FIT1047 Tutorial 8 – Sample Solution

Topics and goals

- Network layers and protocols, learn about addresses at various layers
- Application layer – HTTP
- Familiarise yourself with some of the networking tools

Instructions

The tasks are supposed to be done in groups of two or three students.

Task 0: Basic knowledge

Briefly explain the following terms:

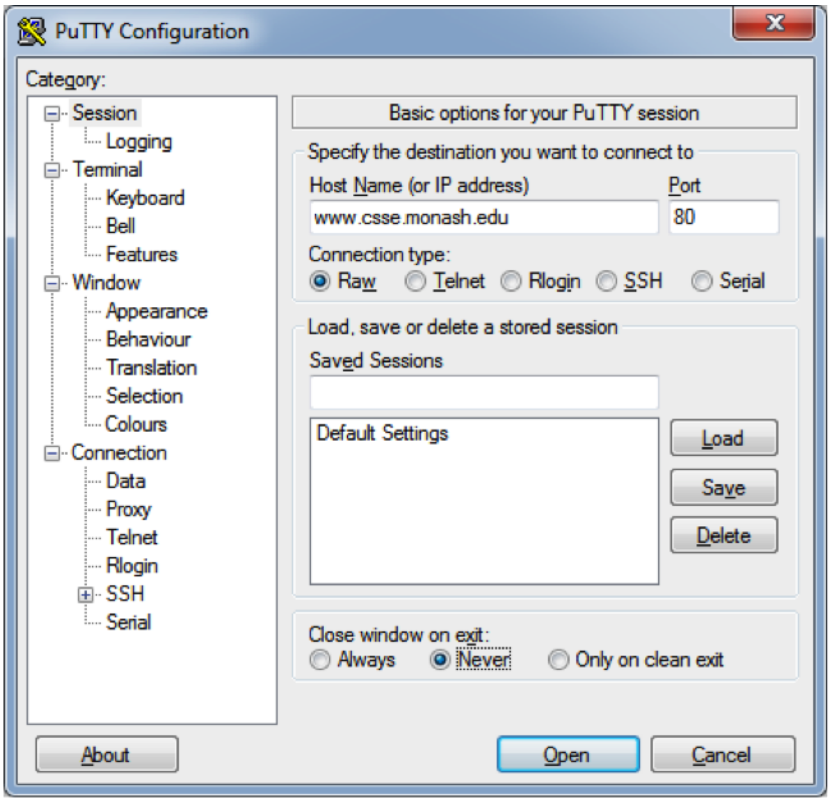
1. Bit rate, bandwidth

   Bit rate: the numbers of bits that can be transmitted over a data communications channel per time unit, usually measured in bits per second (or kilobits, megabits, gigabits per second). Bandwidth: strictly speaking, the width of a range of frequencies (i.e. highest minus lowest frequency). E.g. each channel in WiFi has a bandwidth of 20 MHz. But it's often used as a synonym for bit rate.

2. Latency

   The time it takes for a message from being sent to being received. Note that a network can have a very high bit rate but still feel slow: e.g. consider using a satellite connection for a video conference: you can stream very high quality video over the satellite connection, but due to the high latency, you could experience a delay of several seconds, which makes the connection almost unusable for real-time communication. This is also often a problem when playing real-time online video games.

3. Networks layers, PDU, encapsulation



Specify the destination you want to connect to

Host Name (or IP address): www.csse.monash.edu

Port: 80

Connection type: [✓ Raw] Telnet Rlogin SSH Serial

Load, save or delete a stored session

Saved Sessions:

Default Settings

Close window on exit: Always [✓ Never] Only on clean exit

Figure 1: The PuTTY telnet client

Task 1: "Browsing" the web

In this exercise, you will directly "talk" to a web server, without using a web browser. We will use a **telnet** program, which can establish a connection to a server and then send and receive textual data over that connection.

Figure 1 shows a screen shot of the PuTTY telnet client, which is installed on the Monash lab computers. The screenshot shows you how to set it up for this exercise: check the host name, port, connection type and "close window on exit".

When you click "Open", a black window will appear in which you can enter commands that are sent to the web server at `www.csse.monash.edu`. Your task is to request a particular page, which is at `/~guidot/` on that server.

Note: If you are using Mac OS or Linux on your own machine, open a terminal window and type `telnet www.csse.monash.edu 80` instead.

1. The most basic form of request (defined in an older version of HTTP) looks like this:

   ```
   GET /~guidot/
   ```

   Try entering this request and have a look at the result.

2. The version of HTTP currently in use is 1.1. A request in this version must be composed of a `GET` line (ending in `HTTP/1.1` to specify the newer protocol version), followed by a `Host:` line that identifies the host that the request is sent to (in this case it should be `www.csse.monash.edu`), followed by an empty line. Try out this type of request as well. What difference do you notice?

   There are two main differences: (1) the response contains additional headers that deliver some metadata about the requested web page; (2) the connection is not closed immediately, but remains open for a few seconds, which enables the browser (or us) to make another request without first opening a new connection. This behaviour was introduced in order to make HTTP requests more efficient for modern web pages, which often need to request hundreds of files ("assets", such as JavaScript, CSS and images) in order to display a single page.

Task 2: Packet sniffing

A packet analyser (sometimes also called "packet sniffer") is a program that can log all packets that are received and transmitted over a network interface. We will be using *Wireshark*, a very popular open-source tool for packet analysis. You can download it on your own computer from `www.wireshark.org`, or use the version installed on the Monash lab computers.

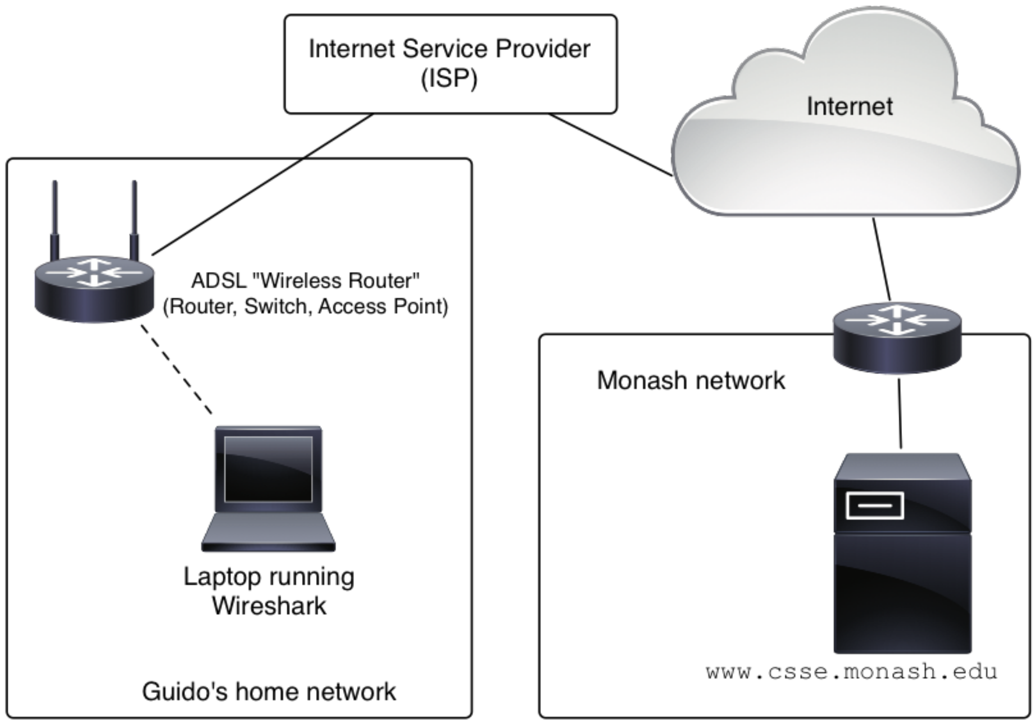
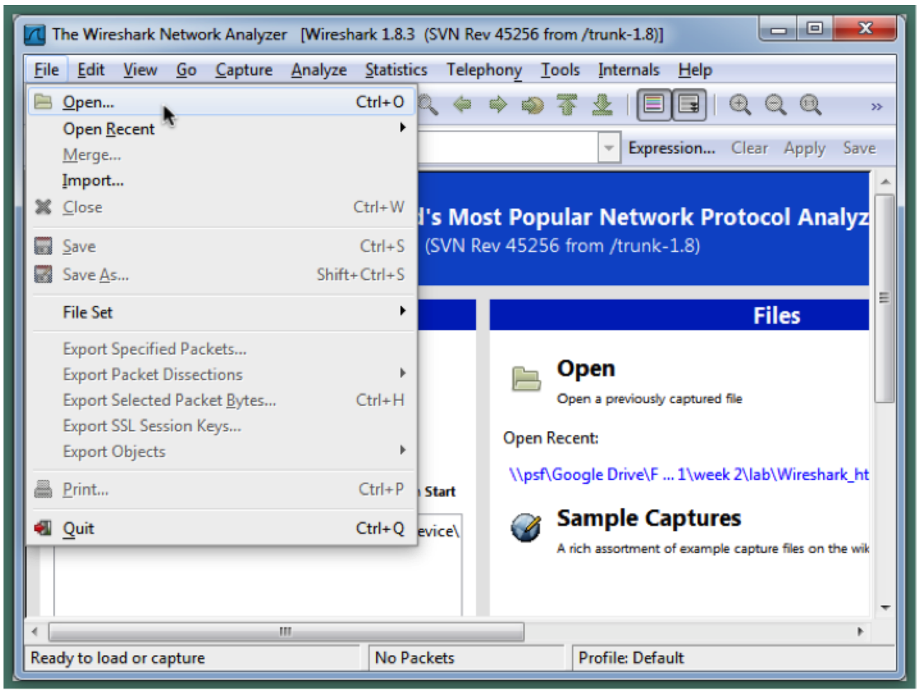


Figure 2: The network where the traffic was captured

This week, we will analyse a sequence of packets captured on Guido's network at home. You can download the log file that Wireshark produced from the FIT1047 Moodle site (week 7). The file is called `Wireshark_http_example.pcap`. The diagram in Figure 2 explains how Guido's computer is connected to the Monash web server.

1. Start Wireshark and load the capture file (see Fig. 3). After opening the file, the main window should look like the one in Fig. 4.
2. Select "frame" number 6 (as in Fig. 4). This frame shows a request sent from Guido's home computer to the web server at `www.csse.monash.edu`, requesting his homepage
3. Familiarise yourself with the three main sections (panes) of the Wireshark window:
    - The packet list pane displays a summary of each packet captured. When you click on a packet here, the other two panes are updated with the details for that packet.
    - The packet details pane below shows information about the selected packet.
    - The packet bytes pane displays the raw data for the selected packet. It highlights the data for the field that is selected in the packet details pane.



The Wireshark Network Analyzer [Wireshark 1.8.3 (SVN Rev 45256 from /trunk-1.8)]
File
Edit
View
Go
Capture
Analyze
Statistics
Telephony
Tools
Internals
Help
Open... Ctrl+O
Open Recent
Merge...
Import...
Close Ctrl+W
Save Ctrl+S
Save As... Shift+Ctrl+S
File Set
Export Specified Packets...
Export Packet Dissections
Export Selected Packet Bytes... Ctrl+H
Export SSL Session Keys...
Export Objects
Print... Ctrl+P
Quit Ctrl+Q
Expression...
Clear
Apply
Save
's Most Popular Network Protocol Analyz
(SVN Rev 45256 from /trunk-1.8)
Files
Open
Open a previously captured file
Open Recent:
\\psf\Google Drive\F ... 1\week 2\lab\Wireshark_ht
Sample Captures
A rich assortment of example capture files on the wik
Start
evice\
Ready to load or capture
No Packets
Profile: Default

Figure 3: Wireshark File menu

Navigate between the panes and explore the relationships between the displayed pieces of information.

4. Click on frame 6.

   a) How long is it in bytes?

      432 bytes

   b) Which application layer protocol does it use?

      HTTP

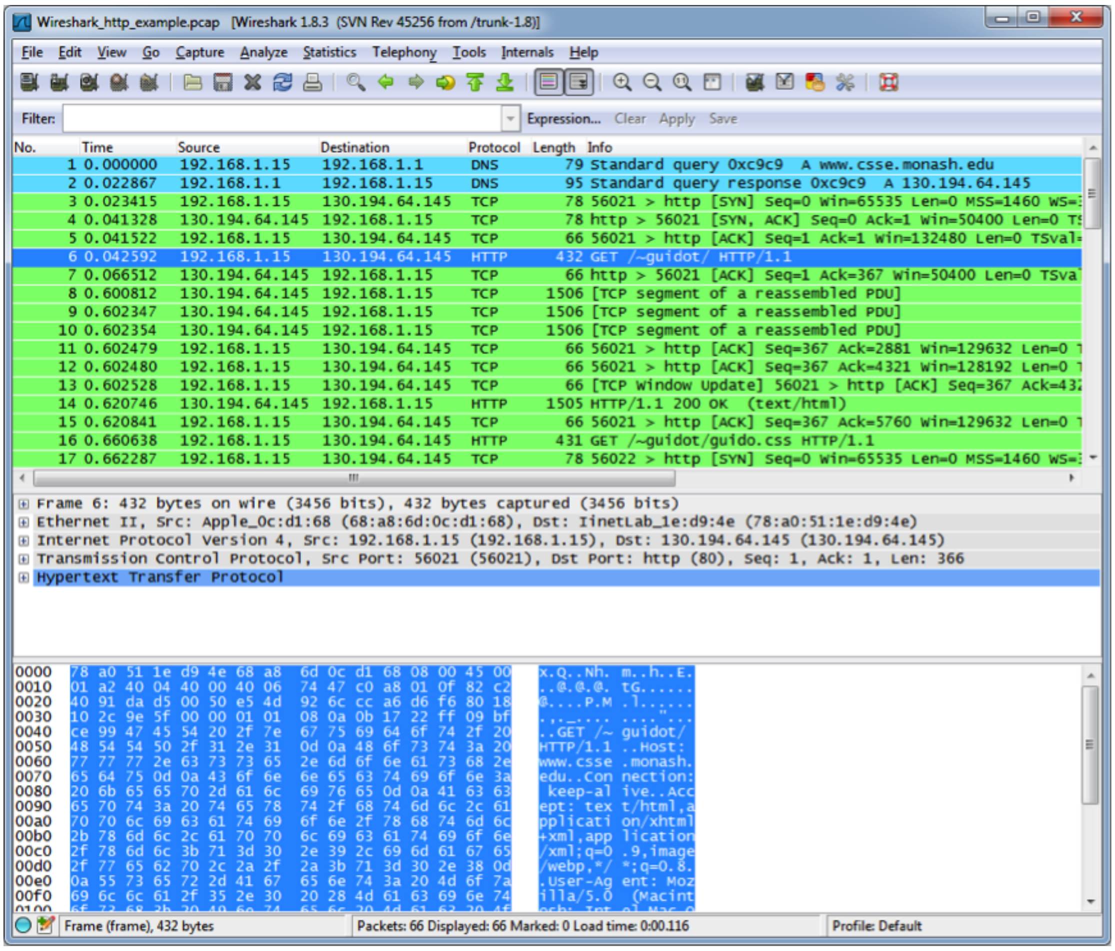
5. There are five lines in the packet details pane for frame 6, each of which can be expanded by clicking on the "+" symbol.

   a) Can you identify what they stand for?

      From top to bottom: the entire frame captured by Wireshark; the data link layer; the network layer; the transport layer; the application layer.

   b) Which protocols are being used?

      From top to bottom: Ethernet; IP version 4; TCP; HTTP.


Wireshark_http_example.pcap [Wireshark 1.8.3 (SVN Rev 45256 from /trunk-1.8)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.    Time     Source          Destination     Protocol Length Info
     1 0.000000 192.168.1.15    192.168.1.1     DNS        79 Standard query 0xc9c9  A www.csse.monash.edu
     2 0.022867 192.168.1.1     192.168.1.15    DNS        95 Standard query response 0xc9c9  A 130.194.64.145
     3 0.023415 192.168.1.15    130.194.64.145  TCP        78 56021 > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=
     4 0.041328 130.194.64.145  192.168.1.15    TCP        78 http > 56021 [SYN, ACK] Seq=0 Ack=1 Win=50400 Len=0 TS
     5 0.041522 192.168.1.15    130.194.64.145  TCP        66 56021 > http [ACK] Seq=1 Ack=1 Win=132480 Len=0 TSval=
     6 0.042592 192.168.1.15    130.194.64.145  HTTP      432 GET /~guidot/ HTTP/1.1
     7 0.066512 130.194.64.145  192.168.1.15    TCP        66 http > 56021 [ACK] Seq=1 Ack=367 Win=50400 Len=0 TSva
     8 0.600812 130.194.64.145  192.168.1.15    TCP      1506 [TCP segment of a reassembled PDU]
     9 0.602347 130.194.64.145  192.168.1.15    TCP      1506 [TCP segment of a reassembled PDU]
    10 0.602354 130.194.64.145  192.168.1.15    TCP      1506 [TCP segment of a reassembled PDU]
    11 0.602479 192.168.1.15    130.194.64.145  TCP        66 56021 > http [ACK] Seq=367 Ack=2881 Win=129632 Len=0
    12 0.602480 192.168.1.15    130.194.64.145  TCP        66 56021 > http [ACK] Seq=367 Ack=4321 Win=128192 Len=0
    13 0.602528 192.168.1.15    130.194.64.145  TCP        66 [TCP Window Update] 56021 > http [ACK] Seq=367 Ack=43
    14 0.620746 130.194.64.145  192.168.1.15    HTTP     1505 HTTP/1.1 200 OK  (text/html)
    15 0.620841 192.168.1.15    130.194.64.145  TCP        66 56021 > http [ACK] Seq=367 Ack=5760 Win=129632 Len=0
    16 0.660638 192.168.1.15    130.194.64.145  HTTP      431 GET /~guidot/guido.css HTTP/1.1
    17 0.662287 192.168.1.15    130.194.64.145  TCP        78 56022 > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=
```

```
⊞ Frame 6: 432 bytes on wire (3456 bits), 432 bytes captured (3456 bits)
⊞ Ethernet II, Src: Apple_0c:d1:68 (68:a8:6d:0c:d1:68), Dst: IinetLab_1e:d9:4e (78:a0:51:1e:d9:4e)
⊞ Internet Protocol Version 4, Src: 192.168.1.15 (192.168.1.15), Dst: 130.194.64.145 (130.194.64.145)
⊞ Transmission Control Protocol, Src Port: 56021 (56021), Dst Port: http (80), Seq: 1, Ack: 1, Len: 366
⊞ Hypertext Transfer Protocol
```

```
0000  78 a0 51 1e d9 4e 68 a8  6d 0c d1 68 08 00 45 00   x.Q..Nh. m..h..E.
0010  01 a2 40 04 40 00 40 06  74 47 c0 a8 01 0f 82 c2   ..@.@.@. tG......
0020  40 91 da d5 00 50 e5 4d  92 6c cc a6 d6 f6 80 18   @....P.M .l......
0030  10 2c 9e 5f 00 00 01 01  08 0a 0b 17 22 ff 09 bf   .,._.... ...."...
0040  ce 99 47 45 54 20 2f 7e  67 75 69 64 6f 74 2f 20   ..GET /~ guidot/
0050  48 54 54 50 2f 31 2e 31  0d 0a 48 6f 73 74 3a 20   HTTP/1.1 ..Host:
0060  77 77 77 2e 63 73 73 65  2e 6d 6f 6e 61 73 68 2e   www.csse .monash.
0070  65 64 75 0d 0a 43 6f 6e  6e 65 63 74 69 6f 6e 3a   edu..Con nection:
0080  20 6b 65 65 70 2d 61 6c  69 76 65 0d 0a 41 63 63    keep-al ive..Acc
0090  65 70 74 3a 20 74 65 78  74 2f 68 74 6d 6c 2c 61   ept: tex t/html,a
00a0  70 70 6c 69 63 61 74 69  6f 6e 2f 78 68 74 6d 6c   pplicati on/xhtml
00b0  2b 78 6d 6c 2c 61 70 70  6c 69 63 61 74 69 6f 6e   +xml,app lication
00c0  2f 78 6d 6c 3b 71 3d 30  2e 39 2c 69 6d 61 67 65   /xml;q=0 .9,image
00d0  2f 77 65 62 70 2c 2a 2f  2a 3b 71 3d 30 2e 38 0d   /webp,*/ *;q=0.8.
00e0  0a 55 73 65 72 2d 41 67  65 6e 74 3a 20 4d 6f 7a   .User-Ag ent: Moz
00f0  69 6c 6c 61 2f 35 2e 30  20 28 4d 61 63 69 6e 74   illa/5.0  (Macint
```

```
Frame (frame), 432 bytes    Packets: 66 Displayed: 66 Marked: 0 Load time: 0:00.116    Profile: Default
```

Figure 4: Wireshark main window

c) What are the names of the PDUs for each protocol?

   Ethernet: frame; IP: packet; TCP: segment; HTTP: message.

d) Recall that each protocol layer encapsulates the message from the layer above and adds a header. What are the sizes of the headers for each PDU used in frame 6?

   You can find the size by clicking on the relevant protocol line and counting the number of bytes highlighted in the bottom pane. Some protocols also tell you the size when you open the details view. Ethernet: 14 bytes; IP: 20 bytes; 32 bytes; HTTP: we can consider the entire contents to be the actual message, there are no additional headers (such as the ones for TCP or IP).