COMP90007 Internet Technologies Workshop

Week 12 - Lab

Ruixi
huor@student.unimelb.edu.au

Slides

https://github.com/SiaHuo/COMP90007Workshops

1 Objectives

- To learn about using the telnet to establish transport layer connection-oriented communication with a remote host.
- 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 and how requests are made in a web browser and how responses are returned.

Tools

There are two software requirements for this lab:

- **Wireshark**

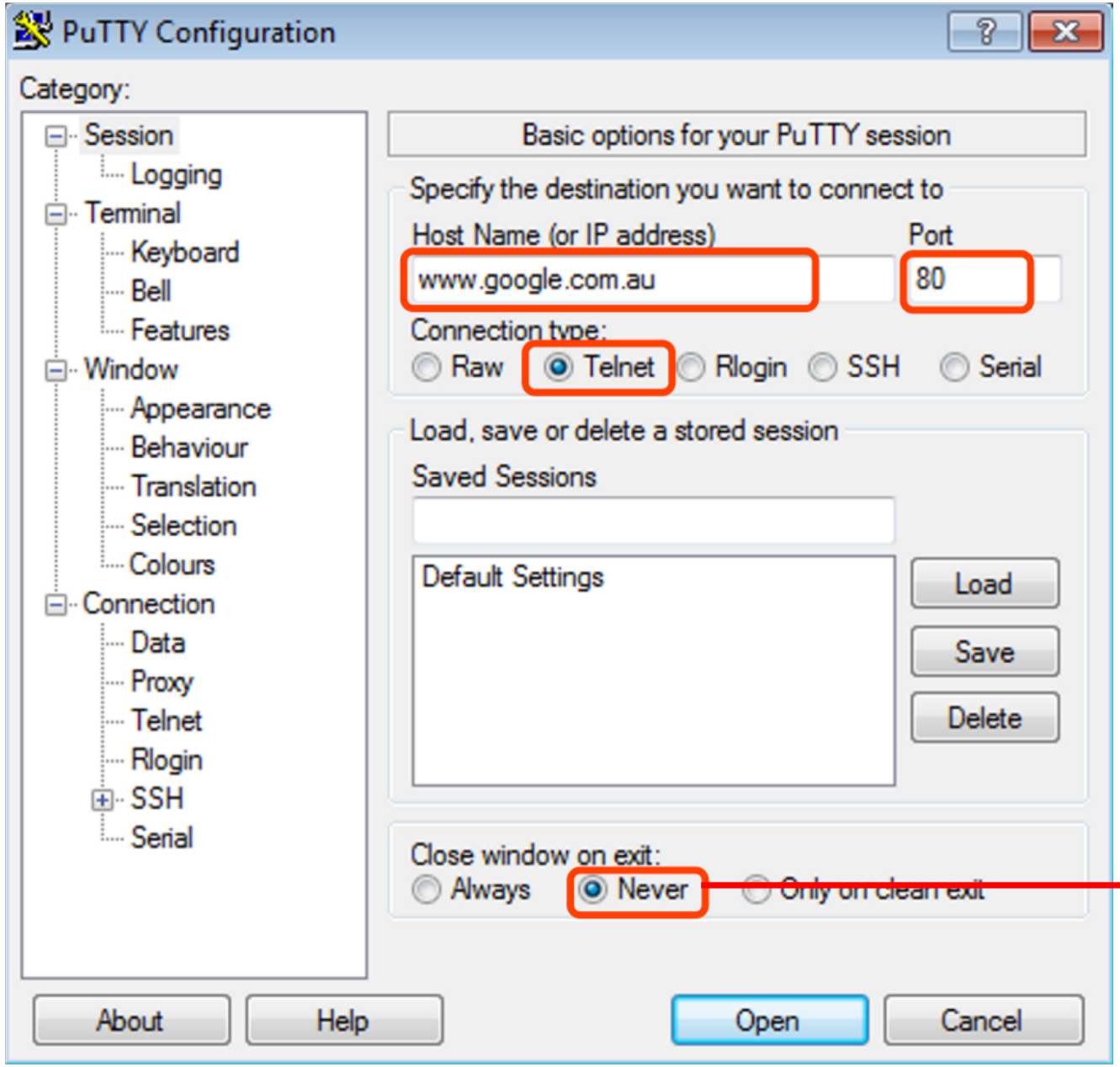
  Wireshark is a sniffing tool that allows the user to examine a packet trace, that is, a record of network traffic over time. It can be downloaded at http://www.wireshark.org/download.html.

- **PuTTY or an equivalent telnet client** **Windows user only**

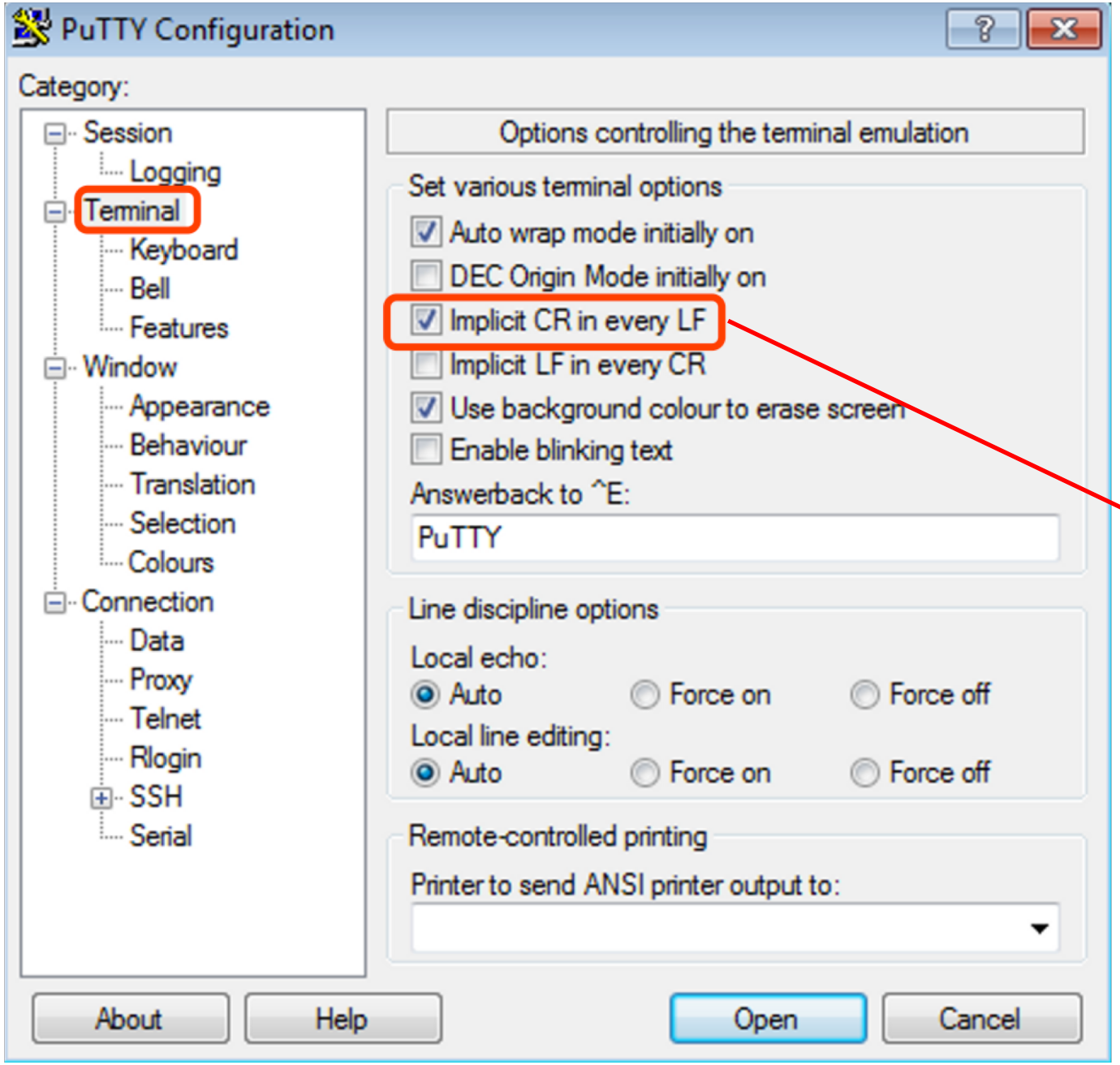
  PuTTY is Windows telnet and SSH client which is commonly used to connect and login to remote machines. We will be using PuTTY as a telnet client in this lab. PuTTY can be downloaded from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.

  If you are running OS X or Linux, a telnet client should be pre-installed on your operating systems. It can be run by typing `telnet` on your command line terminal.

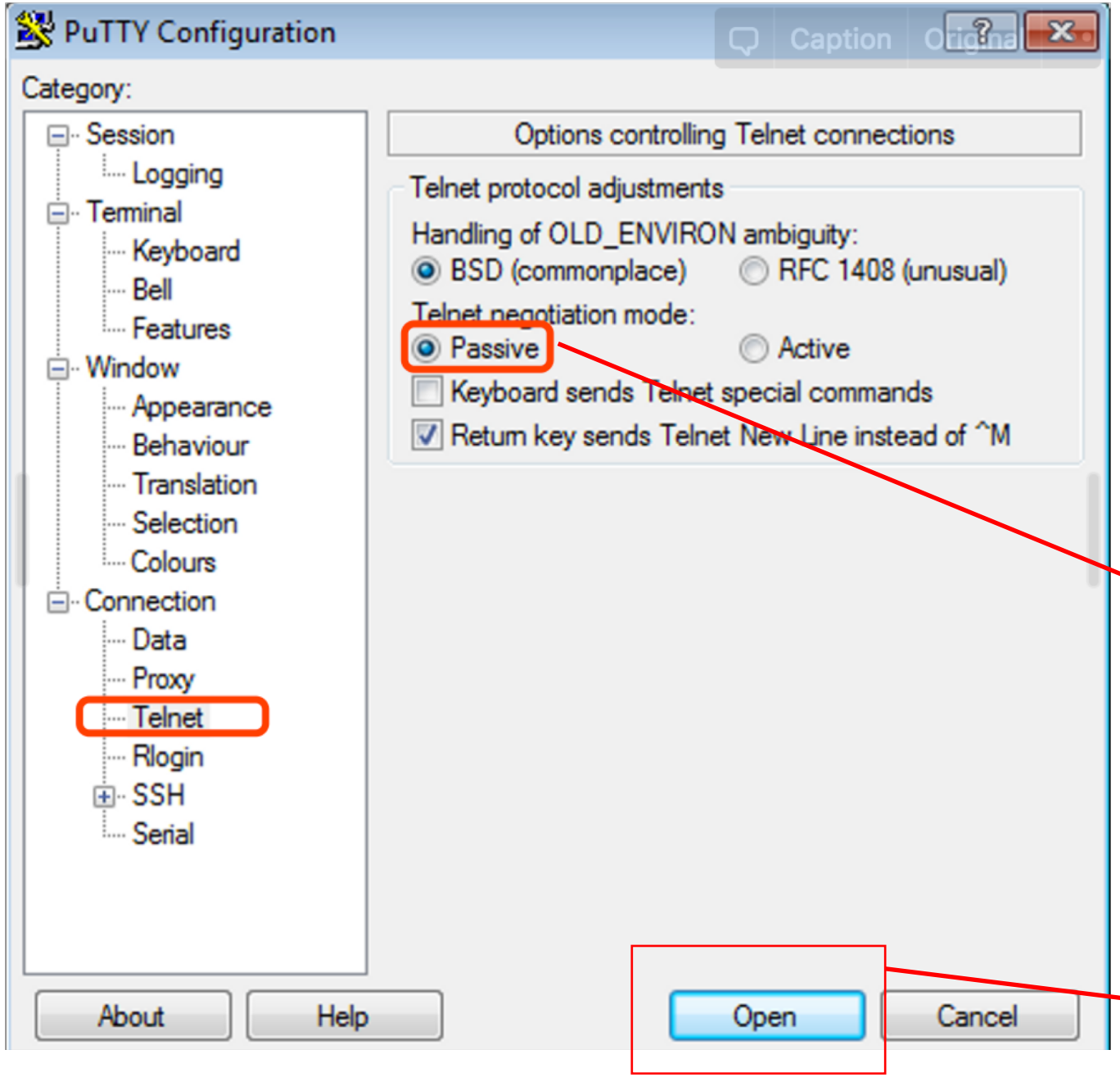
Windows user



This will make sure you can read the output after the web server closes the connection.



This will ensure proper line breaks are displayed, stemming from differences with Unix (\n) and Windows (\r\n) line break conventions.



The HTTP requests don't seem to work if you don't set this option.

DON NOT Click OPEN NOW. Prepare ur wireshark

Wireshark · Capture Interfaces

Input | Output | Options

Interface	Traffic	Link-layer Header	Promisc	Snaplen (B)	Buffer (MB)	Monitor	Ca
Wi-Fi: en0		Ethernet		default	2		tc
p2p0		Raw IP		default	2	—	
awdl0		Ethernet		default	2	—	
Thunderbolt Bridge: bridge0		Ethernet		default	2	—	
utun0		BSD loopback		default	2	—	
Thunderbolt 1: en1		Ethernet		default	2	—	
Thunderbolt 2: en2		Ethernet		default	2	—	
Loopback: lo0		BSD loopback		default	2	—	
gif0		BSD loopback		default	2	—	
stf0		BSD loopback		default	2	—	

Enable promiscuous mode on all interfaces

Manage Interfaces...

Capture filter for selected interfaces: tcp port http

Compile BPFs

Help

Close

Start

www.google.com.au - PuTTY

```
GET / HTTP/1.0


GET / HTTP/1.0
Host: www.google.com.au


GET / HTTP/1.1
Host: www.google.com.au
```

Hit enter twice

A Host header field must be sent in all HTTP/1.1 request messages. A 400 (Bad Request) status code will be sent to any HTTP/1.1 request message that lacks a Host header field or contains more than one.

```
my_new_host_name:inetutils-1.9.4 Sia$ telnet www.google.com.au 80
Trying 216.58.196.131...
Connected to www.google.com.au.
Escape character is '^]'.
GET / HTTP/1.0
Host:www.google.com.au

HTTP/1.0 200 OK
Date: Sun, 14 Oct 2018 22:32:59 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2018-10-14-22; expires=Tue, 13-Nov-2018 22:32:59 GMT; path=/; domain=.google.com.au
Set-Cookie: NID=142=UAbWu7k8InIezDeyXoAYIDxNoz4ALjon4taLqlL6Y1ZpMh1HGSZN4nB79Y_foqSGFzQgr95UH_31R7I47G4wrmRjy
tT0adTGtss91Q8V1OMV3Rz5aFAQ7hjDn29bny-jQh0RHShP9dhht_cAuvFH1VpABWU0O4CL1kvm7B8LvO4; expires=Mon, 15-Apr-2019
22:32:59 GMT; path=/; domain=.google.com.au; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
```

Optional for HTTP 1.0

Status: success

Header Information

```
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en-AU"><head><meta content="text
/html; charset=UTF-8" http-equiv="Content-Type"><meta content="/images/branding/googleg/1x/googleg_standard_c
olor_128dp.png" itemprop="image"><title>Google</title><script nonce="3l0BB2yAmzLv0YIa34iASQ==">(function(){wi
ndow.google={kEI:'G8TDW4yCLtSj-QbSxIqABQ',kEXPI:'0,18168,1335579,57,1957,583,434,282,1124,1225,731,142,183,64
,45,183,154,207,422,96,302,492,62,1,138,177,201,112,2337212,106,50,32,32929,1294,12383,4855,32692,2074,13173
,867,316,11847,5281,1100,853,2482,2,2,4604,2197,364,1217,2102,2314,3191,224,2218,260,1027,4080,575,1119,2,130
6,2222,210,58,2,4,190,1107,1712,2611,2096,1294,8,1569,774,1232,883,134,282,2,1124,593,2744,525,22,437,162,5,2
,2,1316,649,526,3,774,157,656,659,283,556,3180,69,699,243,108,338,10,116,1110,234,386,8,804,199,81,7,3,26,462
```

HTML

Wi-Fi: en0 (tcp port http)

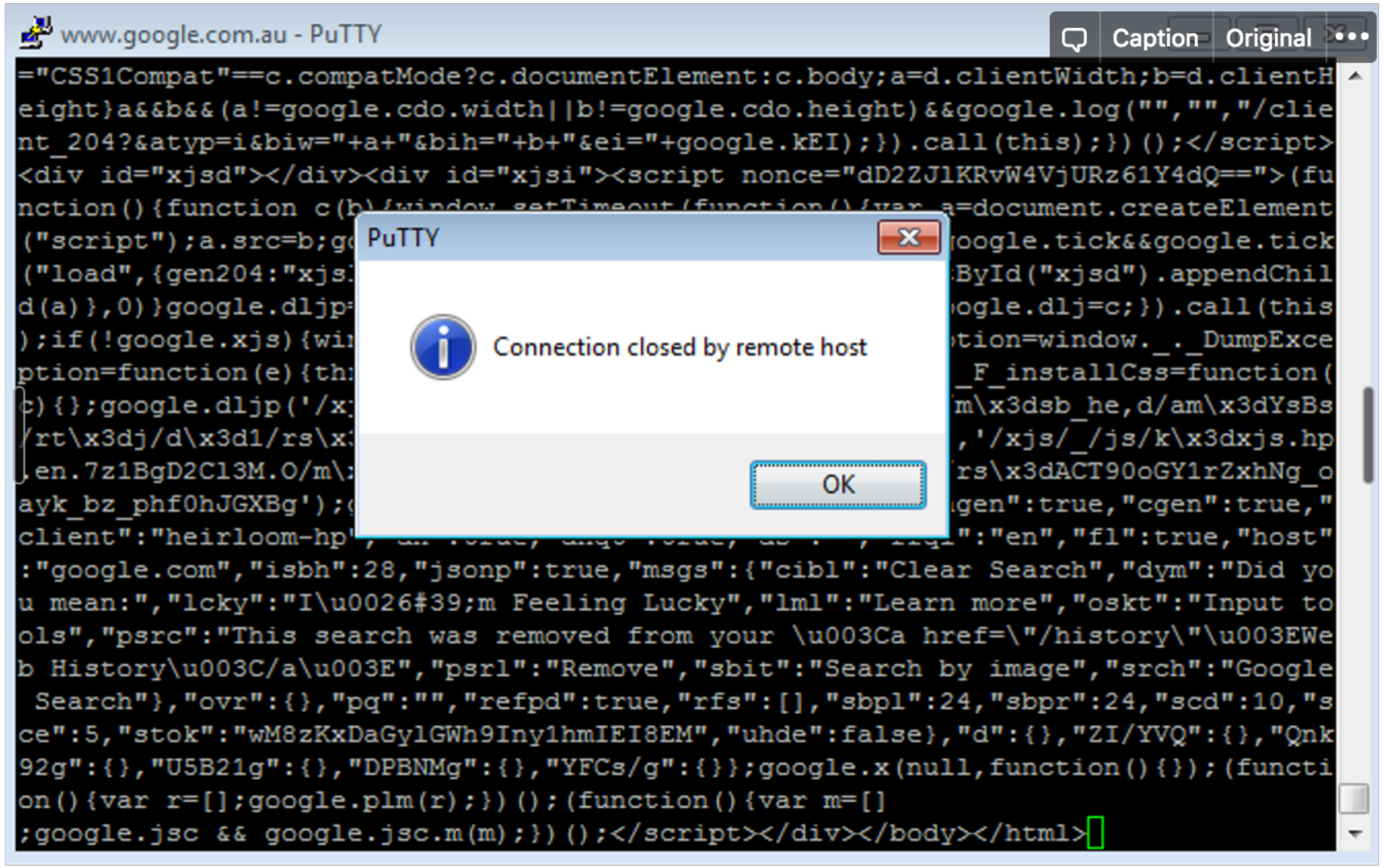
Apply a display filter ... <⌘/>

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.12.58.26	216.58.196.131	TCP	78	58821 → 80 [SYN] Seq=0 Win=65535 Len=0 MS…
2	0.013981	216.58.196.131	10.12.58.26	TCP	74	80 → 58821 [SYN, ACK] Seq=0 Ack=1 Win=601…
3	0.014073	10.12.58.26	216.58.196.131	TCP	66	58821 → 80 [ACK] Seq=1 Ack=1 Win=132384 L…
4	16.657495	10.12.58.26	syd15s04-in-f3.1e1…	TCP	82	[TCP segment of a reassembled PDU]
5	16.671503	syd15s04-in-f3.1e1…	10.12.58.26	TCP	66	80 → 58821 [ACK] Seq=1 Ack=17 Win=60416 L…
6	16.870326	10.12.58.26	syd15s04-in-f3.1e1…	HTTP	68	GET / HTTP 1.0
7	16.884169	syd15s04-in-f3.1e1…	10.12.58.26	TCP	66	80 → 58821 [ACK] Seq=1 Ack=19 Win=60416 L…
8	16.885189	syd15s04-in-f3.1e1…	10.12.58.26	TCP	1417	[TCP segment of a reassembled PDU]
9	16.885197	syd15s04-in-f3.1e1…	10.12.58.26	HTTP	427	HTTP/1.0 400 Bad Request (text/html)
10	16.885199	syd15s04-in-f3.1e1…	10.12.58.26	TCP	66	80 → 58821 [FIN, ACK] Seq=1713 Ack=19 Win…
11	16.885321	10.12.58.26	syd15s04-in-f3.1e1…	TCP	66	58821 → 80 [ACK] Seq=19 Ack=1713 Win=1306…
12	16.885321	10.12.58.26	syd15s04-in-f3.1e1…	TCP	66	58821 → 80 [ACK] Seq=19 Ack=1714 Win=1306…
13	16.886014	10.12.58.26	syd15s04-in-f3.1e1…	TCP	66	58821 → 80 [FIN, ACK] Seq=19 Ack=1714 Win…
14	16.899720	syd15s04-in-f3.1e1…	10.12.58.26	TCP	66	80 → 58821 [ACK] Seq=1714 Ack=20 Win=6041…

```
▶ Frame 6: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
▼ Ethernet II, Src: Apple_c6:12:53 (ac:bc:32:c6:12:53), Dst: All-HSRP-routers_98 (00:00:0c:07:ac:98)
  ▶ Destination: All-HSRP-routers_98 (00:00:0c:07:ac:98)
  ▶ Source: Apple_c6:12:53 (ac:bc:32:c6:12:53)
    Type: IPv4 (0x0800)
▼ Internet Protocol Version 4, Src: 10.12.58.26 (10.12.58.26), Dst: syd15s04-in-f3.1e100.net (216.58.196.131)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT)
    Total Length: 54
```

If you are using HTTP1.0 in PuTTY



Question 8

What are some of the differences between HTTP 1.0 and HTTP 1.1?

HTTP 1.0	HTTP 1.1
- Does not officially require a Host Header to be included.	- Requires a Host Header as per its specification
- Open a new connection for each request/ response pair, following which the connection would be closed. (leads to efficiency problems – TCP slow start).	- Allows persistent connections, allowing more than one request/ response on the same HTTP connection.

Mac user

Mac user only

The telnet was removed on macOS high Sierra

Step1: Install Homebrew

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Step2: Install telnet using homebrew

```
brew install telnet
```

Another method to use telnet

```
nc www.google.com.au 80
```

Finally, Play with telnet

```
telnet www.google.com.au 80
```

```
GET / HTTP/1.0
```

```
GET / HTTP/1.1
HOST: www.google.com.au
```

Hit enter twice

Sia — -bash — 80×24

```
Last login: Mon Oct 15 18:11:06 on ttys001
my_new_host_name:~ Sia$
```

telnet www.google.com.au 80

nc www.google.com.au 80

GET / HTTP/1.0

GET / HTTP/1.0
Host: www.google.com.au

GET / HTTP/1.1
Host: www.google.com.au

Hit enter twice

```
my_new_host_name:inetutils-1.9.4 Sia$ telnet www.google.com.au 80
Trying 216.58.196.131...
Connected to www.google.com.au.
Escape character is '^]'.
GET / HTTP/1.0
Host:www.google.com.au

HTTP/1.0 200 OK
Date: Sun, 14 Oct 2018 22:32:59 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2018-10-14-22; expires=Tue, 13-Nov-2018 22:32:59 GMT; path=/; domain=.google.com.au
Set-Cookie: NID=142=UAbWu7k8InIezDeyXoAYIDxNoz4ALjon4taLqlL6Y1ZpMh1HGSZN4nB79Y_foqSGFzQgr95UH_31R7I47G4wrmRjy
tT0adTGtss91Q8V1OMV3Rz5aFAQ7hjDn29bny-jQh0RHShP9dhht_cAuvFH1VpABWU0O4CL1kvm7B8LvO4; expires=Mon, 15-Apr-2019
22:32:59 GMT; path=/; domain=.google.com.au; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
```

Optional for HTTP 1.0

Status: success

Header Information

```
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en-AU"><head><meta content="text
/html; charset=UTF-8" http-equiv="Content-Type"><meta content="/images/branding/googleg/1x/googleg_standard_c
olor_128dp.png" itemprop="image"><title>Google</title><script nonce="3l0BB2yAmzLv0YIa34iASQ==">(function(){wi
ndow.google={kEI:'G8TDW4yCLtSj-QbSxIqABQ',kEXPI:'0,18168,1335579,57,1957,583,434,282,1124,1225,731,142,183,64
,45,183,154,207,422,96,302,492,62,1,138,177,201,112,2337212,106,50,32,32929,1294,12383,4855,32692,2074,13173
,867,316,11847,5281,1100,853,2482,2,2,4604,2197,364,1217,2102,2314,3191,224,2218,260,1027,4080,575,1119,2,130
6,2222,210,58,2,4,190,1107,1712,2611,2096,1294,8,1569,774,1232,883,134,282,2,1124,593,2744,525,22,437,162,5,2
,2,1316,649,526,3,774,157,656,659,283,556,3180,69,699,243,108,338,10,116,1110,234,386,8,804,199,81,7,3,26,462
```

HTML

Wi-Fi: en0 (tcp port http)

Apply a display filter ... <⌘/>

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.12.58.26	216.58.196.131	TCP	78	58821 → 80 [SYN] Seq=0 Win=65535 Len=0 MS…
2	0.013981	216.58.196.131	10.12.58.26	TCP	74	80 → 58821 [SYN, ACK] Seq=0 Ack=1 Win=601…
3	0.014073	10.12.58.26	216.58.196.131	TCP	66	58821 → 80 [ACK] Seq=1 Ack=1 Win=132384 L…
4	16.657495	10.12.58.26	syd15s04-in-f3.1e1…	TCP	82	[TCP segment of a reassembled PDU]
5	16.671503	syd15s04-in-f3.1e1…	10.12.58.26	TCP	66	80 → 58821 [ACK] Seq=1 Ack=17 Win=60416 L…
6	16.870326	10.12.58.26	syd15s04-in-f3.1e1…	HTTP	68	GET / HTTP 1.0
7	16.884169	syd15s04-in-f3.1e1…	10.12.58.26	TCP	66	80 → 58821 [ACK] Seq=1 Ack=19 Win=60416 L…
8	16.885189	syd15s04-in-f3.1e1…	10.12.58.26	TCP	1417	[TCP segment of a reassembled PDU]
9	16.885197	syd15s04-in-f3.1e1…	10.12.58.26	HTTP	427	HTTP/1.0 400 Bad Request (text/html)
10	16.885199	syd15s04-in-f3.1e1…	10.12.58.26	TCP	66	80 → 58821 [FIN, ACK] Seq=1713 Ack=19 Win…
11	16.885321	10.12.58.26	syd15s04-in-f3.1e1…	TCP	66	58821 → 80 [ACK] Seq=19 Ack=1713 Win=1306…
12	16.885321	10.12.58.26	syd15s04-in-f3.1e1…	TCP	66	58821 → 80 [ACK] Seq=19 Ack=1714 Win=1306…
13	16.886014	10.12.58.26	syd15s04-in-f3.1e1…	TCP	66	58821 → 80 [FIN, ACK] Seq=19 Ack=1714 Win…
14	16.899720	syd15s04-in-f3.1e1…	10.12.58.26	TCP	66	80 → 58821 [ACK] Seq=1714 Ack=20 Win=6041…

```
▶ Frame 6: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
▼ Ethernet II, Src: Apple_c6:12:53 (ac:bc:32:c6:12:53), Dst: All-HSRP-routers_98 (00:00:0c:07:ac:98)
  ▶ Destination: All-HSRP-routers_98 (00:00:0c:07:ac:98)
  ▶ Source: Apple_c6:12:53 (ac:bc:32:c6:12:53)
    Type: IPv4 (0x0800)
▼ Internet Protocol Version 4, Src: 10.12.58.26 (10.12.58.26), Dst: syd15s04-in-f3.1e100.net (216.58.196.131)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT)
    Total Length: 54
```

```
my_new_host_name:inetutils-1.9.4 Sia$ telnet www.google.com.au 80
Trying 216.58.196.131...
Connected to www.google.com.au.
Escape character is '^]'.
GET / HTTP/1.0
Host:www.google.com.au

HTTP/1.0 200 OK
Date: Sun, 14 Oct 2018 22:32:59 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2018-10-14-22; expires=Tue, 13-Nov-2018 22:32:59 GMT; path=/; domain=.google.com.au
Set-Cookie: NID=142=UAbWu7k8InIezDeyXoAYIDxNoz4ALjon4taLqlL6Y1ZpMh1HGSZN4nB79Y_foqSGFzQgr95UH_31R7I47G4wrmRjy
tT0adTGtss91Q8V1OMV3Rz5aFAQ7hjDn29bny-jQh0RHShP9dhht_cAuvFH1VpABWU0O4CL1kvm7B8LvO4; expires=Mon, 15-Apr-2019
22:32:59 GMT; path=/; domain=.google.com.au; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding

<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en-AU"><head><meta content="text
/html; charset=UTF-8" http-equiv="Content-Type"><meta content="/images/branding/googleg/1x/googleg_standard_c
olor_128dp.png" itemprop="image"><title>Google</title><script nonce="3l0BB2yAmzLv0YIa34iASQ==">(function(){wi
ndow.google={kEI:'G8TDW4yCLtSj-QbSxIqABQ',kEXPI:'0,18168,1335579,57,1957,583,434,282,1124,1225,731,142,183,64
,45,183,154,207,422,96,302,492,62,1,138,177,201,112,2337212,106,50,32,329294,1294,12383,4855,32692,2074,13173
,867,316,11847,5281,1100,853,2482,2,2,4604,2197,364,1217,2102,2314,3191,224,2218,260,1027,4080,575,1119,2,130
6,2222,210,58,2,4,190,1107,1712,2611,2096,1294,8,1569,774,1232,883,134,282,2,1124,593,2744,525,22,437,162,5,2
,2,1316,649,526,3,774,157,656,659,283,556,3180,69,699,243,108,338,10,116,1110,234,386,8,804,199,81,7,3,26,462
```

Question 2

Open up the specification on HTTP/1.0 at http://tools.ietf.org/html/rfc1945. Under section 9, what do the different categories of status codes 2xx, 3xx, 4xx and 5xx refer to?

Refer to RFC 1945 section 9. From the contents page:

9.1 Informational 1xx 32
9.2 Successful 2xx 32
9.3 Redirection 3xx 34
9.4 Client Error 4xx 35
9.5 Server Error 5xx 37

Question 3

In Wireshark, right click on a HTTP packet and click *Follow TCP stream*. Here you can observe the connection-oriented nature of the communication between the host and your computer. Play around and explore with this window, and when you're ready click *Close*.

A display filter would have been automatically applied to filter the TCP connection associated with the HTTP request. Click any packet and expand the *Internet Protocol Version 4* field in the middle section of Wireshark. Correlate the fields with Figure 5.

On the menu bar on top, click *Statistics* $\Rightarrow$ *Flow Graph*. Select *Displayed packets* under *Choose packets* and click *OK*. This makes it easier to see the packets being transferred back and forth. Can you identify the packets corresponding to the TCP 3-way handshake and connection teardown? How do the `Seq` numbers and `Ack` numbers work? Confirm with your answer with your tutor.

Wi-Fi: en0 (tcp port http)

Apply a display filter ... <⌘/>

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.12.58.26	216.58.196.131	TCP	78	58821 → 80 [SYN] Seq=0 Win=65535 Len=0 MS…
2	0.013981	216.58.196.131	10.12.58.26	TCP	74	80 → 58821 [SYN, ACK] Seq=0 Ack=1 Win=601…
3	0.014073	10.12.58.26	216.58.196.131	TCP	66	58821 → 80 [ACK] Seq=1 Ack=1 Win=132384 L…
4	16.657495	10.12.58.26	syd15s04-in-f3.1e1…	TCP	82	[TCP segment of a reassembled PDU]
5	16.671503	syd15s04-in-f3.1e1…	10.12.58.26	TCP	66	80 → 58821 [ACK] Seq=1 Ack=17 Win=60416 L…
6	16.870326	10.12.58.26	syd15s04-in-f3.1e1…	HTTP	68	GET / HTTP 1.0
7	16.884169	syd15s04-in-f3.1e1…	10.12.58.26	TCP	66	80 → 58821 [ACK] Seq=1 Ack=19 Win=60416 L…
8	16.885189	syd15s04-in-f3.1e1…	10.12.58.26	TCP	1417	[TCP segment of a reassembled PDU]
9	16.885197	syd15s04-in-f3.1e1…	10.12.58.26	HTTP	427	HTTP/1.0 400 Bad Request (text/html)
10	16.885199	syd15s04-in-f3.1e1…	10.12.58.26	TCP	66	80 → 58821 [FIN, ACK] Seq=1713 Ack=19 Win…
11	16.885321	10.12.58.26	syd15s04-in-f3.1e1…	TCP	66	58821 → 80 [ACK] Seq=19 Ack=1713 Win=1306…
12	16.885321	10.12.58.26	syd15s04-in-f3.1e1…	TCP	66	58821 → 80 [ACK] Seq=19 Ack=1714 Win=1306…
13	16.886014	10.12.58.26	syd15s04-in-f3.1e1…	TCP	66	58821 → 80 [FIN, ACK] Seq=19 Ack=1714 Win…
14	16.899720	syd15s04-in-f3.1e1…	10.12.58.26	TCP	66	80 → 58821 [ACK] Seq=1714 Ack=20 Win=6041…

```
▶ Frame 6: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
▼ Ethernet II, Src: Apple_c6:12:53 (ac:bc:32:c6:12:53), Dst: All-HSRP-routers_98 (00:00:0c:07:ac:98)
  ▶ Destination: All-HSRP-routers_98 (00:00:0c:07:ac:98)
  ▶ Source: Apple_c6:12:53 (ac:bc:32:c6:12:53)
    Type: IPv4 (0x0800)
▼ Internet Protocol Version 4, Src: 10.12.58.26 (10.12.58.26), Dst: syd15s04-in-f3.1e100.net (216.58.196.131)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT)
    Total Length: 54
```

Question 4

Apart from GET, what are the ot
sentence description for each o

The GET method retrieves the wel
the client.

The HEAD method is similar to th
returned, and only the headers. T
whether the page has been updatec

The POST method allows clients t

```
my_new_host_name:~ Sia$ telnet www.google.com.au 80
Trying 216.58.196.131...
Connected to www.google.com.au.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.0 200 OK
Date: Mon, 15 Oct 2018 07:28:21 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2018-10-15-07; expires=Wed, 14-Nov-2018 07:28:21 GMT; path=/;
 domain=.google.com
Set-Cookie: NID=141=cd6fL5XgbT4HBPJ1HVpjhJBlX1Et2NpB7d4h8-6ClE1Rg0qa-zboXlik3qbv
a7Hkn1vy10nH6RIAGmfk_ovtOHqAsqHC9xmidB9YTms7VeyZHnqqPJmqULtRnp5qeddz; expires=Tu
e, 16-Apr-2019 07:28:21 GMT; path=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding

Connection closed by foreign host.
```

Question 7

What security implications does HTTP have, relating your answer to what you have captured over Wireshark.

HTTP is unencrypted. All data is transmitted as plain-text across the network medium, as captured in Wireshark. HTTPS is the encrypted version of HTTP, and should be used when transmitted sensitive information across the web.

Bonus Question

What advantages does the new HTTP/2 standard bring?

The biggest improvement HTTP/2 brings is the ability to request resources in parallel over one TCP connection. Previously in HTTP/1.1, this was done by opening multiple TCP connections. Fewer connection establishments means fewer 3-way handshakes and the use of fewer networking resources.

HTTP/1.x was text-based and was readable by humans, as we observed in the lab. HTTP/2 is a binary protocol and leverages this to provide better efficiency. In addition, compression is used for the HTTP headers in HTTP/2 to further save bandwidth.