AYMARD Victor
ING3 - TD5

Initiation Réseaux
TP1 - Get started with Wireshark

1 Objective

1. Review basic concepts of computer networks: TCP/IP model, encapsulation, protocol stack, etc.
2. Learn and get familiar with the network protocol analyzer Wireshark.

2 TCP/IP Reference Model

The Internet protocol suite is the computer networking model and set of communications protocols used on the Internet and similar computer networks. It is commonly known as TCP/IP, because its most important protocols, the Transmission Control Protocol (TCP) and the Internet Protocol (IP), were the first networking protocols defined in this standard. Often also called the Internet model, it was originally also known as the DoD model, because the development of the networking model was funded by **DARPA** (Defense Advanced Research Projects Agency), an agency of the United States Department of Defense (DoD).

Source: https://en.wikipedia.org

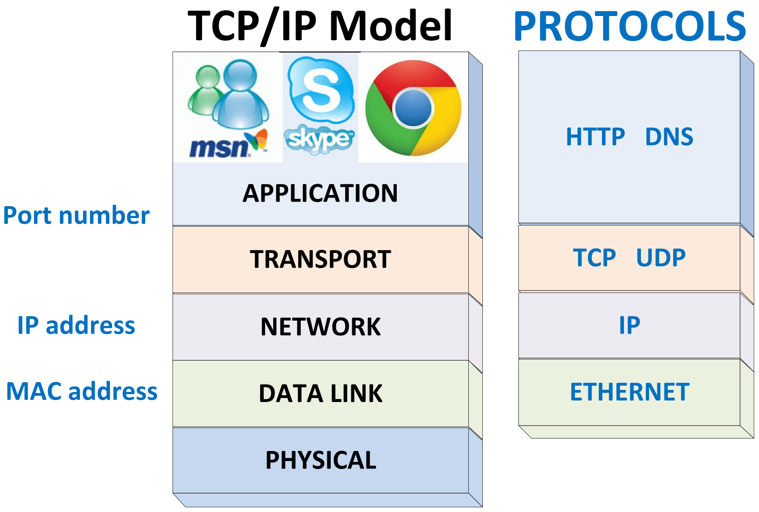

Figure 1: TCP/IP Reference model and the protocol stack

3 Wireshark

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues.

Wireshark is cross-platform? It runs on Linux, OS X, BSD, Solaris, some other Unix-like operating systems, and Microsoft Windows. Wireshark is very similar to **tcpdump**, but has a graphical front-end, plus some integrated sorting and filtering options. It is often used to debug and analysis of computer networks, develop protocols, etc.

For more information about Wireshark, please refer to http://www.wireshark.org

For more information about tcpdump, please refer to http://www.tcpdump.org

4 What can we do with the Wireshark?

- ## Part I: Capturing traffic

1. Open **Wireshark**. Attention: if you are using Linux, please to go the **root** mode.
2. Click **Capture** and then **Interfaces...**
3. Choose the Ethernet interface you want to observe. Tips: Choose the one connected with the Internet.
4. Click **Start**. Here we go!

- ## Part II: Analyze the protocols

1. Open one web browser (Chrome, Firefox, IE, as you wish), go to `http://www.ece.fr/ecole-ingenieur/`.
2. Click **Capture** and then **Stop** to stop the capturing after one minute. (OK, we already have enough data to check!)
3. Note that all the traffic and protocols are marked with different colors. How colorful the screen is! (and what a mess! shuuuuu....)

> There are three windows on the screen.
> The first one shows all captured frames ordered by capture time. Wireshark allows you to change the order of frames to show by other criterion, e.g., source address, destination address, protocol type, etc..
> The second one is the decoded content of the frame chosen for study. Especially, it is here we can see how a frame has been encapsulated by different protocols at different layers.
> The third one shows the hexadecimal values of the frame chosen. Note that it can highlight the part corresponding to a protocol chosen in the second window.

4. Let's try to filter the captured traffic. This can be done by the **Filter** in the tool bar.
5. Type **HTTP** in the **Filter** tool bar and type **Enter**. Now only HTTP frames are shown on the screen.
6. Let's first focus on the first window. Find the frame of **HTTP request** which contains the **GET** to ask for the website of ECE. Meanwhile, find the source IP address, the one of your PC.
   The number of this frame is: 729
   The IP address of your PC is: 10.4. 185. 156
   Find the destination IP address: 13. 104. 4. 52
   Think about why your PC knew the destination IP address since the only information you have entered is the URL.
7. Search for the **HTTP response** which contains the return code.
   The return code you found is 200 OK, which means that you are connected. (E.g., code **404** means the website **Not found**)
8. The **HTTP** is an application protocol. Its data cannot be transmitted directly on the network partly because the transmission needs other information, like the Quality of Service, the destination address, etc. Therefore, the application data will be encapsulated by lower layers. To observe this process, focus now on the second window which gives you several items.
   - **Frame Nb.** is a brief summary of the captured frame, including the fame arrival time, frame length, the protocols used in the frame and the port number of the application.
   - **Ethernet II** gives the information of the Layer 2 header, including the source and destination MAC address and the protocol type of Layer 3.
     The MAC address of your PC is 44:85;00:19 ;CE:59
     The destination MAC address is 01:1e:bd:dd:fb:00, belonging to the server.
   - **Internet Protocol (is used)** (IP) summarizes the information used for Layer 3, the Network Layer.
     The version of the IP protocol used to transmit this frame is 4
   - **Transport Layer Protocol (is used)** includes the related information, like the source and destination port numbers.
     Which transport layer protocol has been used for HTTP frame ? TCP (TCP or UDP)
     The source port of your PC is 56300

The destination port of the web server is 80

- **Application Protocol (is used)** shows the information related to the application. In the case of **HTTP**, there is information about the web browser you used for the web page, etc.

9. Now search for the **dns** in the **Filter** tool bar. Among all the DNS frames in the first window, look for the **Standard query** for **www.ece.fr**.
   The number of this frame is: 36
   The destination IP address of the frame is: 10.3.1.31. It is the IP address of the DNS server which is in charge of translating the http://www.ece.fr/ecole-ingenieur/ to the IP address of its web server.

   > The Domain Name System (DNS) translates domain names, which can be easily memorized by humans, to the numerical IP addresses needed for the purpose of computer services and devices worldwide.
   > Source: https://en.wikipedia.org

10. Just after the **Standard query** for **www.ece.fr**, you can find the **DNS Standard query response** which returns the IP address.
11. Have a close look at the **DNS** frame in the second window.
    The Transport Layer protocol is UDP
    The port number of the **DNS** is 53

Part III: TCP and UDP

1. Type **tcp** in the **Filter** tool bar. Locate the **HTTP request** frame which contains the **GET** by its number. Just before this frame, you can observe the three-way handshake of TCP to establish a connection.

   > To establish a connection, TCP uses a three-way handshake. It occurs:
   > – SYN: The active open is performed by the client sending a SYN to the server. The client sets the segment's sequence number to a random value A.
   > – SYN-ACK: In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number i.e. A+1, and the sequence number that the server chooses for the packet is another random number, B.
   > – ACK: Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value i.e. A+1, and the acknowledgement number is set to one more than the received sequence number i.e. B+1.
   > Source: https://en.wikipedia.org

2. Type **udp** in the **Filter** tool bar. Locate the **DNS query** frame for **www.ece.fr** by its number. Notice that there is no three-way handshake to establish a connection. Indeed, UDP is a connectionless protocol.

   > UDP uses a simple connectionless transmission model with a minimum of protocol mechanism. It has no handshaking dialogues, and thus exposes any unreliability of the underlying network protocol to the user's program. There is no guarantee of delivery, ordering, or duplicate protection.
   > Source: https://en.wikipedia.org

Annex: Some useful Linux commands

```
# apt-get install wireshark   //Install Wireshark as root user
# grep 'tcp' /etc/protocols   //Check the protocol number assigned to TCP (or UDP)
# grep 'http' /etc/services   //Check a port number used by an application protocol
```