

Lab 7 TCP

TCP (Transmission Control Protocol) is the main transport layer protocol used in the Internet. It is covered in Chapter 6.5 of the textbook. Review the text section before doing this lab.

Objective

- Know the packet format of TCP segment
- Understand the connection establishment and release processes of TCP
- Understand the sliding window protocol used for TCP data transmission (e.g., sequence/ack number)
- Understand TCP operations, such as how to calculate the checksum, TCP retransmission scheme, etc.

Requirements

You need to install following tools on your computer beforehand:

- **Wireshark:** This lab uses the Wireshark software tool to capture and examine a packet trace. Refer to previous labs for details.
- **Browser:** This lab uses a web browser to find or fetch pages as a workload. Any web browser will do.
- **wget/curl:** This lab uses *wget* (Linux and Windows) and *curl* (Mac) to fetch web resources. Refer to previous labs for details.

Exercise

Task 1: Capture and Explore Trace of TCP

Many applications use TCP as a transport, including web browsers. So, we will simply perform a web download to exercise a TCP connection. However, note that TCP is able to transfer data in both directions at the same time, but in the download, content is only sent from the remote server to the local computer (after the initial request).

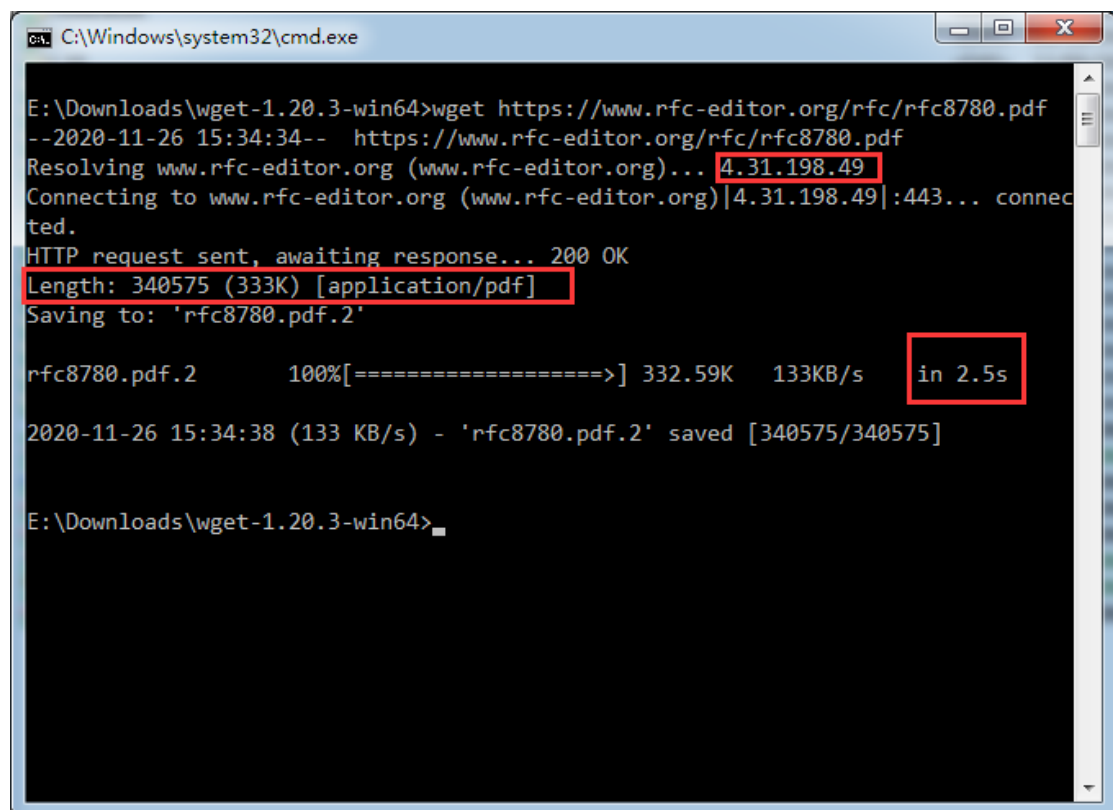
1. Find a URL of a single *moderately-sized web resource*, which you can download using [HTTP](#).

Tips:

- The web resources can be an *image*, *pdf/doc/docx* document, a single *html* file, etc.
 - [We prefer HTTP to HTTPS](#), as HTTPS encrypts all traffic transmitted. However, HTTPS is also OK if you can't find a website of HTTP.
2. Fetch the URL with *wget* or *curl* to check that you are able to retrieve the content of the resource over at least several of network time seconds. For example, you

can fetch an IETF RFC document (shown in the following figure):

wget https://www.rfc-editor.org/rfc/rfc9293.pdf



```
C:\Windows\system32\cmd.exe

E:\Downloads\wget-1.20.3-win64>wget https://www.rfc-editor.org/rfc/rfc8780.pdf
--2020-11-26 15:34:34-- https://www.rfc-editor.org/rfc/rfc8780.pdf
Resolving www.rfc-editor.org (www.rfc-editor.org)... 4.31.198.49
Connecting to www.rfc-editor.org (www.rfc-editor.org)|4.31.198.49|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 340575 (333K) [application/pdf]
Saving to: 'rfc8780.pdf.2'

rfc8780.pdf.2      100%[=====>] 332.59K   133KB/s   in 2.5s

2020-11-26 15:34:38 (133 KB/s) - 'rfc8780.pdf.2' saved [340575/340575]

E:\Downloads\wget-1.20.3-win64>
```

In the output of *wget*, you can determine the IP address of the web server, e.g., 4.31.198.49 in the above example.

3. Launch Wireshark and start a capture with a filter of "*ip.addr==4.31.198.49*". [IP address is the IP of the webserver.](#)
4. After the capture is started, repeat the *wget/curl* command above. This time, the packets will be recorded by Wireshark.
5. When the command is complete, stop Wireshark and inspect the trace you captured (Check each fields of TCP header).

Tips:

- We have provided you a Wireshark trace on course website for your reference.
- All packets except the initial HTTP GET and last packet of the HTTP response should be listed as TCP.
- Picking a long packet ensures that we are looking at a download packet from the server to your computer.
- Looking at the protocol layers, you should see an IP block before the TCP block. This is because the TCP segment is carried in an IP.

Answer the following questions:

1. [Explain how to calculate the checksum for TCP.](#)

Task 2: TCP Connection Setup/Teardown

1. Draw a time sequence diagram of the *three-way handshake* according to your trace, up to and including the first data packet (the HTTP GET request) sent by your computer when the connection is established. Put your computer on the left side and the remote server on the right side. Include the following features on your diagram:
 - The *Sequence* and *ACK number*, if present, on each segment. The ACK number is only carried if the segment has the *ACK flag* set.
 - The time in milliseconds, starting at zero, each segment was sent or received at your computer.
 - The round-trip time to the server estimated as the difference between the SYN and SYN-ACK segments.
2. Check connection Options in your Wireshark trace. [If any, describe what TCP Options are carried on the SYN packets for your trace.](#)

Tips:

- Common Options include: Maximum Segment Size (MSS) to tell the other side the largest segment that can be received. Timestamps to include information on segments for estimating the round-trip time. NOP (No-operation) and End of Option list that serve to format the Options but do not advertise capabilities.
 - Options can also be carried on regular segments after the connection is set up when they play a role in data transfer. This depends on the type of each Option. For example: the MSS option is not carried on each packet because it does not convey new information; timestamps may be included on each packet to keep a fresh estimate of the RTT; and options such as SACK (Selective Acknowledgments) are used only when data is received out of order.
 - You can check options on data packets in your trace.
3. FIN/RST Teardown:
Draw a picture of the teardown in your trace, starting from when the first FIN or RST is issued until the connection is complete. As before, show the *Sequence* and *ACK numbers* on each segment. If you have FINs, use the time difference to estimate the round-trip time.

Tips:

- The TCP connection is taken down after the download is complete. This is typically done with FIN (Finish) segments. Each side sends a FIN to the other and acknowledges the FIN they receive.
- Alternatively, the connection may be torn down abruptly when one end sends an RST (Reset). This packet does not need to be acknowledged by the other side.

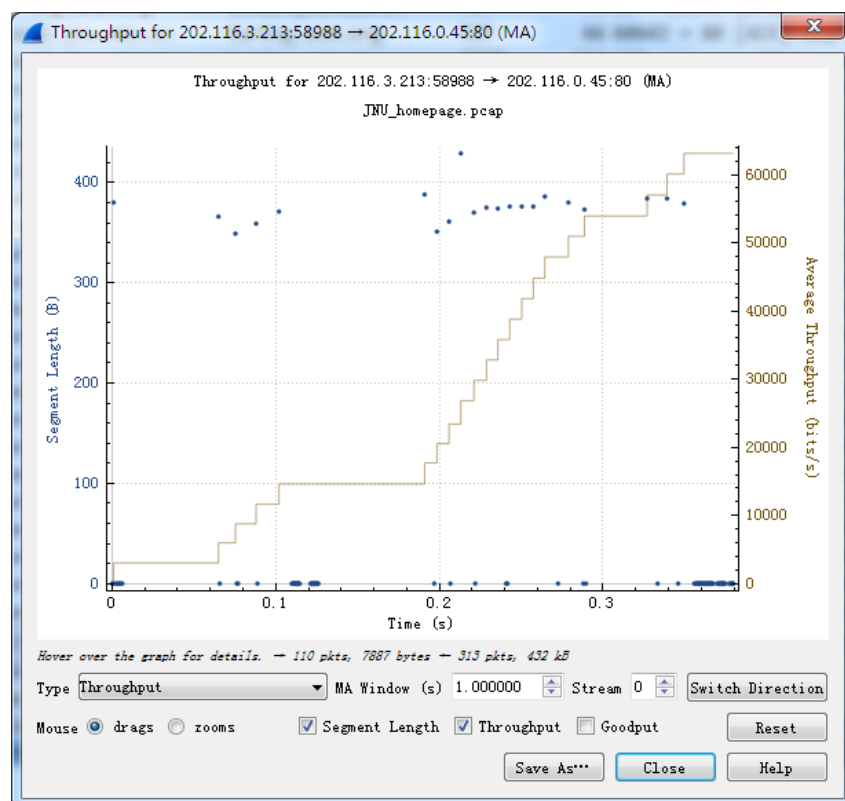
Answer the following questions:

1. [Why three-way handshaking is needed in the process of TCP connection establishment? What will happen if we do not use three-way handshaking?](#)

2. Explain the process of TCP connection release?
3. Is there is a “maximum segment size (MSS)” option in the first two message header when TCP establishes a connection in your trace. What’s its value? Considering the maximum length of Ethernet frame specified by IEEE802.3, explain how to obtain the value of MSS.

Task 3: TCP Data Transfer

1. Study the TCP data transfer process using “TCP Stream Graph”. In the Menu: “Statistics” → “TCP Stream Graph” → “Throughput”, see following figure:



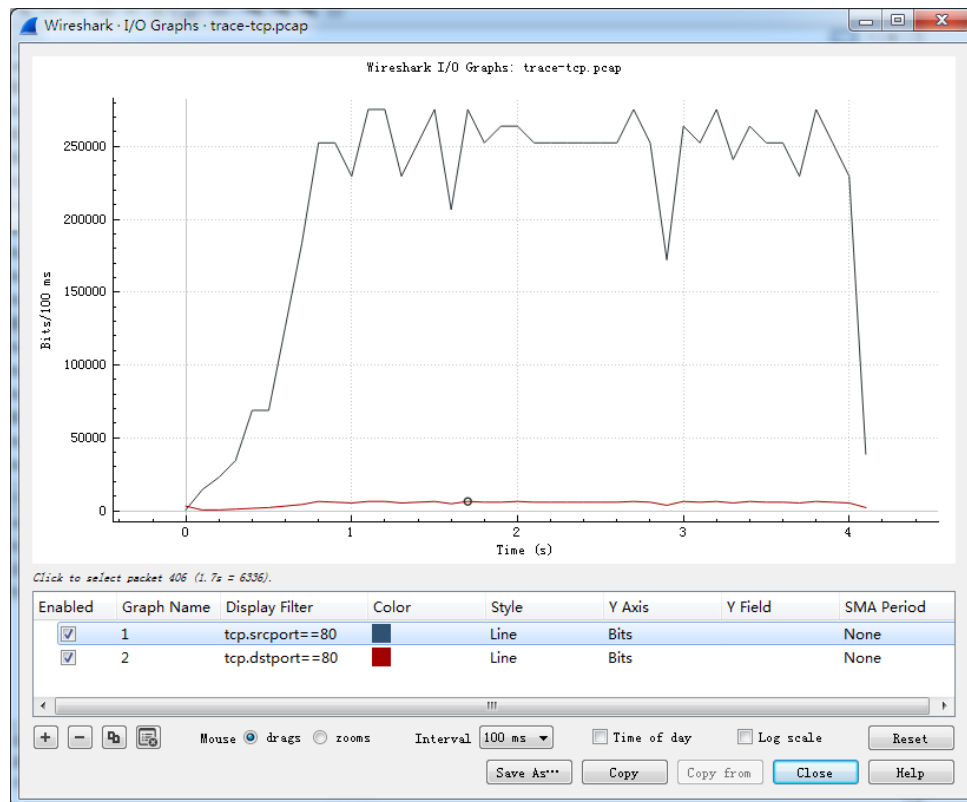
2. Study the TCP data transfer process using “IO Graph”. In the Menu: “Statistics” → “IO Graph”, see following figure:

Tips:

- On the x-axis, adjust the tick interval and pixels per tick. The tick interval should be small enough to see into the behavior over the trace, and not so small that there is no averaging. 0.1 seconds is a good choice for a trace of several seconds. The pixels per tick can be adjusted to make the graph wider or narrower to fill the window.
- On the y-axis, change the unit to be Bits/Tick. The default is Packet/Tick. By changing it, we can easily work out the bits/sec throughput by taking the y-axis value and scaling as appropriate, e.g., 10X for ticks of 0.1 seconds
- Add a filter expression to see only the download packets. So far, we are looking at all of the packets. Assuming the download is from the usual web server port of 80, you can filter for it with a filter of “tcp.srcport==80”.

Don't forget to press Enter, and you may need to click the "Graph" button to cause it to redisplay.

- To see the corresponding graph for the upload traffic, enter a second filter in the next box. Again assuming the usual web server port, the filter is "*tcp.dstport==80*". After you press Enter and click the Graph button, you should have two lines on the graph.



Answer the following questions:

- What is the rough data rate in the download direction in packets/second and bits/second once the TCP connection is running well?
- What percentage of this download rate is the content that client requests, e.g., the PDF file in above example? Show your calculation.

Tips: This download rate contains both overhead (e.g., TCP headers) and the real content that the client requests. To find out, look at a typical download packet (there should be many similar, large download packets). You can see how long it is, and how many bytes of TCP payload it contains.

- What is the rough data rate in the upload direction in packets/second and bits/second due to the ACK packets?

Task 4: Retrieving a static HTML webpage

In this task, you will capture the HTTP (based on TCP) traffic of the first website of the Internet, which is hosted on *info.cern.ch*. The URL of the first website in the world is: <http://info.cern.ch/hypertext/WWW/TheProject.html>

The detailed steps are as follow:

1. Determine the IP address of domain: *info.cern.ch*. (**Tips:** You can ping the domain.)

Suppose the IP address of the webserver is: 188.184.21.108.

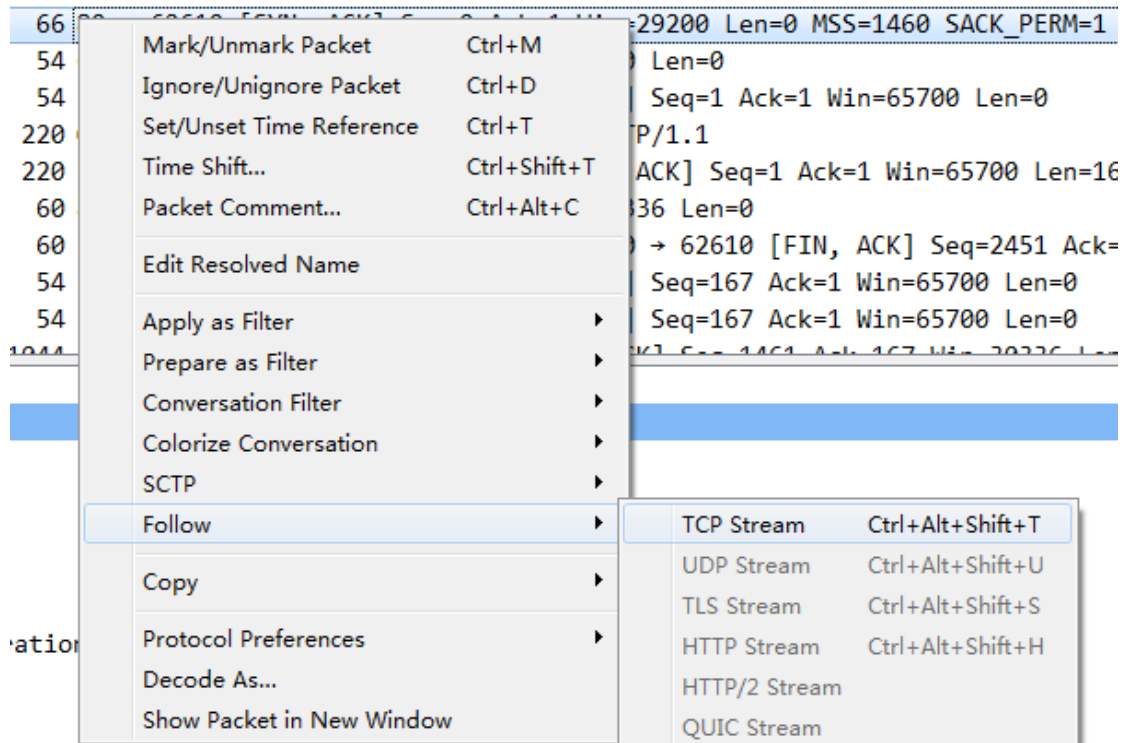
2. Launch Wireshark and start a capture with a filter of "*tcp.port == 80 && ip.addr==188.184.21.108*" (**The IP address needs to be changed accordingly**).
3. Fetch the URL with *wget* or *curl*:

```
wget http://info.cern.ch/hypertext/WWW/TheProject.html
```

4. Stop the Wireshark and inspect each TCP segment the trace you captured.
5. Similar to Task 2, you can inspect the process of the conversation, specially focusing on the SYN/ACK/FIN flags, *Sequence number* and *ACK number* for each TCP segment.
6. Try to reconstruct the HTML webpage based on the captured traffic in Wireshark.

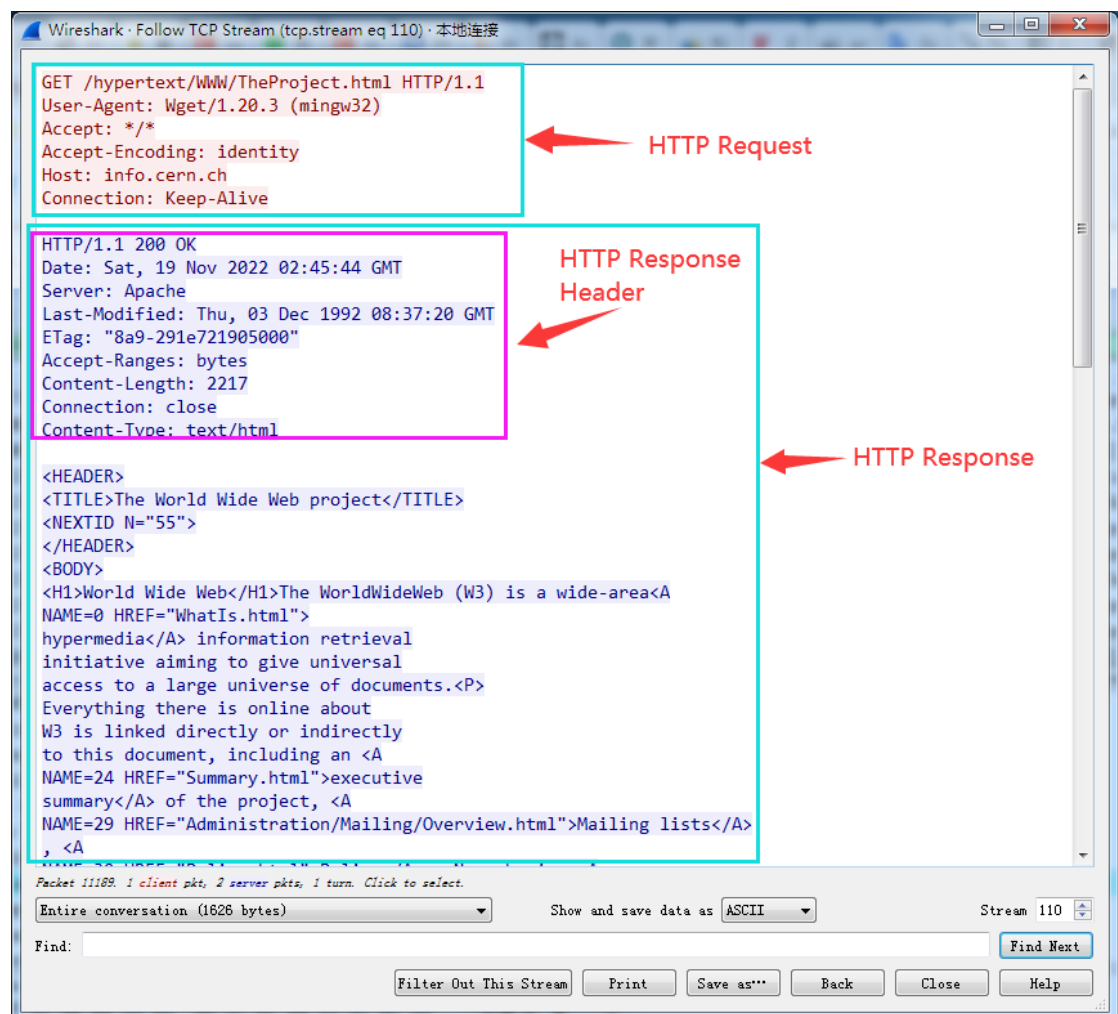
Tips:

- There are two ways to reconstruct the webpage. You can find TCP segments sent from the webserver, choose the TCP payload and copy as printable text. However, some unprintable characters need to be removed when you paste them to text file.
- Another way is to right click on any TCP segment of the conversation, choose "Follow" → "TCP Stream". This will list payloads for all TCP segments.



- In this experiment, since the webpage is every simple, **there is only one HTTP request and one HTTP response**. You can also choose display HTTP messages on one direction, e.g., messages from remote web server to local

computer.



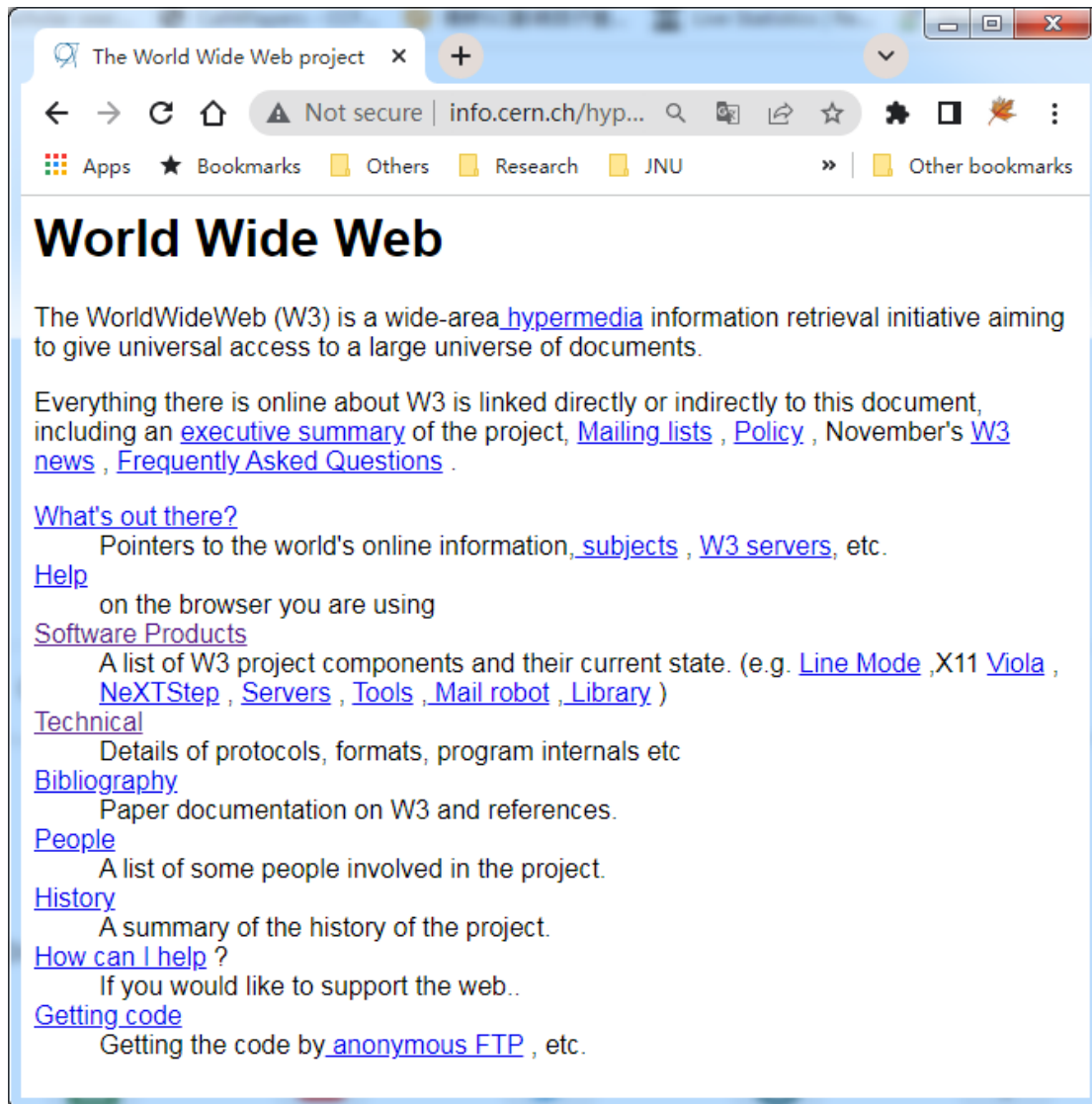
- You can create a plaintext file, copy **the payload of the HTTP response (excluding the HTTP response header)** to the file and change the extension of the file to be "html", e.g., . Then you can open the file in a web browser (see the following figure).

Answer the following questions:

- How many TCP segments are used to carry the HTTP request message and the HTTP response message, respectively?

Tips: There are only two HTTP messages: the client sends one HTTP request message to the web server, and the web server sends back one HTTP response message. All HTTP messages are carried in the payload of TCP segments, i.e., HTTP treat TCP connection as a reliable byte-stream channel.

- What's the total length of the HTTP response message?
Has IP fragmentation occurred during the transmission of the HTTP response message? Why or why not?



Task 5: Web traffic capture and analysis (NOT Required. OR you can find other login websites that use HTTP)

In this task, you will capture the HTTP (based on TCP) traffic of the login conversion for campus network in Jinan University. The detailed steps are as follow:

1. Logout your network account of JNU if you are already logged in.
2. Open any website (other than website in JNU, e.g., [www.baidu.com](#)) to trigger the *login webpage* for the campus network in a web browser.



- Launch Wireshark to capture data and configure filter (extract HTTP protocol). Fill in the user name and password, and click “Login” button.

Tips:

- Please use your **student ID** as the user name, and a **random string** as the password. **DO NOT use your real password in the lab report!!!**
- Stop the Wireshark, locate and analyze the HTTP request and response messages.
 - Check the content of messages sent from your computer to the web server.

Tips: You may be interested in the **HTTP POST message**, which is issued by your web browser when you click the “Login” button.

Question: Can you find the user name and password in the captured TCP packet payload? Are these sensitive information encrypted? If not, do you have any suggestions?

Tips:

- You need to first locate the **HTTP POST** message. For example, you can use filter of “http” or “ip.addr == 192.168.11.67” (**The IP address needs to be changed accordingly**), and find the key word “POST” in the information of packet.

597	2.836613	172.17.0.1	web.robinson.network	TCP	479	60350 → 80 [ACK] Seq=1 Ack=1 Win=65217 Len=425
988	4.809708	172.17.0.1	net.nic.jnu.edu.cn	HTTP	133	POST /eportal/InterFace.do?method=login HTTP/1.1 (applicat
1050	4.965689	net.nic.jnu.edu.cn	172.17.0.1	HTTP	478	HTTP/1.1 200 OK (text/html)

```

Transmission Control Protocol, Src Port: 60369, Dst Port: 80, Seq: 1, Ack: 1, Len: 1278
Hypertext Transfer Protocol
HTML Form URL Encoded: application/x-www-form-urlencoded
  Form item: "userId" = "202211111"
  Form item: "password" = "11111"
  Form item: "service" = ""
  Form item: "queryString" = "wlanuserip%3D38ede9139ec247e2b264be51e53c3305%26wlanacname%3D20cf58f"

```

0350	3b 20 4a 53 45 53 53 49	4f 4e 49 44 3d 33 45 44	; JSESSI ONID=3ED
0360	33 36 39 38 46 32 46 46	39 42 45 35 35 32 38 33	3698F2FF 98E55283
0370	33 41 45 37 34 44 31 45	45 31 39 38 31 0d 0a 0d	3AE74D1E E1981...
0380	0a 75 73 65 72 49 64 3d	32 30 32 32 31 31 31 31	-userId= 202211111
0390	31 26 70 61 73 73 77 6f	72 64 3d 31 31 31 31 31	1&password=11111
03a0	26 73 65 72 76 69 63 65	3d 26 71 75 65 72 79 53	&service=&queryString=wlanuserip
03b0	74 72 69 6e 67 3d 77 6c	61 6e 75 73 65 72 69 70	%253D38ede9139ec
03c0	25 32 35 33 44 33 38 65	64 65 39 31 33 39 65 63	247e2b26 4be51e53
03d0	32 34 37 65 32 62 32 36	34 62 65 35 31 65 35 33	

6. Try other websites, e.g., <https://mail.jnu.edu.cn/>.

Question: Check whether sensitive information are encrypted this time. What's the reason? Do you have any suggestions?

(Tips: Check the initials of the two URLs. What's the difference between *http* and *https*?)

Task 6: Explore on your own (Not required in the lab report)

We encourage you to explore TCP on your own once you have completed this lab. Some ideas:

- Explore the congestion control and the classic AIMD behavior of TCP. To do this, you will likely want to capture a trace while you are sending (not receiving) a moderate amount of data on a TCP connection. You can then use the "TCP Stream Graph" tools as well as other analysis to observe how the congestion window changes over time.
- Explore the reliability mechanisms of TCP more deeply. Capture a trace of a connection that includes segment loss. See what triggers the retransmissions and when. Also look at the round-trip time estimator.
- Look at the use of options including SACK to work through the details. You should see information about ranges of received bytes during times of segment loss.
- TCP is the transport layer underlying the web. You can see how your browser makes use of TCP by setting up concurrent connections.
- Write a socket program to implement a simple client/server communication application.

Questions:

- What is the disadvantage of transmitting real time voice data through TCP protocol? What is the problem of transmitting data file through UDP protocol? (Tips: consider the delay and packet order during transmission)?