tcpdump command I used:

```
sudo tcpdump -i ens33 -n port 1080 -w http_1080.pcap
sudo tcpdump -i ens33 -n port 1081 -w tcp_1081.pcap
sudo tcpdump -i ens33 -n port 1082 -w tcp_1082.pcap
```

1. **Reassemble each unique HTTP Request/Response for http_1080.pcap (the other two are encrypted, so you will not be able to reassemble easily). The output of this part should be the Packet type (request or response) and the unique <source, dest, seq, ack> TCP tuple for all the TCP segments that contain data for that request.**

```
HTTP REQUEST [('130.245.42.141', '34.193.77.105', '1692976978', '942771623')]
Response:
['34.193.77.105', '130.245.42.141', '3385535882', '3927246399']
['34.193.77.105', '130.245.42.141', '942771623', '1692977371']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '2060456136', '3834615676')]
Response:
['34.193.77.105', '130.245.42.141', '4117671396', '790394647']
['34.193.77.105', '130.245.42.141', '3834615676', '2060456529']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '3927246007', '3385535882')]
Response:
['34.193.77.105', '130.245.42.141', '3385535882', '3927246399']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '3825584121', '894040155')]
Response:
['34.193.77.105', '130.245.42.141', '4225278226', '1223129213']
['34.193.77.105', '130.245.42.141', '894040155', '3825584519']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '1519693845', '4025958516')]
Response:
['34.193.77.105', '130.245.42.141', '3385535882', '3927246399']
['34.193.77.105', '130.245.42.141', '4025958516', '1519694238']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '2653592391', '2599999143')]
Response:
['34.193.77.105', '130.245.42.141', '4225278226', '1223129213']
['34.193.77.105', '130.245.42.141', '2599999143', '2653592789']
['34.193.77.105', '130.245.42.141', '3096622929', '1829187849']
['34.193.77.105', '130.245.42.141', '2965334766', '450495367']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '2978422741', '2873917737')]
Response:
['34.193.77.105', '130.245.42.141', '3385535882', '3927246399']
['34.193.77.105', '130.245.42.141', '2873917737', '2978423069']
```

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '1223128812', '4225278226')]
Response:
['34.193.77.105', '130.245.42.141', '3385535882', '3927246399']
['34.193.77.105', '130.245.42.141', '4225278226', '1223129213']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '2095985015', '4250809387')]
Response:
['34.193.77.105', '130.245.42.141', '3385535882', '3927246399']
['34.193.77.105', '130.245.42.141', '4250809387', '2095985412']
['34.193.77.105', '130.245.42.141', '4225278226', '1223129213']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '31105941', '3657112380')]
Response:
['34.193.77.105', '130.245.42.141', '4225278226', '1223129213']
['34.193.77.105', '130.245.42.141', '3657112380', '31106337']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '1829187453', '3096622929')]
Response:
['34.193.77.105', '130.245.42.141', '4225278226', '1223129213']
['34.193.77.105', '130.245.42.141', '3096622929', '1829187849']
['34.193.77.105', '130.245.42.141', '2965334766', '450495367']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '3358249171', '317762981')]
Response:
['34.193.77.105', '130.245.42.141', '317762981', '3358249605']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '3988477648', '1603092679')]
Response:
['34.193.77.105', '130.245.42.141', '4117671396', '790394647']
['34.193.77.105', '130.245.42.141', '1603092679', '3988478041']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '450494970', '2965334766')]
Response:
['34.193.77.105', '130.245.42.141', '4225278226', '1223129213']
['34.193.77.105', '130.245.42.141', '2965334766', '450495367']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '84522070', '3674383617')]
Response:
['34.193.77.105', '130.245.42.141', '4117671396', '790394647']
['34.193.77.105', '130.245.42.141', '3674383617', '84522466']
['34.193.77.105', '130.245.42.141', '1603092679', '3988478041']

HTTP REQUEST [('130.245.42.141', '34.193.77.105', '790394249', '4117671396')]
Response:
['34.193.77.105', '130.245.42.141', '3096622929', '1829187849']
['34.193.77.105', '130.245.42.141', '4117671396', '790394647']
['34.193.77.105', '130.245.42.141', '1603092679', '3988478041']

```
HTTP REQUEST [('130.245.42.141', '34.193.77.105', '672423938', '1453819050')]
Response:
['34.193.77.105', '130.245.42.141', '3385535882', '3927246399']
['34.193.77.105', '130.245.42.141', '1453819050', '672424341']
```

2. **Identify which HTTP protocol is being used for each PCAP file. Note that two of the sites are encrypted so you must use your knowledge of HTTP and TCP to programmatically solve this question. Include the logic behind your code in the write-up.**

```
------------------------------PCAP File: http_1080.pcap--------------------------------
Tcp connection count = 17
```

The tcpdump taken over 1080 port is HTTP 1.0.
Reasons are, we can see from the pcap file that there are 17 tcp connections made for loading the webpage. If we see the page source of the website, it has 17 objects which needs to be requested through http. So, the browser makes 17 connections to fetch each object.

```
------------------------------PCAP File: tcp_1081.pcap--------------------------------
Tcp connection count = 6
```

The tcpdump taken over 1081 port is HTTP 1.1.
Reasons are, we can see from the pcap file that there are 6 tcp connections made by the browser when we fetch the website, and those connections are reused. This is how HTTP 1.1 makes persistent connection.

```
------------------------------PCAP File: tcp_1082.pcap--------------------------------
Tcp connection count = 2
```

The tcpdump taken over 1082 port is HTTP 2.0.
Reasons are, as we can see from the pcap file that there are 2 tcp connection established by the browser when we fetch the website, and that persistent connection is reused.

3. **Finally, after you've labeled the PCAPs with their appropriate versions of HTTP, answer the following: Which version of the protocol did the site load the fastest under? The Slowest? Which sent the most number of packets and raw bytes? Which protocol sent the least? Report your results and write a brief explanation for your observations.**

```
------------------------------PCAP File: http_1080.pcap--------------------------------HTTP 1.0
Tcp connection count = 17
Time Taken = 0.2483530044555664
Packet Count = 1741
Raw data size = 2260322
------------------------------PCAP File: tcp_1081.pcap--------------------------------HTTP 1.1
Tcp connection count = 6
Time Taken = 0.19395709037780762
Packet Count = 1501
Raw data size = 2266763
------------------------------PCAP File: tcp_1082.pcap--------------------------------HTTP 2.0
Tcp connection count = 2
Time Taken = 0.1563270092010498
Packet Count = 1507
Raw data size = 2277807
```

Which version of the protocol did the site load the fastest under?
HTTP 2.0
The Slowest?
HTTP 1.0
Which sent the most number of packets and raw bytes?
HTTP 1.0 sent the most number of packets and HTTP 2.0 sent the most raw bytes.
Which protocol sent the least?
HTTP 2.0 sent the least number of packets and HTTP 1.0 sent the least raw bytes.

Observation:
As you can see from the data, http 2.0 loaded the quickest and http 1.0 loaded the slowest. Since http 1.0 cannot request information until previous information was delivered and requires tcp connection to be closed and reopened, it makes sense that this was the slowest. It is quickest for http 2.0 since it only requires one connection, and it can pipeline with this. Http 1.0 has multiple tcp connections in parallel and therefore experiences a delay (compared to 2.0) in closing these. Http 1.0 sent the greatest number of packets, while http 2.0 sent the least. The number of packets sent are relatively the same, however, the difference is due to compression. Http 2.0 compresses its payload and header, thereby making it send

the least number of packets since each packet can contain more information. Http 1.1 is similar in that it compresses its payload, but it is not as good as http 2.0. Http 1.0 on the other hand, does not compress anything. Once again, the number of bytes sent are relatively the same. However, in this case http 1.0 has the least number of bytes while http 2.0 has the most. My best intuition for why this would be due to new advances made in later models. This can be security related, or possibly compression related. To my knowledge, there are several ways to compress material, and to correctly decompress the information, it is necessary to know how it was compressed, therefore there may be some additional flags in place.