Problem 1

Consider the following string of ASCII characters that were captured by *Wireshark* when the browser sent an HTTP GET message (i.e., this is the actual content of an HTTP GET message). The characters <CR><LF> are carriage-return and line-feed characters. Answer the following questions, indicating where in the HTTP GET message below you find the answer.

```
GET /classes/spring17/cs118/project-1.html HTTP/1.1<CR><LF>
Host: web.cs.ucla.edu<CR><LF>
Connection: keep-alive<CR><LF>
Upgrade-Insecure-Requests: 1<CR><LF>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/56.0.2924.87 Safari/537.36<CR><LF>
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8<CR><LF>
Referer: http://web.cs.ucla.edu/classes/spring17/cs118/homeworks.html<CR><LF>
Accept-Encoding: gzip, deflate, sdch<CR><LF>
Accept-Language: en-US,en;q=0.8,lv;q=0.6,ru;q=0.4<CR><LF>
If-None-Match: "5a17-54c4847c4f640-gzip"<CR><LF>
If-Modified-Since: Mon, 03 Apr 2017 19:36:49 GMT<CR><LF>
```

1. What is the **full** URL of the document requested by the browser?
2. What version of HTTP is the browser running?
3. What type of browser initiates this message? Why is the browser type needed in an HTTP request message?
4. Can you find the IP Address of the host on which the browser is running from the captured HTTP request?

1. web.cs.ucla.edu/classes/spring17/cs118/project-1.html

2. HTTP/1.1

3. Mozilla/5.0. Browser type is needed since the server will send different versions of the objects to different browsers so they can render things correctly.

4. No, the IP address is only specified when the TCP conection is established, not in request/response messages.

Problem 2

For each of the questions below, describe answer in terms of low-level packet sequences, drops, or network-level packet reordering.

1. A specific case where HTTP/1.1 wins in performance compared to HTTP/1.0
2. A specific case where HTTP with web caching wins in performance compared to HTTP without caching

1. When requesting n objects without parallel TCP connections. HTTP/1.1 is persistent meaning the TCP connection will remain opened for the rest of 2n messages. HTTP/1.0 is non persistent, so there will be a new TCP connection for each object, increasing the total message count to 4n (compared to HTTP/1.1's 2 + 2n).

2. Consider a network where a certain website is accessed by all the computers, all the time. With web caching (proxying), this request/response is cached in the proxy server so the next time someone from the network requests the same objects, there is no need to fetch it from the serving remote server. This will eliminate the communication between proxy server and remote server completely.

Problem 3

Suppose within your Web browser you click on a link to obtain a Web page. The IP address for the associated URL is cached in your local host, so a DNS look-up is not needed. Suppose that the Web page associated with the link is a small HTML file, consisting only of references to 100 very small objects on the same server. Let RTT_0 denote the RTT between the local host and the server containing the object. How much time elapses (in terms of RTT_0) from when you click on the link until your host receives all of the objects, if you are using:

1. HTTP/1.0 without parallel TCP connections?
2. HTTP/1.0 with parallel TCP connections?
3. HTTP/1.1 without parallel connections, but with pipelining?

Ignore any processing, transmission, or queuing delays in your calculation.

1. HTTP/1.0 without parallel TCP:
1 object: RTT + RTT = 2 RTT
101 objects: 2 RTT * 101 = 202 RTT

2. HTTP/1.0 with parallel TCP:
2 RTT per connection, all happening at the same time. Therefore,
For HTML object: 2 RTT
For the 100 objects: 2 RTT
Total: 4 RTT

3. HTTP/1.1 with pipelining only
Open connection: 2 * RTT
101 objects: roughly 100 * RTT
Total: roughly 102 RTT

Problem 4

BitTorrent is a communication protocol for peer-to-peer file sharing which is used to distribute data (or files) over the Internet.

1. Consider a new peer A that joins BitTorrent swarm without possessing any chunks. Since peer A has nothing to upload, peer A cannot become a top uploader for any of the other peers. How then will peer A get the first chunk?
2. Explain why BitTorrent is primarily useful for popular files but not for unpopular files.
3. Consider two DHTs (Distributed Hash Table) with a mesh overlay topology and a circular overlay topology, respectively. What are the advantages and disadvantages of each design?

1. P2P network protocol ensures that new nodes will have the change to obtain some chunks before getting "weeded out" for lack of contribution. In this case, peer A's neighbors will transfer chunks when initially joined. Node A will then exchange these chunks with its neighbors, eventually obtaining more chunks and becoming a top uploader.

2. A P2P network does not have a central server that hosts all the data, it is the nodes that are collectively sharing the data. Therefore, if the file is unpopular, no one in the network might have it. On the contrary, most of the nodes will have popular files, making it faster to spread it across the network.

3. The advantage of a mesh topology is that only one hop is required to route the message to a peer, and there are multiple delivery paths. The disadvantage is that the structure is complex and difficult to develop and maintain.

The advantage of a circular overlay is that the structure is fairly simple, and only has one main delivery path. The disadvantage is that it may take up to N-1 hopes in a network with N nodes to delivery a message.

Problem 5

The server tries to distribute a file of $F = 15Gbits$ to N clients (peers). The server has an upload rate of $u_s = 30Mbps$, and each peer has a download rate of $d_p = 2Mbps$ and upload rate of $u_p = 1Mbps$. How long does it take to distribute if there are 1,000 peers for both **client-server distribution** and **P2P distribution**.

tcs = max (NF / us, F / dp)
= max (500,000, 750)
= 500,000 s

tp2p = max (F / us, F / min(di), NF / (us + sum(ui))
= max (500, 750, 15000Gb / (30 + 1000) Mbps)
= max (500, 750, 14,563.11)
= 14,563.11 s