1. Calculate the total time required to transfer a 1000-KB file in the following cases, assuming an RTT of 100 ms, a packet size of 1 KB data, and an initial 2×RTT of "handshaking" before data is sent:
    - The bandwidth is 1.5 Mbps, and data packets can be sent continuously.

$$Initial\ Handshaking = 2\ \times RTT = 2 \times 100ms = 200ms$$

$$Time\ to\ transmit = \frac{1000KB \times 1024B\ \times 8bits}{1.5Mbps} = 5461.33ms$$

$$Total\ Time = Initial\ Handshaking + Time\ to\ transmit + Propagation$$

$$Total\ Time = 5461.33ms + 200ms\ + 100/2ms = 5711.33ms$$

    - The bandwidth is 1.5 Mbps, but after we finish sending each data packet, we must wait one RTT before sending the next.

$$\#\ of\ packets = \frac{file\ size}{packet\ size}$$

$$\#\ of\ packets = \frac{1000KB}{1KB} = 1000\ packets$$

      We need to wait 1 RTT after the first packet, for each packet, so we need to add 999RTTs to the total time.

$$add\ 1\ RTT\ to\ each\ packet = Total\ time + 999RTT$$

$$add\ 1\ RTT\ to\ each\ packet = 5711.33ms + 999 \times 100ms = 105611.33ms$$

    - The bandwidth is "infinite," meaning that we take transmit time to be zero, and up to 20 packets can be sent per RTT.
      Infinite bandwidth implies that the total time to transmit will approach zero. With a limit of 20 packets then:

$$Needed\ RTTs = \frac{\#\ of\ packets}{packets\ per\ RTT} = \frac{1000}{20} = 50$$

      Need to adjust the RTT for the last packet as it only needs half an RTT, then:

$$Needed\ RTTs = 50 - 0.5 = 49.5$$

$$Total\ Time = 2RTT + 49.5RTT$$

$$Total\ Time = 200ms + 4950ms = 5150ms$$

    - The bandwidth is infinite, and during the first RTT we can send one packet, during the second RTT we can send two packets, during the third we can send four, and so on.

      Corresponds to $1 + 2 + 4 + \ldots + 2^n = 2^{n+1} - 1$

      Looking for: $2^{n+1} - 1 \geq 1000$, answer would be n = 9.

      After adding RTT/0.5 because of propagation:

$$Total\ Time = 2RTT + 9.5RTT = 1150ms$$

2. One property of addresses is that they are unique; if two nodes had the same address it would be impossible to distinguish between them. What other properties might be useful for network addresses to have? Can you think of any situations in which network addresses might not be unique?
   Addresses could follow a priority that could indicate which node needs to be reached out to when there are many sharing an address. We could use addresses of varying length so that the user may adjust them based on their respective system requirements. When a phone rings in a home, typically, all phones connected to the network ring at the same time. This could be because they all share the same address.

3. For each of the following operations on a remote file server, discuss whether they are more likely to be delay sensitive or bandwidth sensitive:
   1. Open a file
      More likely delay sensitive as, since contents aren't being read (probably), it's simply a matter of executing commands. Since there isn't much data transfer involved it's very unlikely bandwidth becomes an issue
   2. Read the contents of a file
      More likely bandwidth sensitive as the files being read could have a lot of data and that data will have to be sent back from the remote server.
   3. List the contents of a directory
      More likely delay sensitive as it's an execution of a command and it's very unlikely that the directory is large enough to create large amounts of data that rely on bandwidth more heavily.
   4. Display the attributes of a file
      Same case as with directories, files don't usually have that many relevant attributes to occupy too much data.
4. Suppose that a certain communications protocol involves a per-packet overhead of 100 bytes for headers. We send 1 million bytes of data using this protocol; however, when one data byte is corrupted, the entire packet containing it is lost. Give the total number of overhead + loss bytes for packet data sizes of 1000, 5000, 10000, and 20000 bytes. Which of these sizes is optimal?

$$\#\ of\ packets = \frac{file\ size}{packet\ size}$$

$$\#\ of\ packets = \frac{1000000B}{data\ size}$$

$$Total\ overhead = 100B\ \times \#\ of\ packets$$

$$Lost\ data = data\ size$$

$$Total\ size = \ Total\ overhead + lost\ data$$

- 1000B

$$Total\ size = \ 100B\ \times \frac{1000000B}{1000B} + 1000B = 101000B$$

- 5000B

$$Total\ size = \ 100B\ \times \frac{1000000B}{5000B} + 5000B = 25000B$$

- 10000B

$$Total\ size = 100B \times \frac{1000000B}{10000B} + 10000B = 20000B$$

- 20000B

$$Total\ size = 100B \times \frac{1000000B}{20000B} + 20000B = 25000B$$

The optimal size is 10000B

5. Suppose we want to transmit the message 11001001 and protect it from errors using the CRC polynomial $x^3 + 1$
   $x^3 + 0x^2 + 0x^1 + 1$ -> 1001
   1. Use polynomial long division to determine the message that should be transmitted.
      Message: 11001001 -> 11001001000

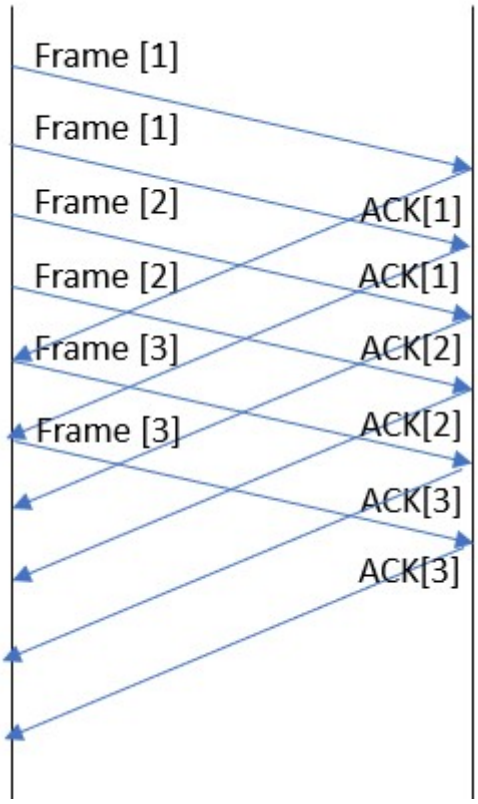
      ```
      11001001000    |1001
      1001
       1011
       1001
          1000
          1001
               1100
               1001
                1010
                1001
                 011
      ```

      The remainder is 011 and appended to the message: 11001001011

   2. Suppose the leftmost bit gets inverted in transit. What is the result of the receiver's CRC calculation?
      Message: 11001001011 -> 01001001011

      ```
      01001001011    |1001
       1001
              1011
              1001
               010
      ```

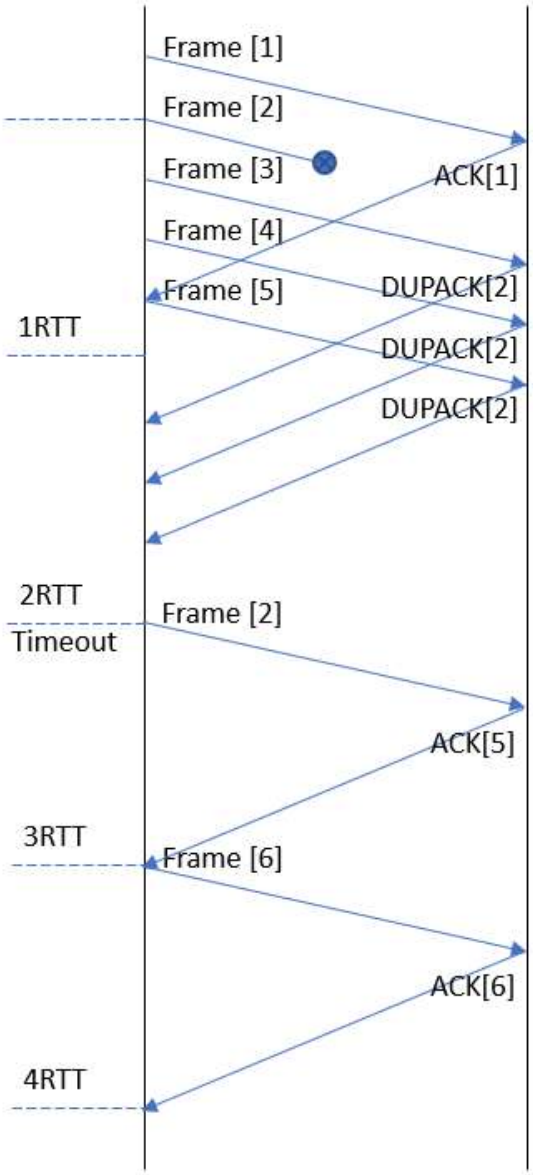
      Reminder is 10 and the receiver detects an error.
6. In stop-and-wait transmission, suppose that both sender and receiver retransmit their last frame immediately on receipt of a duplicate ACK or data frame; such a strategy is superficially reasonable because receipt of such a duplicate is most likely tomean the other side has experienced a timeout.
   1. 1. Draw a timeline showing what will happen if the first data frame is somehow duplicated, but no frame is lost. How long will the duplications continue? This situation is known as the Sorcerer's Apprentice bug.



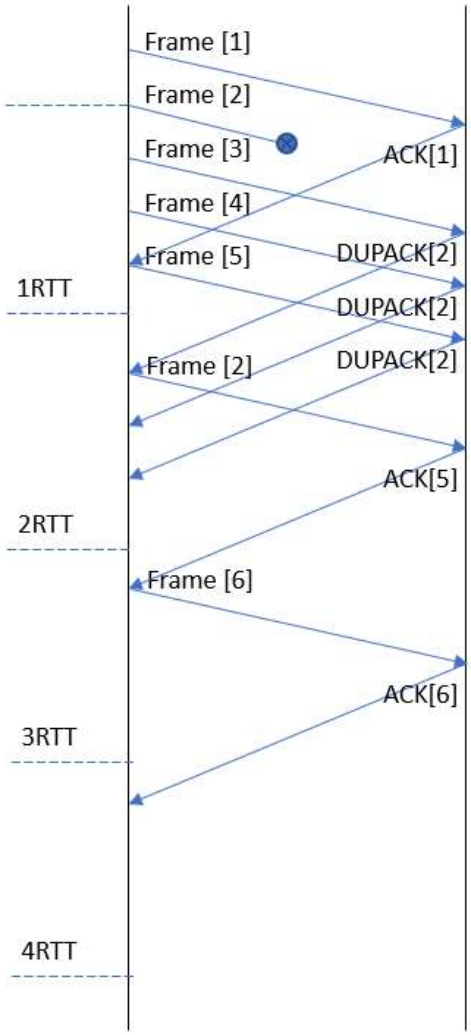
Duplications would continue until the communication ceases.

2. Suppose that, like data, ACKs are retransmitted if there is no response within the timeout period. Suppose also that both sides use the same timeout interval. Identify a reasonably likely scenario for triggering the Sorcerer's Apprentice bug
   If an ACK is lost and they the sender and receiver have the same timeout times then the Frame and ACK will be sent at the same time and the bug will take place

7. Draw a timeline diagram for the sliding window algorithm with SWS = RWS = 4 frames for the following two situations. Assume the receiver sends a duplicate acknowledgement if it does not receive the expected frame. For example, it sends DUPACK[2] when it expects to see FRAME[2] but receives FRAME[3] instead. Also, the receiver sends a cumulative acknowledgment after it receives all the outstanding frames. For example, it sends ACK[5] when it receives the lost frame FRAME[2] after it already received FRAME[3], FRAME[4], and FRAME[5]. Use a timeout interval of about 2 × RTT.
   1. Frame 2 is lost. Retransmission takes place upon timeout (as usual).



2. Frame 2 is lost. Retransmission takes place either upon receipt of the first DUPACK or upon timeout. Does this scheme reduce the transaction time? Note that some end-to-end protocols (e.g., variants of TCP) use a similar scheme for fast retransmission.


Frame [1]
Frame [2]
Frame [3]
ACK[1]
Frame [4]
Frame [5]
DUPACK[2]
1RTT
DUPACK[2]
Frame [2]
DUPACK[2]
ACK[5]
2RTT
Frame [6]
ACK[6]
3RTT
4RTT