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/winter19/cs118/hw2.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/winter19/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 Jan 2019 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. The document request was **http://web.cs.ucla.edu/classes/winter19/cs118/hw2.html**. The Host: field indicates the servers name and **/classes/winter19/cs118/hw2.html** indicates the file name (path).
2. HTTP version 1.1
3. Chrome, Safari, or Mozilla.
4. No, this information is not contained in an HTTP message anywhere. So there is no way to tell this from looking at the exchange of HTTP messages alone. One would need information from the IP datagrams (that carried the TCP segment that carried the HTTP GET request) to answer this question.

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. An HTML containing a lot of referenced objects (N). HTTP/1.1 uses a persistent TCP connection. It requires 1 (TCP connection) + 1 (GET index.html) + 1 (or more) (GET all referenced objects) RTTs. HTTP/1.0 uses non-persistent TCP connection, requires 2xN RTT time.
2. If clients' requests contain objects that are already in the cache, it reduces traffic on the content server's link as well as response time for clients.

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? The number of parallel connections is 50.
3. HTTP/1.1 without parallel connections, but with pipelining?

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

1. 202 RTT_0
2. 2 RTT_0 + 200 RTT_0 / 50 = 6 RTT_0
3. 3 RTT_0

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. In BitTorrent, a peer picks a random peer and optimistically unchokes the peer for a short period of time. Therefore, peer A will eventually be optimistically unchoked by one of its neighbors, during which time peer A will receive chunks from that neighbor.
2. The efficiency of BitTorrent is predicated on popularity. The more people downloading, the larger the distribution network gets. But for the unpopular files, there are only a handful of clients sharing part of chunks.
3. Mesh DHT: The advantage is in order to a route a message to the peer (with ID) that is closest to the key, only one hop is required; the disadvantage is that each peer must track all other peers in the DHT.
   Circular DHT: the advantage is that each peer needs to track only a few other peers; the disadvantage is that O(N) hops are needed to route a message to the peer that is closest to the key.

Note: this problem is optional.

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**.

Client-server distribution time is $D_{cs} = max\{NF/u_s, F/d_p\}$ where N is the number of peers and F is the file size.
P2P distribution time is $D_{p2p} = max\{F/u_s, F/d_p, NF/(u_s + \sum_{i=1}^{N} u_i\}$

$F = 15Gbits = 15 \times 1000Mbits$
$N = 1000$
$u_s = 30Mbps$
$d_p = 2Mbps$
$u_p = 1Mbps$

$D_{cs} = max(\frac{1000 \times 15 \times 1000}{30}, \frac{15 \times 1000}{2}) = 500ksec$

$D_{p2p} = max(\frac{15 \times 1000}{30}, \frac{15 \times 1000}{2}, \frac{1000 \times 15 \times 1000}{30 + (1000 \times 1)}) = 14563.1sec$