Exercise Session 1: Basic Networking Tools

COM-208: Computer Networks

Welcome to the first Computer Networks exercise session! Today's session will be a full lab. The goal of the lab is to get you familiar with basic networking tools. Before you start, please watch the lecture videos and read the "Doing the Labs" document posted on Moodle.

Network interfaces and their names

Every computer in the world has at least one **network interface**. Whenever an entity outside the computer wants to communicate with the computer, it needs to name one of its network interfaces. Different entities use different names to refer to a computer's network interface: the network layer uses **IP addresses**, the link layer uses **MAC addresses**, the computer's operating system (OS) uses **local interface names**.

The **ifconfig** utility lists a computer's network interfaces and displays or updates their configuration.

Type `ifconfig` in the command line and answer the following questions:

- How many network interfaces does your computer have?
- Can you guess why it has more than one?
- What is the IP address of each interface?
- What is the MAC address of each interface?
- Why could it be that some interfaces do not have a MAC address? (This is a tough question. It's normal if you don't know how to approach it yet. Come back to it at the end of the lab, after having worked on the Internet layers.)

DNS names and IP addresses

Humans use special names, called **DNS names**, to refer to computers (more precisely, to the network interfaces of computers). When you instruct your computer to communicate with a remote computer that has a given DNS name, your computer translates, under the covers, the given DNS name to an IP address.

The **host** utility helps you map DNS names to IP addresses. E.g., if you type `host` *target* in the command line, where *target* is a DNS name or IP address, that will display the IP address(es) and potentially other DNS names of the target network interface.

Use the `host` utility to answer the following questions:

- What are the IP addresses of www.epfl.ch?
- Why could it be that www.epfl.ch maps to more than one IP addresses?
- What is the IP address of www.google.com?
- Answer the same question again in an hour or so. Has anything changed? If so, what could be the reason for the change?

Reachability

The **ping** utility helps you check whether a remote computer is "reachable" from your computer. E.g., if you type `ping` *target* in the command line, where *target* is a DNS name or IP address, that will tell you whether your computer can reach the target network interface.

When one computer pings another, it sends to it a small packet, requesting a response, and it measures the time it takes from the moment each request is sent until the corresponding response is received. This time has a special name: it is called the **round-trip time (RTT)** between your computer and the target.

Use `ping` to answer the following questions:

- Are the following computers (more precisely, network interfaces) reachable from yours: www.epfl.ch, www.20min.ch, www.swisscom.ch, 8.8.8.8, www.microsoft.com, www.auth.gr, en.sjtu.edu.cn, www.adelaide.edu.au?
- Notice the RTT reported by `ping` for each target computer. Do you see a pattern? Which targets tend to have longer RTTs?
- If you let it run, `ping` makes many efforts to reach each target. As you can see, the RTT (to the *same* target) changes with every effort. What could possibly be the reason for this change?

- At least one of the targets should be unreachable through `ping`. Try to reach it by typing its DNS name in your web browser. How could it be that the same target is unreachable through `ping` but reachable through your browser?

Network paths and packet switches

When two computers (end-systems) communicate with each other over the Internet, their communication traverses multiple **packet switches**. There are two general types of packet switches on the Internet: link-layer switches and network-layer switches (the latter are also called **routers**).

The **traceroute** utility lists the routers that are located between your computer and a remote one. E.g., if you type `traceroute` *target* in the command line, where *target* is a DNS name or IP address, that will display a list of router DNS names and/or IP addresses and the RTTs that were measured between your computer and each router.

(The idea behind how traceroute works is a little wonder, and we will explore it later in the semester. If you feel curious, you can already google it.)

Use `traceroute` to answer the following questions:

- How many routers are there between your computer and www.mcgill.ca?
- How many of these routers are inside the EPFL network? How many, would you guess, are inside EPFL's Internet Service Provider (ISP)?
- Between which of these routers, do you think, your packets cross the Atlantic?
- Now `traceroute` to www.google.com. Does the network path from your computer to www.mcgill.ca overlap with the path from your computer to www.google.com?
- Traceroute again to www.google.com in an hour or so. Is the output (the sequence of routers) the same as before? What does the answer say about the network path from your computer to www.google.com?
- There exist **traceroute servers** that allow you to traceroute from them to any other computer in the world. For example, this one, or this one. Traceroute from a traceroute server to your computer, then from your computer to that server. Are the two paths symmetric? You can find other traceroute servers at www.traceroute.org and play around with them, e.g., traceroute from one server to another, check if the paths are symmetric, and try to guess the geographic locations of the servers.

Remote connection

The **ssh** utility enables you to establish a secure communication channel between your computer and a remote one. E.g., you can type `ssh` *username@target* in the command line, where *target* is the remote computer's DNS name or IP address.

Use `ssh` to connect to one of the computers in INF3, e.g., icin3pc01.epfl.ch, using your gaspar username. If you are working through vdi, you may be unsuccessful, but this does not really affect the next steps of the lab; just leave the `ssh` command running, waiting to be prompted for your gaspar password.

Port numbers

A computer's Operating System (OS) assigns a **port number** to every process running on the computer. Certain processes are always assigned the *same* port number on every computer where they run. Differently said, certain port numbers are universally reserved for specific processes. In any Unix-like environment, the file `/etc/services` lists these special processes and port numbers.

Either over the ssh connection you previously established or, if you were unsuccessful, on your own computer, open `/etc/services` and take a look at the contents, e.g., by typing `cat /etc/services` in the command line.

- Do you recognize any of the special processes (also called "services") listed on the left?
- Which is the port number that is reserved for ssh-server processes? What about web-server processes and mail-server processes?
- Next to each port number, there is a "udp" or "tcp" label. Do you remember what these are from the video lectures?
- Interspersed with the protocol names and port numbers are some human names. What do you think those are? Do you recognize any of them?

Active "communication sessions"

The **netstat** utility displays the contents of various network-related data structures that are stored in your computer. E.g., if you type `netstat -t` in the command line, that will display the list of "communication sessions" that are active between your computer and remote computers (actually, they are TCP connections, but you are not supposed to know about them yet...)

- The “Local Address” column lists processes that are running in the application layer of your computer. Notice that the names of all (or most of) these processes share a common prefix. Why is that? What does this prefix correspond to? (Did you run into it earlier in this lab?)
- The “Foreign Address” column lists all the processes that are running in the application layer of a remote computer that your computer is communicating with. Can you tell which one corresponds to INF3 computer you have ssh-ed into? (Reminder: if you are working through vdi, you may have been unable to connect to an INF3 computer, in which case `ssh` is still waiting to be prompted for a password; do not kill it and do not exit the password prompt.)

Layers and headers

The Internet architecture operates in layers. As a result, a packet that traverses the Internet looks, in a way, like an onion: On the “outside,” it is “wrapped up” in a link-layer header (which can be understood only by the link layer of computers and packet switches). If we “peel away” the link-layer header, we will find a network-layer header (which can be understood only by the network layer of computers and packet switches). If we also peel away the network-layer header, we will find a transport-layer header (which can be understood only by the transport layer of the end-point computers). And if we peel that away, too, we will find the application-layer header and data, which is the actual message that this packet is carrying.

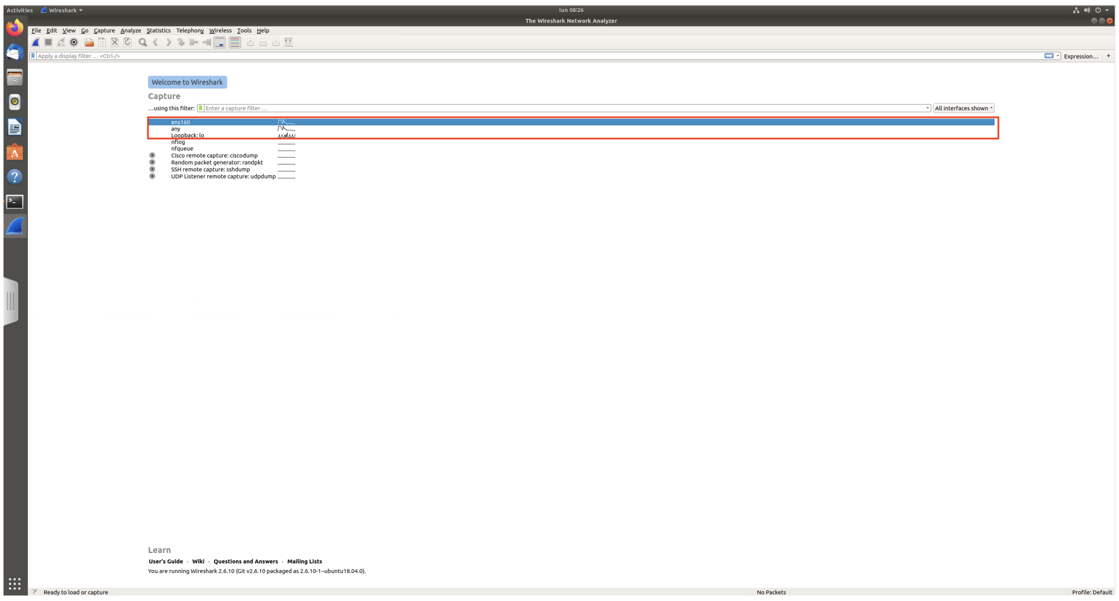
So, if we look inside an Internet packet, we will find a lot more information than the application-layer message that the packet is carrying: we will find meta-data, in the form of headers, which are needed by the various Internet layers in order to get the message from its source to its destination.

We will now use an application called **Wireshark** to do precisely that: look inside Internet packets. To get started, do the following:

- Start your web browser and clear its cache. If you are using Firefox, click on the ≡ symbol on the upper-right, go to Settings $\longrightarrow$ Privacy & Security $\longrightarrow$ Cookies and Site Data $\longrightarrow$ Clear Data.
- Start the Wireshark tool, e.g., by typing `wireshark` in the command line. You should see a list of your computer’s network interfaces (see Fig 1). Identify the one whose packets you will capture. If you are working through an INF3 computer or connected through vdi, you want to capture packets from your ethernet network interface. If you are working on a wirelessly connected computer, you want to capture packets from your WiFi interface.
- Start a capture by double-clicking on the target network interface. You should see data rolling inside the top part of your Wireshark window. These are the packets

that are departing from and arriving at your network interface. They most likely make no sense, and that's normal (by the end of the course, they will).

- Use your web browser to visit http://www.mit.edu.
- Stop capturing packets when the web page is fully loaded, by clicking on the square red button at the left of the top menu.
- Right underneath the top menu, you can specify a filter that you want to apply to the packets that you see.


Activities
Wireshark
lun 08:26
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
ens160
any
Loopback: lo
nflog
nfqueue
Cisco remote capture: ciscodump
Random packet generator: randpkt
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
Ready to load or capture
No Packets
Profile: Default

You are running Wireshark 2.6.10 (Git v2.6.10 packaged as 2.6.10-1~ubuntu18.04.0).

Figure 1: Wireshark startup page

Answer the following questions:

- What messages were exchanged at the **application layer**, i.e., between your web browser and the MIT web server?

  Type `http` in the filter line, you should see all the packets carrying HTTP messages. HTTP (Hypertext Transfer Protocol) is the communication protocol used between web browsers and web servers. Now check the "Info" column to find the information that these messages are carrying. Is the MIT server using HTTP or a different protocol to send the content of the website to your browser (Hint: look at the HTTP response and its header fields)?

- Which technology/communication protocol was used at the **transport layer**? There are two of them, TCP (Transmission Control Protocol) and UDP (User Datagram Protocol), and you need to figure out which one was used. To answer, click on one of the packets in the top section of your Wireshark window, then check the detailed information about this packet that appears in the middle section of your window. You should see information about each layer. Near the bottom, you should see a line that refers to the application layer (it says “Hypertext Transfer Protocol”). What does the line on top of that say?

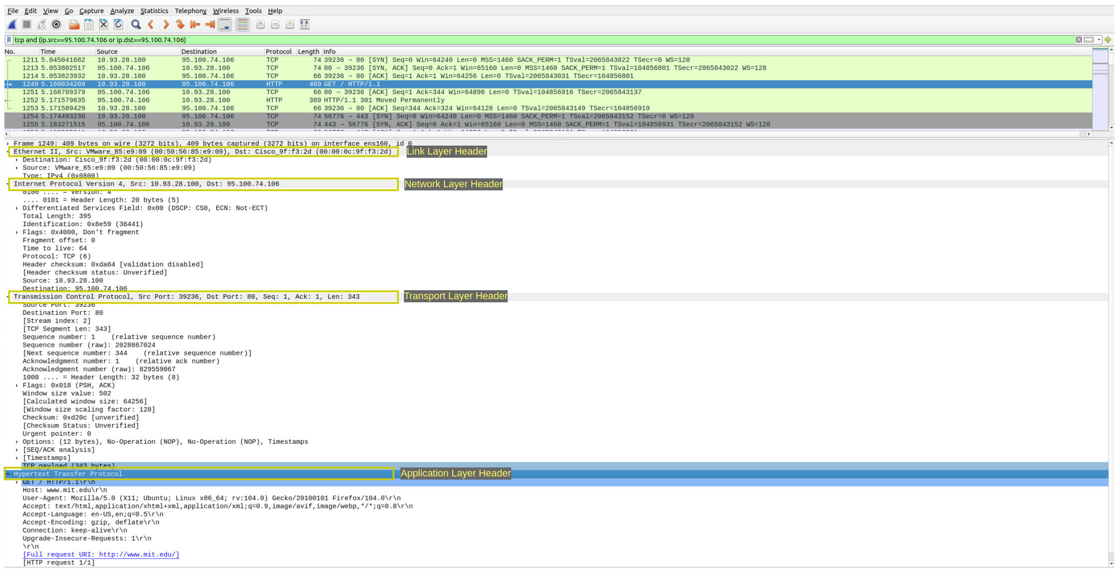
- What messages were exchanged at the **transport layer**, i.e., between the transport layer on your computer and the transport layer on the computer running the MIT web server?

  This is a little bit trickier to answer. First of all, you need replace `http` in the filter line with the correct transport-layer technology/communication protocol, which you figured out in the previous question. But if you do just that, then you will see ALL the messages exchanged by your computer using that protocol, whereas you only want the ones exchanged with the computer running the MIT web server. So, you need to add something more to the filter. Poke around a bit in Wireshark documentation on how to specify filters, and you should figure it out.

  A key point here is that the application-layer messages and the transport-layer messages were not carried in separate Internet packets. Rather, the same packets carried BOTH transport-layer and application-layer information, but the transport-layer information was stored inside the transport-layer header of each packet, whereas the application-layer information was stored inside the application-layer header and data.

Now we will examine the concept of **encapsulation**, meaning that each message encapsulates a message that belongs to a higher layer. E.g., a network-layer message consists of a network-layer header plus a transport-layer message, which consists of a transport-layer header plus an application-layer message, which consists of an application-layer header plus data.

Display the messages exchanged at the application layer (the HTTP messages) and click on one of them. Can you spot the different layers? Fig 2 shows where each header starts. Notice that each header has different fields from the other headers. Each field is there to serve a specific functionality related to that layer. In this course, we will go through each layer, from bottom to top, and explain its functionalities. Towards the end, you will understand how the different layers interact and what most of these fields mean.



```
tcp and (ip.src==95.100.74.106 or ip.dst==95.100.74.106)
No.  Time        Source         Destination    Protocol Length Info
1211 5.045041682 10.93.28.100   95.100.74.106  TCP       74 39236 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2065843022 TSecr=0 WS=128
1213 5.053802517 95.100.74.106  10.93.28.100   TCP       74 80 → 39236 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=104856801 TSecr=2065843022 WS=128
1214 5.053823932 10.93.28.100   95.100.74.106  TCP       66 39236 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2065843031 TSecr=104856801
1249 5.160034209 10.93.28.100   95.100.74.106  HTTP     409 GET / HTTP/1.1
1251 5.168709379 95.100.74.106  10.93.28.100   TCP       66 80 → 39236 [ACK] Seq=1 Ack=344 Win=64896 Len=0 TSval=104856916 TSecr=2065843137
1252 5.171579635 95.100.74.106  10.93.28.100   HTTP     389 HTTP/1.1 301 Moved Permanently
1253 5.171589429 10.93.28.100   95.100.74.106  TCP       66 39236 → 80 [ACK] Seq=344 Ack=324 Win=64128 Len=0 TSval=2065843149 TSecr=104856919
1254 5.174493236 10.93.28.100   95.100.74.106  TCP       74 56776 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2065843152 TSecr=0 WS=128
1255 5.183271515 95.100.74.106  10.93.28.100   TCP       74 443 → 56776 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=104856931 TSecr=2065843152 WS=128

Frame 1249: 409 bytes on wire (3272 bits), 409 bytes captured (3272 bits) on interface ens160, id 0
Ethernet II, Src: VMware_85:e9:09 (00:50:56:85:e9:09), Dst: Cisco_9f:f3:2d (00:00:0c:9f:f3:2d)
   Destination: Cisco_9f:f3:2d (00:00:0c:9f:f3:2d)
   Source: VMware_85:e9:09 (00:50:56:85:e9:09)
   Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 10.93.28.100, Dst: 95.100.74.106
   0100 .... = Version: 4
   .... 0101 = Header Length: 20 bytes (5)
   Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
   Total Length: 395
   Identification: 0x8e59 (36441)
   Flags: 0x4000, Don't fragment
   Fragment offset: 0
   Time to live: 64
   Protocol: TCP (6)
   Header checksum: 0xda84 [validation disabled]
   [Header checksum status: Unverified]
   Source: 10.93.28.100
   Destination: 95.100.74.106
Transmission Control Protocol, Src Port: 39236, Dst Port: 80, Seq: 1, Ack: 1, Len: 343
   Source Port: 39236
   Destination Port: 80
   [Stream index: 2]
   [TCP Segment Len: 343]
   Sequence number: 1    (relative sequence number)
   Sequence number (raw): 2028867024
   [Next sequence number: 344    (relative sequence number)]
   Acknowledgment number: 1    (relative ack number)
   Acknowledgment number (raw): 829559067
   1000 .... = Header Length: 32 bytes (8)
   Flags: 0x018 (PSH, ACK)
   Window size value: 502
   [Calculated window size: 64256]
   [Window size scaling factor: 128]
   Checksum: 0xd20c [unverified]
   [Checksum Status: Unverified]
   Urgent pointer: 0
   Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
   [SEQ/ACK analysis]
   [Timestamps]
   TCP payload (343 bytes)
Hypertext Transfer Protocol
   GET / HTTP/1.1\r\n
   Host: www.mit.edu\r\n
   User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:104.0) Gecko/20100101 Firefox/104.0\r\n
   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
   Accept-Language: en-US,en;q=0.5\r\n
   Accept-Encoding: gzip, deflate\r\n
   Connection: keep-alive\r\n
   Upgrade-Insecure-Requests: 1\r\n
   \r\n
   [Full request URI: http://www.mit.edu/]
   [HTTP request 1/1]
```

Figure 2: The different layers of a message

But now let us see if you can figure out some:

- How many bytes does the HTTP message contain? To answer, check the packet details in the middle section of your Wireshark window. Look at the transport-layer information and, in particular, the `Len` field, which specifies the size of the application-layer message that is encapsulated inside the transport-layer message.
- How many bytes do the transport-layer and network-layer headers add to the HTTP message?
- How many bytes does the link layer add?

This is it! If you feel like it, use Wireshark to capture your computer's communications (outside the lab). You may be surprised...