CMP-4005 -- Homework 1
Sebastián Romero (00216765)

Answer the following questions.

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:**

    1. **The bandwidth is 1.5 Mbps, and data packets can be sent continuously.**
    200ms (2 handshakes) + 1Mb / 1.5Mbps = 200ms +

    $1000KB * \frac{8b}{1B} = 8x10^6b$

    $8x10^6b * \frac{1s}{1.5x10^6b} = 5.33\ s$

    Total time = 0.2 (handshakes) + 5.33 + 0.1/2 (propagation delay)= 5.58 s

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

    #packets = 1000
    Total time = + 0.1/2 (propagation delay) 999*0.1 (RTT wait per pkt) + 5.53 = 105.47 s

    3. **The bandwidth is "infinite," meaning that we take transmit time to be zero, and up to 20 packets can be sent per RTT.**

    1000/20 = 50 RTT

    Total time = 0.2 (handshakes) + 0.1/2 (propagation delay) + $50 * 0.1$= 5.25 s

    4. **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.**

    $$1000 = 2^{n+1} - 1$$

    $$n = 8.96 \approx 9$$

    Total time = 0.2 (handshakes) + 0.1/2 (propagation delay) + $9 * 0.1$= 1.15 s

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?**

   In addition to being unique, network addresses should be hierarchical, routable, and efficient to use.

   Hierarchical: Hierarchical addressing helps in organizing the addresses into manageable groups, which makes it easier to manage and scale the network.

   Routable: A routable address is one that can be used to identify a specific network or host within a network and can be used for routing data from one network to another.

   Efficient: Network addresses should be efficient in terms of the number of bits used to represent them. An address that requires too many bits will result in wastage of address space.

   Situations where network addresses are not unique. For example, in some cases, network address translation (NAT) is used to translate private addresses into public addresses, which can result in multiple nodes having the same public address. In such cases, it is the combination of the public address and port number that uniquely identifies a node. Additionally, some protocols, such as multicast, use non-unique addresses to send data to multiple recipients simultaneously.

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:**

   The file size is what impacts the most, larger files will be sensitive to bandwidth, smaller files are sensitive to delay.

   1. **Open a file**

      Opening a file typically involves establishing a connection to the remote server and verifying access permissions, so it may involve a delay. However, once the connection is established, opening the file itself is usually not very bandwidth-intensive. Therefore, this operation is more likely to be delay-sensitive.

   2. **Read the contents of a file**

      Reading the contents of a file requires transferring data from the remote server to the client, so it is usually more bandwidth-intensive than delay-sensitive. However, the delay may still be a factor if the remote server needs to perform any processing before sending the data.

3. **List the contents of a directory**

   Listing the contents of a directory involves transferring a list of filenames and metadata from the remote server to the client. The amount of data transferred is typically small, so it is usually not very bandwidth intensive, it is most likely delay-sensitive.

4. **Display the attributes of a file**

   Displaying the attributes of a file typically involves transferring a small amount of metadata from the remote server to the client, so it is usually not very bandwidth-intensive, it is most likely delay-sensitive.

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?

Packet size of 1000:

#packets = $\frac{1*10^6}{1*10^3} = 1000$

Overhead size = 100*$1000 = 100.000\ B$

Loss= packet size + overhead size = $1.000 + 100.000 = 101KB$

Packet size of 5000:

#packets = $\frac{1*10^6}{5*10^3} = 200$

Overhead size = 100*$200 = 20.000\ B$

Loss size = $5.000 + 20.000 = 25KB$

Packet size of 10.000:

#packets = $\frac{1*10^6}{1*10^4} = 100$

Overhead size = 100*$100 = 10.000\ B$

Loss size = $10.000 + 10.000 = 2OKB$

Packet size of 20.000:

#packets = $\frac{1*10^6}{2*10^4} = 50$

Overhead size = 100*$50 = 5.000\ B$

Loss size = $20.000 + 5.000 = 25KB$

We get a lower loss when using data packages of 10.000B

5. Suppose we want to transmit the message 11001001 and protect it from errors using the CRC polynomial $x^3 + 1$

1. **Use polynomial long division to determine the message that should be transmitted.**

   Divisor = 1001

   #Zeros = 4-1=3

   Dividend = 11001001000

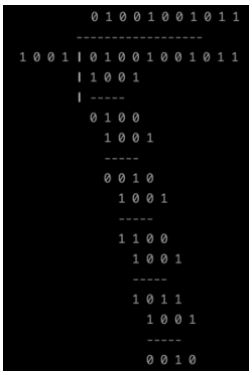
```
     1001
--------------
100|11001001000
  1001
  ----
   0101
   1001
   ----
   0100
   1001
   ----
   0011
```

   The sent message is 11001001011

2. **Suppose the leftmost bit gets inverted in transit. What is the result of the receiver's CRC calculation?**

   Received message = 01001001011

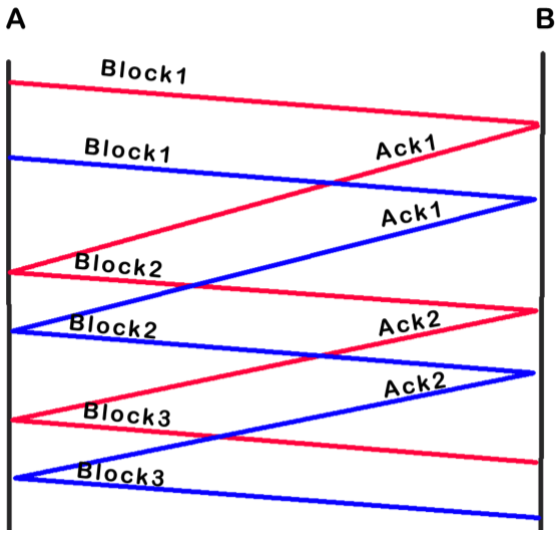
   01001001011/1001 =



   The reminder is different than 0, which means there was 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 to mean the other side has experienced a timeout.

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.



As we can see, the sender keeps retransmitting the duplicate frame immediately upon receiving a duplicate ACK from the receiver. The receiver, in turn, keeps sending ACKs for each received frame, including duplicates. Without a timeout or other mechanism to break the cycle, the duplication may continue indefinitely.

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.

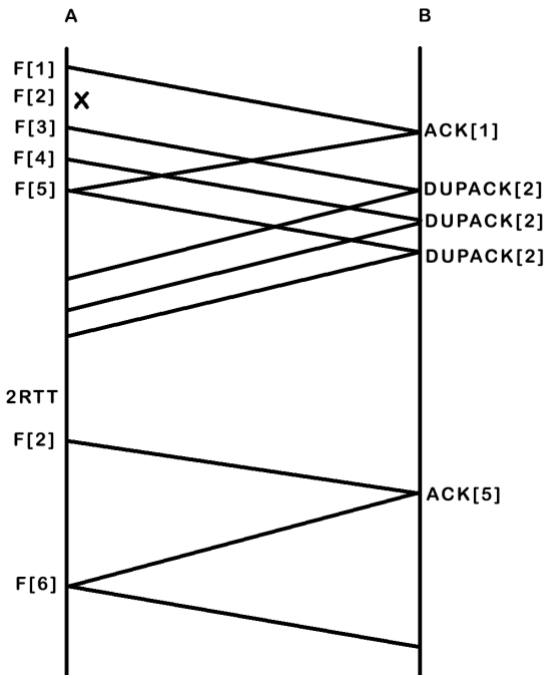
If both sides use the same timeout interval and ACKs are retransmitted if there is no response within the timeout period, a likely scenario for triggering the Sorcerer's Apprentice bug is as follows:

The ACK is lost in transmission, and the sender does not receive it. The sender, after not receiving an ACK within the timeout period, assumes that the data frame was lost and retransmits it. The receiver receives the duplicate data frame and sends an ACK back to the sender.

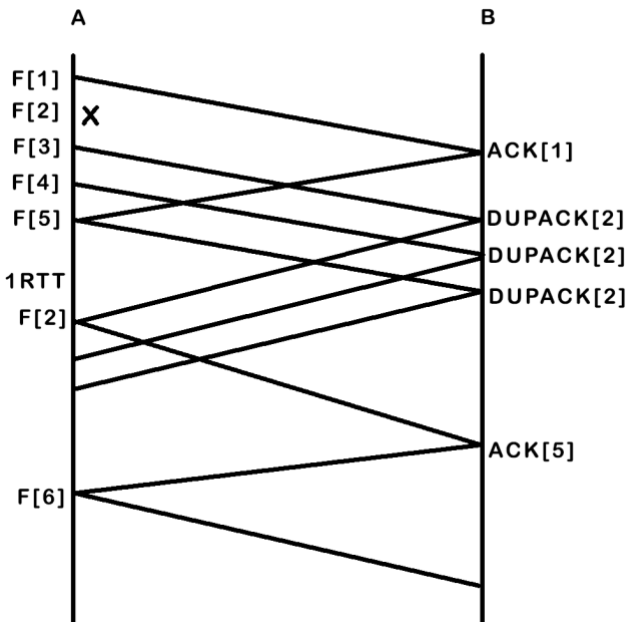
This process repeats indefinitely, with each retransmission adding to the traffic on the network.

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.**



This case had better time performance, reducing transaction time by about 1 RTT