.168.2.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.121/0.121/0.121/0.000 ms
mininet> h1 ping -c 1 192.168.2.2
connect: Network is unreachable
mininet>
```

While the router is able to connect to both subnetworks, the hosts in the subnetworks cannot connect. In order for them to connect, we need to tell them whereto to send packets for the other subnet. This is done by filling in a local routing table using the *route* command. In this case, we need to tell host h1 to forward packets designated for the 'right' subnet to the router available at IP address 192.168.1.1. The same holds for h2. Confirm connectivity.

- h1 route add -net 192.168.2.0/24 gw 192.168.1.1
- h2 route add -net 192.168.1.0/24 gw 192.168.2.1
- h1 ping -c 1 192.168.2.2

Answer the following questions:

Q3.1: How do the subnet address and network mask relate to the forwarding rule and the IP address of incoming packets?

Q3.2: How is the right forwarding rule selected for an incoming IP packet?

Q3.3: What is the difference between the two previously installed forwarding rules?

Q3.4: If a packet matches multiple forwarding rules, which one is selected? E.g., is there any particular order?

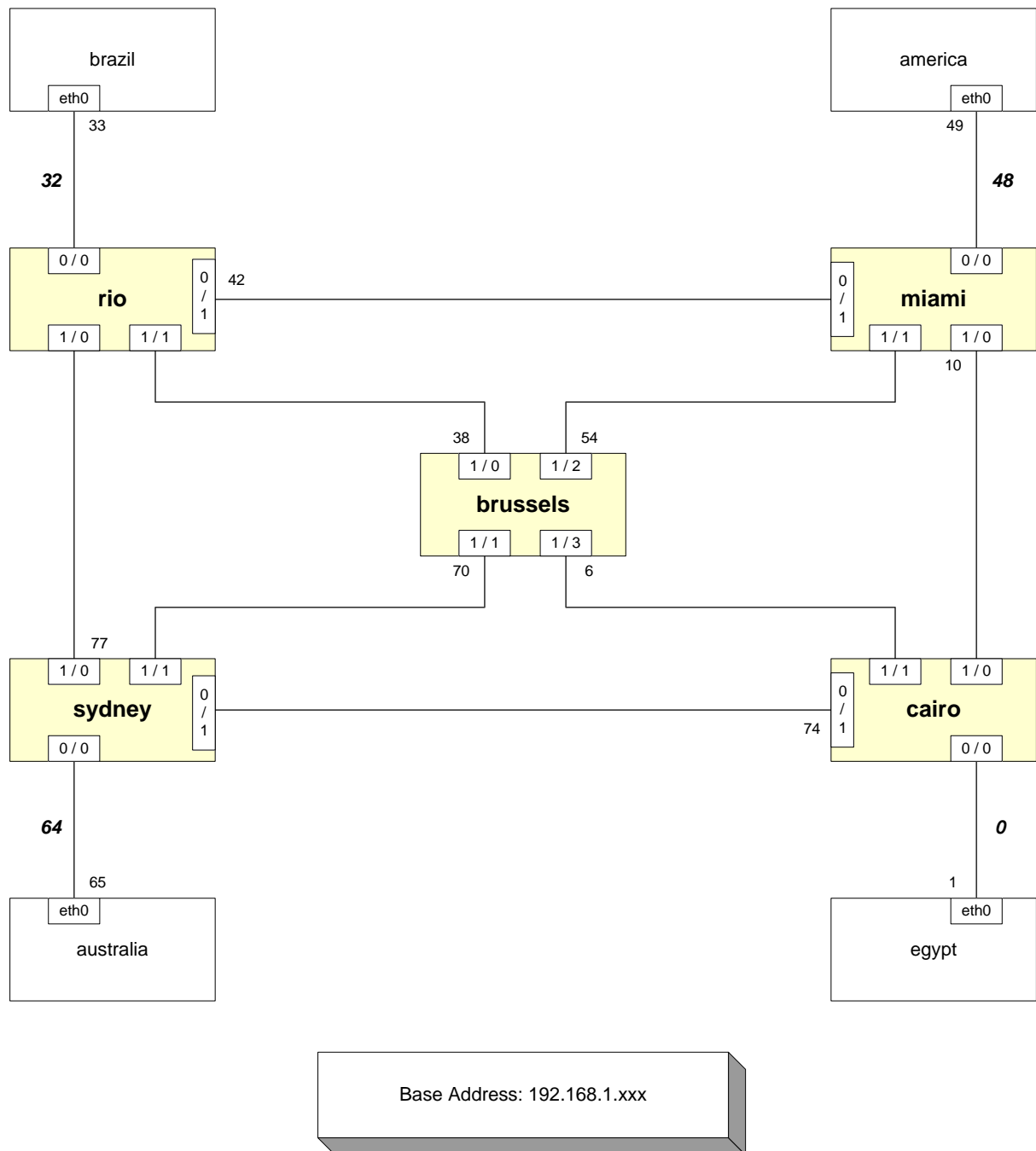


Figure 1: Layout of a topology. In bold, the routers are represented, while the hosts are in normal text. The subnet IP addresses are placed in bold beside the subnet.

Q3.5: In the topology displayed in Figure 1, we have used Class C addresses for our routers and hosts. The subnet mask we have used has been omitted from the figure. Find the subnet mask that we have used for all subnets and explain how a router can determine the destination subnet based on a destination IP and subnet mask.

Q3.6: Find the maximum number of subnets we can define in the network with this subnet mask.

Q3.7: Give the IP addresses of all subnets in the figure.

Realize the topology of Figure 1 in Mininet (you may use Miniedit).

In Brazil there is a person who would like to communicate with somebody in Egypt.

Q3.8: In order to reach this goal, which hosts and routers do you need to configure?

Q3.9: Are there multiple answers to the previous question? If yes, what possible deviations do you have and which one did you choose? Motivate your answer.

Configure all interfaces of the hosts and routers you consider necessary to fulfill communication between Brazil and Egypt. Confirm your configuration by showing the host and router interface configuration using the command *ifconfig*.

Q3.10: Configure the appropriate routing rules using the command route. Which rules did you choose to configure and why?

Confirm connectivity by showing a ping between the hosts Brazil and Egypt.

Exercise 4: Errors and TCP (Hand-in before 02/06)

Q4.1: A message $C = 100110011001$ has been received that contains a Cyclic Redundancy Check (CRC) at the end that was made with generator $G = 11011$. Compute the CRC. Has the message been received correctly?

The purpose of the remainder of this assignment is to use Wireshark to familiarize yourself with TCP.

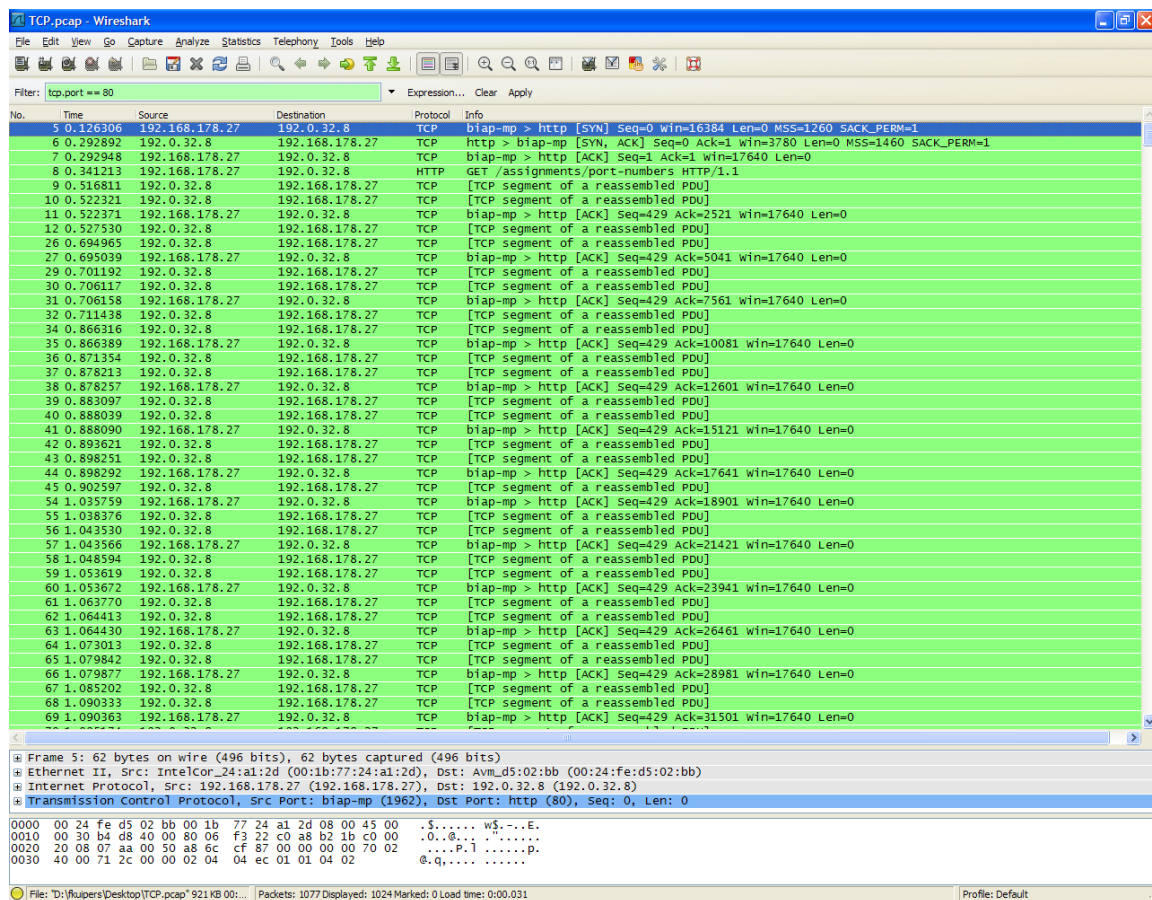
Start Mininet ('sudo mn') and activate a simple http server on h1:

```
mininet> h1 python -m SimpleHTTPServer 80 &
```

Port 80 refers to HTTP.

Subsequently, on h2, start a capture with Wireshark. Now let h2 get a page from h1 through the command 'h2 wget <http://10.0.0.1>'. Wait until the page has completely loaded and then stop your capture.

The Wireshark capture will display all the captured data, but for this assignment we are only interested in TCP data corresponding to port 80. So let's use the display filter "tcp.port == 80" to only display TCP traffic related to port 80 (if another port number is used, then use that one). You will see a screen similar to:



Q4.2: Identify the three-way handshake that is used to set up the TCP connection. For each of these three segments, answer the following questions:

- *List the source and destination sockets.*
- *Which flags are set?*
- *What is the maximum segment size? (Which maximum segment size will be used?)*
- *List the relative and actual sequence number and acknowledgment number (if present). The actual number can be acquired from the packet content (in hex or binary)*
- *What is the header length?*
- *What version of TCP is being used?*

After the HTTP GET command, the actual download of the web page takes place. Click on any TCP segment after the HTTP GET command. Under statistics, you may find TCP Stream graph, which allows you to analyze the Round-trip Time, Throughput, Time/Sequence (Stevens'-style), and Time/Sequence (tcptrace-style).

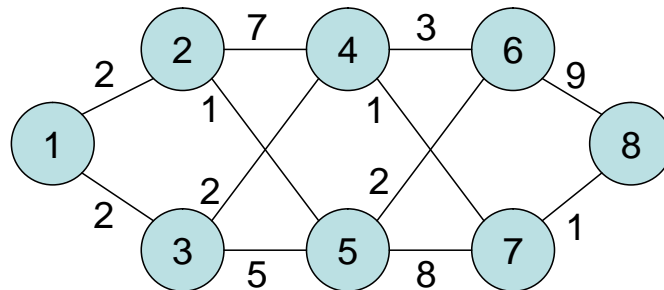
Q4.3: Describe what these four plots tell you about your TCP session.

Q4.4: Return to have a look at the packet capture and:

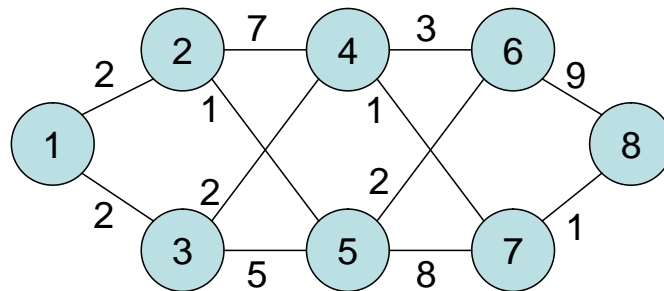
- *Determine where the TCP connection finishes, how do the client and server reach an agreement on the finished state? Determine the states for both ends and determine how long they reside in these states.*
- *Describe another way in which a connection can terminate.*

Exercise 5: Trees and OSPF (Hand-in before 09/06)

Q5.1: On the following network, compute with via both the Dijkstra and Bellman-Ford algorithms the shortest paths from the source node 1 to all other nodes in the network. Give an activity table of your calculations.



Q5.2: On the following network, compute via both the Prim and Kruskal algorithms the minimum spanning tree. Clearly show all steps/calculations, for instance via an activity table.



Q5.3: For the above two networks, determine the min/avg/max degree, edge connectivity, and vertex connectivity.