Queen Mary
University of London

ECS404U: COMPUTER SYSTEMS & NETWORKS

2020/21 – Semester 1

Prof. Edmund Robinson, Dr. Arman Khouzani

Lab Week 11: *Introduction to Computer Networks*
MAC Address, IP Address, DNS, TCP/UDP, HTTP/SMTP

November 30 / December 02, 2020

Deadline for submitting your proof of work: ***Next week's Thursday, at 10:00 AM UK time***

Student Name:

20699145

Student ID:

Yujeong JUNG

Brief Feedback to student (commendations, areas for improvement):

Summary and Learning Objectives

This lab is an introduction to computer networks. As with a lot of topics introduced in the first year undergraduate, it is basic and at times simplistic. But the aim is that we get familiar with basic concepts and ideas. Namely:

- Being able to find the MAC address of a device; Getting intuition why we need a MAC address;
- Getting intuition why we need an IP address; Being able to find the IP address of a device connected to the internet; Tell the difference between IPv4 and IPv6 addresses; A brief familiarity with subnet masking;
- Understanding the basic concept of routing; Being able to identify the IP and MAC addresses of your immediate router (the "default gateway"); Seeing examples of end-to-end routes with **`traceroute`**;
- Getting to grips with why we need a Domain Name System, and being able to get DNS information about some sample domain names;
- Seeing an example of Packet sniffing using **`wireshark`**, and being able to identify different packets based on their transport or application layer protocols;
- Getting an intuition about the difference between UDP and TCP transport protocols; Hence the need for a "sequence number" in TCP;
- Being able to identify the steps of a TCP handshake; specially the establishment of the sequence numbers and their incrementation;
- Seeing two frequently used application layer example protocols: HTTP and SMTP; Being able to extract some data from a HTTP header;

1 Layer 2 – Link Layer: Network Interface, MAC Addresses

Suppose you want to connect your computer to a network. First thing first, you need a network interface card (NIC) whose job is to translate your data (in binary) to carrier signals over your physical medium. The carrier signals can be electromagnetic waves in a guided medium like an ethernet cable (e.g. Cat6) or a coaxial cable (e.g. RG-59), or electromagnetic waves in the unguided medium of air as in a wireless case (e.g. wifi, wimax, cellular 3G/4G/5G, satellite, etc.), or light beams in the case of optical fibre.
But chances are that this medium is shared: e.g. you are not the only person in your negihbourhood trying to connect to the Internet via wifi or cellular network. Each of the devices on the same medium will be "listening" to all the data communication on the medium, and will only "pick up" the ones that belong to them, ignoring the rest. But how can we tell which data is for whom? This is where the MAC address comes to picture: Each device (more precisely, its network interface card) comes with a unique identification number (set into its hardware by its manufacturer). Each frame of data has the MAC address attached to it, so that the devices that share the same medium know which frames belong to them by matching the MAC address of the frame with their own MAC address, and (typically) ignore the rest.

Q1. In the context of MAC addresses, what does the acronym MAC stand for? Does it make sense?

MAC address stands for Media Access Control address. Yes, it's make sense.

Q2. How can we find out the MAC address of our device? Here is a way to do it in Linux/MacOS. Open a terminal, and issue the following command (type it, then press enter):

```
ip addr show
```

You should see information about all the network interfaces on your device. From the list, you should be looking for the "Ethernet" network interface, probably named **`eth0`**. Using the output and perhaps a bit of online research about how to parse the output, write down your MAC address below.

Similar information can be obtained in Windows PowerShell by the following command:

```
Get-NetAdapter
```

or following one of the methods explained in this link.

f0:18:98:10:01:c4

Q3. What is the length of the MAC address (in number of bits)? How many unique MAC addresses can we have with this length?

48 bit / 2^48

Q4. The MAC address can be divided into two parts. The first half carries information about the vendor that produced the network interface card. Using the following website: https://hwaddress.com/, find out the vendor of the NIC on your machine.

OUI: F0-18-98
Company: Apple, Inc.

Q5. A special MAC address is reserved to indicate a data-frame on the link is intended for every device listening on the medium. This is called "*broadcast*" MAC address. So all devices on the link will be picking up such data-frames. Referring to the output of the **`ip addr show`** command from the terminal, write down what is the broadcast MAC address on your ethernet link.

f0:18:98:10:01:c4

2 IP Layer: IP Addresses, Routing

MAC addresses resolves the issue of finding which data is for whom for devices that are immediately connected to the same medium, i.e., they are on the same Local Area Network (LAN). But when we connect to a web server (like `www.google.com` or `https://www.youtube.com`), the computer we are trying to communicate with is far away, not on our LAN! So how do we reach

them (to send a request for their data), and how does their data reach back to us? As you can guess, the data has to be "routed" between these computers. Routing is simply the process of finding a path from a source device to a destination device on a network. The data "packets" are forwarded hop-by-hop along this path to reach their destination. Routing of the data around the Internet is done (primarily) by dedicated hardware devices called "Routers".

How does a router know which next hop to send a packet to? We need a way of addressing each computer on the network so that their packets can be routed to them. This is where the IP Address comes into the picture. Each device that is connected to the Internet is assigned an IP address typically through a protocol called DHCP: Dynamic Host Configuration Protocol. This can be done e.g. by your closest router: the one that connects your LAN to the rest of the Internet. That router is often called the "gateway". Now that you have an IP address, any device on the internet can reach you through your gateway router.

Q6. Explain why the MAC address could not have been used to play the role of the IP address.

IP addresses can be changed because they are not uniquely assigned to each communication device but are assigned by another agent. However, the MAC address cannot be changed because it is the only unique identification number (like serial number) granted to the hardware itself of the communication device.

Q7. What is the length of an IP address (IPv4) in number of bits? How many distinct IPv4 addresses can there be? Give a brief explanation why this motivated the introduction of IPv6 (among other factors).

32bits
2^32 (there are 4,294,967,296 unique values)
IPv4 with 32 bits can give up to 4 billion different addresses. Given the exponential growth in user demand, however, the problem is that the current IPv4 framework cannot meet the ever-demanding demand for Internet addresses.

Q8. Using the same command as before (from the terminal), find out your assigned IP address (both IPv4 and IPv6) and write them down.

```
ip addr show
```

If you are using Windows, you can follow the instruction in this link, or

inet6 fe80::1c66:18d8:7d83:1916%en0 prefixlen 64 secured scopeid 0x6
inet 192.168.35.200 netmask 0xffffff00 broadcast 192.168.35.255

Q9. Explain the fallacy in this argument: "I am not serving any webpages, or other online services. I only want to browse the web. So I understand that the computer that hosts e.g. `www.google.com` needs to have an IP address, but why would my computer need an IP address as well?"

IP address is kind of own home address. That is, you need this IP address in order to send data from the host (computer) to the exact device. The reason for using IP addresses is to receive and transmit data accurately from computers by separating each host.

Q10. Use the **`traceroute`** service (which is also available from this link:https://ping.eu/traceroute/) to get the actual route to some domain names of your choice. For example:

```
traceroute www.google.com
```

Then answer the following questions:

- How many hops does each route take?
- How many hops are common between `www.google.com` and `www.youtube.com`?
- How many hops are common between `www.qmul.ac.uk` and `eecs.qmul.ac.uk`?
- How about between `www.qmul.ac.uk` and `qmplus.qmul.ac.uk`?
- Which one of these routes has the least **latency** (best)? Which one has the highest latency (worst)? (use `ping` (https://ping.eu/ping/) if you don't get latency result from traceroute.)

- 15 hops
- google.com; max 30 hops youtube.com; max 30 hops
- qmul.ac.uk; max 30 hops eecs.qmul.ac.uk; max 30 hops
-qmul.ac.uk; max 30 hops qmplus.qmul.ac.uk; max 30 hops

<Latency>
google; 56 data bytes/ time 3011ms
youtube; 56 data bytes/ time 3015ms
qmul; 56 data bytes/ time 3007 ms
eecs.qmul; 56 bytes / time 3012 ms
qmplus.qmul; 56 data bytes/ time 8288 ms

Best: www.qmul.ac.uk
Worst: qmplus.qmul.ac.uk

Q11. Using the output of the following two commands, find out the IP address and the MAC address of your router (default gateway):

```
ip route list
ip neigh show
```

IP: 192.168.35.200
MAC: f0:18:98:10:01:c4

2.1 Domain Name System

We pushed a lot of important detail under the carpet. Among which is the following: when I want to visit a website like `www.google.com`, or send an email to `arman.khouzani@qmul.ac.uk`, I just type in their name. I never remember having to deal with finding out their IP addresses. Unfortunately, humans cannot remember bit sequences, even if they are compartmentalised and represented in decimal! So we need a way to translate these easy-to-remember-for-humans domain names, to actual IP addresses, so that the routing can take place. This is where **Domain Name Service** steps in. In simple terms, some computers, referred to as **DNS Servers** should keep a lookup table of the domain names vs their IP addresses and provide these information upon request.

Q12. Use the **`dig`** or **`host`** or **`nslookup`** commands (from the terminal), or use this link: https://ping.eu/nslookup/, to get the IP addresses (both IPv4 and IPv6, if available) of the following domain names:

- `www.google.com`
- `www.youtube.com`
- `www.qmul.ac.uk`
- `eecs.qmul.ac.uk`

google; 216.58.211.4
IPv6 address 2a00:1450:400f:80d::2004
youtube;142.250.74.46
216.58.207.206
216.58.211.142
142.250.74.14
172.217.22.174
172.217.21.174
216.58.211.14
IPv6 address 2a00:1450:400f:80a::200e
IPv6 address 2a00:1450:400f:808::200e

Q13. Which DNS server is providing these information?

google;Server: 210.220.163.82
Address: 210.220.163.82#53
youtube;Server: 210.220.163.82
Address: 210.220.163.82#53

3 Transport Layer: TCP handshake (OPTIONAL)

This part was designed for ITL machines, unless you want to install **wireshark** on your own machine. This part is made optional and you can choose to skip to Section 4.

The "data" that we have been speaking of is not transmitted in a continuous blob. Instead, it is broken down into small chunks called datagrams, or segments. These segments can themselves be broken down into "packets". And each packet is routed independently of the previous ones. So it can even happen that a packet follows a completely different path compared to the packets of the same blob of data!

The transmission control protocol (TCP) takes care of segmentation and putting all the segments back to construct the original blob of data. For this reason, it need to establish a *Sequence* number between two hosts that want to communicate. Each segment then gets incremented, so that it can identify if a segment is missing (so it can request a retransmission) and also it can put the segments back in order. This takes place during the initial "handshake", as depicted in Fig 1:

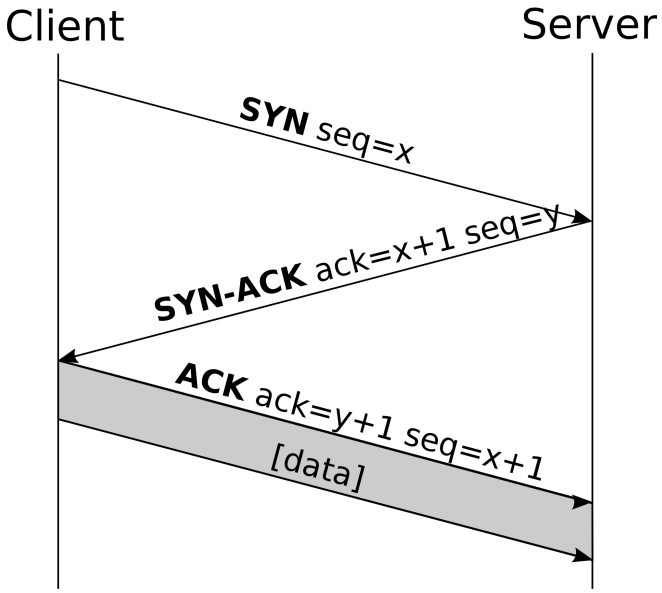

Figure 1: TCP handshake to establish the sequence numbers.

We are going to use the **wireshark** packet inspection tool to see this in action! You can downlaod and install wireshark from its website: https://www.wireshark.org/.

1. After its installation is complete, open **wireshark**.
2. When it opens, choose the network card to start capturing packets on.
3. You should immediately see a barrage of packets being sniffed! It seems there are a lot of traffic taking place behind the scenes while nothing appears to the eye of the user!
4. While the **wireshark** is open and running, open a http page (using say firefox). Now in the **wireshark**, in the `Filter` field, type in `tcp`, so that only TCP packets are shown.
5. Identify the datagrams related to your event of opening an http page.
6. In particular, identify the first three datagrams for this session, and try to match it with the TCP 3-way handshake (as depicted in Figure 1). To make this task easier, you can right-click on one of the packets of this session, and click on `Follow TCP Stream`. Then only the packets of this TCP stream will be shown. So the handshake packets will be **the first three**.

Q14. What sequence numbers are selected? Which party does the selection exactly? Why are there two sequence numbers? **Note:** Wireshark by default shows the "relative" sequence numbers, which start from zero. In order to see the actual sequence numbers, you need to inspect the TCP headers. Alternatively, you can also change the setting under `Preferences » Protocols » TCP` and untick the `Relative Sequence Numbers`.

Q15. TCP requires that delivery of each segment be "acknowledged" by the receiving party, to ensure no segment of the data is lost. By following the TCP stream, explain the relation between the `ACK` number and the sequence numbers and how the **`ACK`** number gets incremented.

Q16. UDP is another transport layer protocol, but unlike TCP it does not require ACKnowledgement, neither does it guarantee in sequence delivery of data segments. This makes it "unreliable". But what kind of applications can you think of which would prefer UDP over TCP? Hint: you can filter for UDP traffic in wireshark and see some examples (just type in **`udp`** in the `apply a display filter` field.).

4 Application Layer: HTTP, SMTP

Typically, most of the data that we are communicating over the network are associated with an application. Handling the data for file transfer is different than handling data for videoconferencing, or for sending an email. Each of these applications involve their own sequence of tasks and specific formatting. This is handled by the Application Layer Protocols. Two of the most popular ones is HTTP (HyperText Transfer Protocol) and SMTP (Simple Mail Transfer Protocol). HTTP is a protocol used to access the World Wide Web (www) and SMTP is a protocol for sending and receiving emails. Here, we will see an example of each.

4.1 HyperText Transfer Protocol (HTTP)

HTTP protocol involves just two steps: an `HTTP request` message (from a "client" to a "server") and an `HTTP response` message (from the server back to the client). Each request or response message has a `header` and a `body`. The "header" section contains information about the "data"

section, e.g., its length and its type (formatting/encoding), as well as other “useful” information like cookies!

Q17. Let’s look at some HTTP response headers. Either use the “developer tools” in your browser, or use the **`curl -I`** command (from terminal), or from this fantastic website https://reqbin.com/curl to only request the header response to be sent back. For example:

```
curl -I www.google.com
```

By investigating the parameters of the response header, determine the encoding of the response body (if it was sent) for the following webpages: **`www.google.com`**, **`www.youtube.com`**, and **`eecs.qmul.ac.uk`** :

google; Content-Type: text/html; charset=ISO-8859-1
Transfer-Encoding: chunked

youtube; Content-Type: text/html; charset=utf-8

eecs.qmul; Content-Type: text/html; charset=UTF-8

Q18. How can the responses include multimedia (like images and videos) but the encodings seem to be only for text?

To express everything we use on a daily basis in computers, we need a process of converting to 0 and 1 according to the promised rules. This process is called "encoding." Encoding is used not only for characters but also for converting sizes of videos, music, etc.
However, many people think that encoding is the encoding of characters because it does not distinguish between character codes and encoding properly.

Q19. Let’s save the whole response and take a look at it! Run the following two commands (from the terminal, and in a directory of your choice) to save the HTTP response to an HTML file.

```
curl www.google.com > google_test.html
curl www.youtube.com > youtube_test.html
```

Now open the resulting HTML files in a browser. What do you see? Is it different from what you see when your browser does the same? How do you explain the difference?

```
$ curl www.youtube.com > youtube_test.html
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0

$ curl www. google .com > google_test . html
```

4.2 Simple Mail Transfer Protocol (SMTP)

From QM+, download a file called `ECS404U_(CSN)_Lab11.txt`. This is a file formatted according to the MIME (Multipurpose Internet Mail Extensions) format, so essentially an email in raw format! The main protocol for sending emails is SMTP (Simple Mail Transfer Protocol). Answer the following questions:

Q20. Given what we learned about TCP vs UDP, which transport layer protocol would be more natural/suitable to be used for SMTP?

TCP is more natural/suitable to be used for SMTP.
Typically, TCP and IP are used together, and if IP handles the delivery of data, TCP tracks and manages the packet. TCP provides a virtual line approach, which means connecting the source and the destination to assign a logical path to send packets. And the 3-way handshaking process means the process of establishing a session to ensure accurate transmission by ensuring the destination and destination. The reason why TCP has this feature is simple. This is because TCP ensures reliability as a connected service. That's why we're using 3-way handshaking, and we're also using functions like data flow control and congestion control. However, these features make it slower than UDP.

Q21. **Optional** Download the file from QM+ and open it in a simple text editor. What is the encoding of the body and the attachment of this email? Who is the sender and receiver of this email? Read the email! (What is the body of the email?)

Q22. **Advanced, Optional** We are now going to open the attachment of the email:

- Remove everything but the base64 encoded block that represents the attachment. Save as `att.txt`.
- Decode the file and save the result with the PNG extension. You can either use this website: https://www.base64decode.org/, or if you are using Linux, run the following instructions from the terminal (in the same path that you saved the `att.txt` in):

```
base64 -D < att.txt > att.PNG
```

If you run `man base64`, you will realise that the `-D` option means decode. Also `< file1` means "get your input from file1", while `> file2` means "put your output in file2". So the command translates as "run the base64 decoder, taking input from **`att.txt`** and putting output into **`att.PNG`**." Open the file, what do you see?