Question 4 ¶

a) What is the propagation delay?

Propogation delay is determined by the propogation speed and the distance betweent the points. The propogation speed is
2*10^8 m/s and the distance is 5000m

In [1]:
```
speed = 2*10**8
distance = 5000
```

We can now calculate the propogation delay

In [2]:
```
propogation_delay = distance/speed
print('The propogation delay will be', propogation_delay*10**6,'us')
```

```
The propogation delay will be 25.0 us
```

b) What is the maximum number of bits that have left A before the first bit arrives at B? This number of bits is called the bandwidth delay product

The number of bits that have left A by the time the first bit arrives at B will be equal to the transmission rate of
the connection times the amount of time that it will take the first bit to reach B (the propogation delay). We can calculate the maximum number of bits as follows

In [3]:
```
transmission_rate = 24*2**20          # bits per second
bandwidth_delay_product = int(transmission_rate*propogation_delay)       # cast it to an int to round down
print('The maximum number of bits that have left A before the first bit arrives at B is', bandwidth_delay_pro
duct,'bits')
```

```
The maximum number of bits that have left A before the first bit arrives at B is 629 bits
```

c) We can think of each bit occupying a piece of the link. If the bandwidth delay product tells us that there are 100 bits in the transmission medium then we can think of each bit occupying 1/100 of the length of the transmission media. What is the width of each bit (the length of the transmission medium it fills)?

We need to know how many bits are in the medium at any given time (the bandwidth_daley_product) and the length of the
medium. Given that the medium is 5000m long and we have 629 bits in the medium at any given time, we can calculate the
width of each bit as follows

In [4]:
```
bit_width = distance/bandwidth_delay_product
print('Each bit is theoretically',bit_width,'m long')
```

```
Each bit is theoretically 7.94912559618442 m long
```

d) Consider sending the file through the packet switched network without message segmentation (as a single packet). Assume that each of the intermediate hosts are store and forward nodes. How long does it take to send the file from A to B?

To begin with, we know that the file is 3GB or 3*2^30 bits, the propogation delay on each link is 0.004s, the processing delay at each link is 0.003s, and there is no queueing delay. We also know that there at 9 intermediate links.

In [5]:
```
file_size = 3*2**30
propogation_delay = 0.004
processing_delay = 0.003
number_links = 9
```

We also need to find out the transmission delay. For the case when we do not segment the file, we will have the pa cket
size be equal to the file size. We also have a transmission rate of 90Mbps. Therefore, we are now able to calculat e the
transmission delay as follows

In [6]:
```
transmission_rate = 90*2**20        # bits per second
transmission_delay = file_size/transmission_rate
```

Finally, to get the overall delay, we realize that at each of the intermediate links as well as at the original ho st A,
we encounter a propogation delay and transmission delay. We also encounter a processing delay at each of the intermediate links as well as the end host, B. We will have a count of 10 for each delay. We can now calculate the total delay as follows

In [7]:
```
total_delay = (number_links + 1)*(processing_delay+propogation_delay+transmission_delay)
print('The total amount of time that it will take to send the file from A to B will be',total_delay,'s, or',
int(total_delay//60), 'm', total_delay%60, 's')
```

```
The total amount of time that it will take to send the file from A to B will be 341.4033333333333 s, or 5 m 4
1.40333333333331 s
```

e) Assume that the file is segmented into packets containing 12000 bits each. Each packet has a header of 200 bits. If a packet is partially full of data the remainder of the data field is filled with zeros before the resulting full size packet is transmitted. How long does it take for the file to be transmitted through the network (assume no queuing delays)

We will have the same file size, propogation delay, processing delay, and number of intermediate links as with par
t d).
However, we will have a new transmission delay, as we are sending smaller packets, this time of size 12000 with a
header that consists of 200 of those bits

In [8]:
```
packet_size = 12000
transmission_delay = packet_size/transmission_rate
```

We can now calculate the amount of delay experienced by each packet, using the same reasoning as in part d)

In [9]:
```
packet_delay = (number_links + 1)*(processing_delay+propogation_delay+transmission_delay)
```

However, now we must also take into account, how many packets we have to send into the system

In [10]:
```
data_per_packet = 12000-200
number_of_packets = file_size/data_per_packet
if number_of_packets%1 != 0:                # rounds up if not a whole number
    number_of_packets = number_of_packets//1 + 1
```

Finally, we take into account how long it will take to send the first n-1 packets into the system and then add on the
time it will take for the nth packet to reach host B. The rate at which we send packets into the system is limited by
the transmission rate and the size of the packets. Once we get this rate, we then have to multiply it by the number of
packets that we are sending into the system

One assumption made is that the processing delay does not effect the rate at which we can send the file into the system. If we take into account that we cannot start recieving the bits of the next packet until the current packet has finished being processed then we would have to add a term of (number_of_packets-1)*processing_delay onto the n_minus_one_time

In [11]:
```
packet_rate = packet_size/transmission_rate
n_minus_one_time = packet_rate * (number_of_packets-1)
total_time = n_minus_one_time + packet_delay
print('Using packets of 12000 bits, it will take', total_time,'s to transmit the file from A to B')
```

```
Using packets of 12000 bits, it will take 34.7831093343099 s to transmit the file from A to B
```

Question 5

a) What is the difference between a basic HTTP GET request and a conditional HTTP get request? When is each type of request used?

A condtional GET is the same as a regular GET, except it includes an if-modified-since field that contains the val
ue of
the last-modified field from the caches local copy of the webpage. The conditional GET is used by local caches to
ensure that their version of webpages are all up to date. The GET request is used when there is not a cached copy
 of
the webpage, when there is no local caches or for the first time requesting a webpage

b) What is different about the responses to a basic HTTP GET command and to a conditional HTTP GET request? What information does each type of response return? Would you expect a different response to the conditional get if the web page had been modified between the two requests?

The response to a conditional GET request will be identical to a regular GET request if the cached webpage is not
 up to
date. However, if the cached webpage is up to date then the server will send a 304 not-modified response and the d
ata
field will be empty, as to save bandwidth and the precieved delay.

c) What browser is making the request? What version of HTTP is that browser running? What version of HTTP is the queried web server running? Give reasons for your answers based on selected packets from the supplied packet captures

The browser making the request is Mozilla/5.0. The browser is running HTTP/1.1. The server is also running version 1.1
We can check which browser and the browsers HTTP version by checking one of the GET requests and expeanding the HTTP
section of the request. Under there, we can see the HTTP version in the request line, and we can see the browers in the
user-agent header line. We can find the servers HTTP version by checking the status line for the servers response message

d) What was the IP address of the computer running the browser? What was the IP address of the web server it queried? Use evidence from the file HTTP2020summary.pcapng.

The source IP address of the computer running the browser is 192.168.1.44 and the server IP address is 69.90.66.160.
We can tell this from any of the HTTP messages, by either looking under the ITP header or by simply looking at the source and destination columns. If the request was a GET request then the source will be the host running the browser,
if it is a response message, then the source will be the web server

e) For one of the HTTP GET responses shown in file conversationonly.pdf how many packets were used to carry the HTTP GET response from the server to the client? How did you determine your answer?

Looking at time 1.413535 in the conversation.pcapng file, we can see an HTTP GET response from the server. This response was sent using 6 packets, 5 of size 1460 bytes and one of 1402 bytes. We can see this in the header just under the TCP header

f) Were persistent or non-persistent connections used to download the webpage information from the server? Explain why you think so?

It would appear, looking at converstation.pcapng, that the webpages are downloaded using non-persistent TCP. This is because of the fact that it would appear that there are no points in time where the server has sent more than one response message to the clients HTTP GET messages, without the client having to send a SYN request through TCP. There are times where the client sends the GET request before the SYN request, however it would appear that the server is not sending the HTTP response message before it establishes a connection, and sends a SYNACK response before it send the HTTP response

g) Can you see any evidence that pipelining was used to download the webpage information from the server? Explain why you think pipelining was used, or why you think pipelining was not used

We can see that pipelining was used because we can see evidence of multiple TCP messages being sent by both the client and the server back to back, without waiting for a response for a particular message. We can see this all the way at time 1.413083 in conversation.pcapng, when the server sends 5 TCP segments in a row, without waiting for a response from the client, confirming that it was recieved properly. Another giveaway, being that we know that the messages were sent using TCP, and TCP was created using pipelining, as well as other things such as cumulative acknowledgements