COMP90007 Internet Technologies: Lab 1

Semester 2, 2020, Week 2

1 Objectives

- To examine the concept of encapsulation in networking and the overheads associated with the layered networking model.
- To learn how protocols and layering are represented in packets through the use of Wireshark.
- To examine IP (Internet Protocol) in detail. IP is the main network layer protocol used throughout the Internet.
- To examine TCP (Transmission Control Protocol) in detail. TCP is the main transport layer protocol used in the Internet.
- To examine HTTP (HyperText Transfer Protocol) in detail. HTTP is the main application layer protocol underlying the Web.

2 Requirements

There are two software requirements for this lab:

- *Wireshark*

  Wireshark is a sniffing tool that allows the user to examine a packet trace, that is, a record of network traffic over time. It can be downloaded at http://www.wireshark.org/download.html.

- *wget*

  'wget' is a tool to fetch web resources. A standalone Windows version can be found at https://eternallybored.org/misc/wget/. On Linux and OS X, 'wget' is usually preinstalled, or can be installed with a package manager. 'curl' is also a suitable alternative.

OS X users running newer versions of the operating system may need to download the *Development Release* of Wireshark.

3 Pre-lab

The tasks in this section are designed to familiarise yourself with the Wireshark software which will be used in this lab. This section should be done *before* your scheduled workshop session.

3.1 Getting started

- Close all browsers and other applications that use the network or the Internet. In Wireshark, go to Capture ⇒ Options.
- Select a network adapter to listen to, tick the box *resolve network-layer names* and untick the box *Use promiscuous mode on all interfaces.*
- Click on *Capture filter* and select *TCP only.*
- You are now ready to capture your first trace. Click *Start* and use your favourite web browser to fetch a URL. Once the web page has loaded, go back to Wireshark and stop the capture.

3.2 Inspecting the trace

You will now see a list of frames captured by Wireshark, including timestamps, protocol types and additional information. Note that we are going to use 'packet' as a general term for captured entities here. Strictly speaking, a unit of information at the link layer is called a **frame**. At the network layer it is called a **packet**, at the transport layer a **segment**, and at the application layer a **message**.

- Select a packet for which the protocol column is 'HTTP' and the info column says 'GET'.

In the first window below the frame list you will see more detailed information about the packet and the headers of each layer (Have a look! Click on the '+' to expand for more details). The second window below shows the raw hexadecimal data the packet is made up of and translations thereof.
The packet we selected is the packet that carries the web (HTTP) request sent from your computer to the server. The packet structure reflects the protocols that are in use. Since we are fetching a web page, we know that the protocol layers being used are. That is, HTTP is the application layer web protocol used to fetch URLs. Like many Internet applications, it runs on top of the TCP/IP transport and network layer protocols. The link and physical layer protocols depend on your network, but are typically combined in the form of Ethernet if your computer is wired, or 802.11 if your computer is wireless.

3.3 Pre-lab questions

Confirm the answer to these questions with your tutor.

1. What does *resolve network-layer names* do and why did we untick the box *Use promiscuous mode on all interfaces.* What does that option do?
2. Did you observe any other traffic other than the HTTP packets? What are several reasons for this additional traffic that is captured? Does this traffic serve any purpose?

4 Lab tasks

In this lab, we will measure the traffic generated in requesting a web page. We will compare the number of bytes sent and received through the network with the actual size of the web resource we request. 'wget' will be used to request and download a single web page while we examine the corresponding network traffic in Wireshark.

Step 1

Close all unnecessary background programs. This includes web browsers, e-mail clients and any other programs which may generate background network traffic, which can skew the measurements you take later on.

Step 2

Open up a Command Prompt and ensure that 'wget.exe' is in the current working directory. An easy way to open up a Command Prompt is to *Shift+Right-click* in the directory you have downloaded 'wget' to and click *Open command window here*.
In the command prompt, enter '`wget www.unimelb.edu.au`', but do not press *Enter* yet. We need to prepare Wireshark to capture the traffic.

Step 3

Open Wireshark and prepare to start a new capture. Under *Capture options*, it will be useful to enter a capture filter to filter out background traffic irrelevant to our measurements. Enter '`tcp port http`' in the capture filter text box. Make sure *Promiscous mode* is unchecked. *Promiscious mode* tells Wireshark to capture all packets that travel through our network interface—even those not intended for your computer.
If you have trouble capturing packets, first ensure you have the correct network interface selected—you may have to experiment a bit to see which interface is carrying the internet traffic on your computer. Alternatively, try clearing the capture filter and applying a *display* filter instead. The *display* filter syntax is a little different to the capture filter syntax. You will want to enter '`tcp.port eq 80`' instead in the *Filter* text box in the Wireshark main window. If you are still unable to see packets being captured, try using port 8000 in the filters. The Unimelb proxy works on this port and may be causing problems.

Step 4

Once you have verified you can capture packets in Wireshark, start a new capture. Almost immediately after, hit *Enter* in the command prompt to download the Unimelb web page to your computer. Once the download has finished, stop the capture in Wireshark. The idea is to capture as little background network traffic as possible.
The packets you should have captured should start with a short TCP packet described as a SYN, which indicate the beginning of a connection. They will be followed by mostly longer packets in the middle (of roughly 1 to 1.5KB), of which the last one is a HTTP packet. This is the main portion of the download. And they will likely end with a short TCP packet that is part of ending the connection.

Step 5

In Wireshark, under the *File* menu, *export packet dissections* as a *CSV* file. Open your saved packet capture in your favourite spreadsheet software, and answer the questions below.

5 Questions

Once you have finished answering these questions, do you think you will have similar results each time? Can your dissimilarities, if any, be explained?

1. What is the source IP address of your computer? What is the destination IP address of the web page you requested?
2. How much traffic in bytes in total was received and transmitted in your request for the Unimelb web page? What is the percentage of traffic that originated from your computer, and what is the percentage of traffic that was sent by the remote host?
3. How do these sizes compare with the file size of the web page you downloaded? Estimate the download protocol overhead, or percentage of the download bytes taken up by protocol overhead.
4. What are the protocol overheads used for? Why are they useful? Aren't they a waste of space?
5. Are you able to identify other packets apart from the HTTP GET request and response packets in your capture? What might these packets be used for?
6. How do the protocol overheads relate to the networking layering architecture you have seen (or will see) in class? Is this architecture efficient? What are the advantages of this 'layered-cake' architecture? What are the disadvantages?
7. **Bonus:** Calculate the inter-arrival times (IAT) of the packets, i.e. the time between each packet arrival. Plot the number of packets against the IAT using a histogram. What sort of distribution do you observe? You may need to capture many more HTTP packets (this time using a web browser, for example) for a proper distribution to be seen.