Nombre: Kristian Mendoza

Fecha: 03/03/2023

Curso: Redes

NRC: 4005

Homework 001

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:

   a) The bandwidth is 1.5 Mbps, and data packets can be sent continuously.

      Con los dos RTT del inicio tenemos que:

      $2 * RTT = 2 * 100\ ms = 200ms$

      Se envían los paquetes de 1KB seguidos entonces se puede decir que el paquete en total es de 1000KB, así que el tiempo de transmisión es:

      $$\text{Tiempo de Transmisión} = \frac{\text{Tamaño del paquete}}{\text{Bandwith}} = \frac{1000KB}{1.5\ Mbps} = \frac{1000 * 8 * 10^3 bits}{1.5 * 10^6 bits/s}$$

      $$= 5.333\,\text{s} = 5333.33\ ms$$

      El **tiempo total** es:

      $$Tiempo\ de\ Transmisión + 2 * RTT$$

      $$5333.33\ ms + 200\ ms$$

      $$5533.33\ ms$$

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

Los dos RTT del inicio en total dan 200 ms

El tiempo de transmisión por paquete de 1KB

$$\text{Tiempo de Transmisión} = \frac{\text{Tamaño del paquete}}{\text{Bandwith}} = \frac{1KB}{1.5\ Mbps} = \frac{8 * 10^3 bits}{1.5 * 10^6 bits/s}$$

$$= 0.005333\ \text{s} = 5.333\ ms$$

Al haber 1000 paquetes el tiempo de transmisión total es de 1000 *5.333 ms = 5333.33 ms

Y el tiempo total de los RTT existentes entre cada paquete serán 999 porque el último paquete ya acaba con la transmisión y no se toma en cuenta el RTT que hay después de este. Entones: RTT* 999 = 999 *100 ms = 99900 ms

El **tiempo total** es:

$$= Tiempo\ de\ Transmisión + 2 * RTT + 999\ RTT$$

$$= 5333.33\ ms + 200\ ms + 99900\ ms$$

$$= 105433.33\ ms$$

c) The bandwidth is “infinite,” meaning that we take transmit time to be zero, and up to 20 packets can be sent per RTT.

Los dos RTT del inicio en total dan 200 ms

Hay 1000 paquetes de 1KB que enviar. Si en cada RTT se envían 20. Entonces se calcula cuantos RTT son necesarios para enviar todos los paquetes:

$$\#\ Rtt = \frac{1000\ paquetes}{20\frac{paquetes}{RTT}} = 50$$

Entonces hay 50 RTT y el tiempo de envio es 50* 100 ms= 5000 ms

El **tiempo total** es:

$$= 2 * RTT + 500\ RTT$$

$$= 200\ ms + 5000\ ms$$

$$= 5200\ ms$$

d) 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.

Los dos RTT del inicio en total dan 200 ms

La sucesión que sigue el número de paquetes enviados es de 2^{n-1} entonces hay que encontrar el número de n RTTs tal que la suma de en total 1000 paquetes:

$$1000 = \sum_{i=0}^{n} 2^{i-1}$$

Y esto ocurre cuando n=10. Con el número de paquetes de 1023.5

Entonces deberán haber 10 RTT que dan un tiempo de 10*100ms= 1000 ms

El **tiempo total** es:

$$= 2 * RTT + 10\ RTT$$

$$= 200\ ms + 1000\ ms$$

$$= 1200\ ms$$

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?

Otras propiedades importantes de las direcciones de red pueden ser que:

- Cada tipo de network address tiene un tamaño de bits fijado.
- Son asignadas por algún proveedor, no son aleatorias.
- Tienen una semántica diseñada para permitir a los humanos entender ciertos aspectos de la dirección.
- Permiten realizar un direccionamiento de la información la red

Una situación en la que una network address sea duplicada puede ser en el caso de que se estaba realizando una actualización de addresses en una red y por alguna equivocación se colocó la misma para dos nodos diferentes. Otra puede ser que, con el fin de causar confusiones en la red, una persona se dedique a duplicar direcciones de nodos. Otra puede ser que un proveedor se haya quedado sin direcciones disponibles y duplicó alguna para cumplir con sus ofrecimientos al implementar un nodo.

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:

a) Open a file

Es más sensible al delay. Ya que hay que hacer un request al file server para que hacer un read del file, puede que el frame que viaja sea uno pequeño solo con esta solicitud.

b) Read the contents of a file.

Es más, sensitive al bandwith, ya que un file puede tener un tamaño grande. Y si se lo quiere leer todo, se debe hacer request del archivo completo.

c) List the contents of a directory.

Es más sensible al bandwidth, ya que todos los contenidos de un directorio suelen estar empaquetados en una misma estructura. Por lo que al tener un solo paquete que puede ser de gran tamaño, el bandwidth juega un papel más importante que el delay.

d) Display the attributes of a file.

Los atributos de un file no suelen ser númerosos, por lo que puede que sea un tamaño de información pequeño, así que el delay influye más en esta operación.

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?

Si se pierde un paquete con probabilidad 1, hay que calcular la opción en la que se transmiten el menor número de bytes con el paquete adicional retransmitido.

1000 bytes per packet

Total bytes per packet:

$$Overhead\ bytes + packet\ bytes = 100 + 1000 = 1100\ bytes$$

Number of packets:

$$Num\ of\ packets = \frac{(1 * 10^6\ bytes)}{1000\ bytes} = 1000$$

Total Bytes send:

$$(Total\ bytes\ per\ packer * Number\ of\ packets) + Lost\ packet\ retransmit$$

$$1100\ bytes * 1000\ packets + 1100\ bytes$$

$$= 1101100\ bytes$$

5000 bytes per packet

Total bytes per packet:

$$Overhead\ bytes + packet\ bytes = 100 + 5000 = 5100\ bytes$$

Number of packets:

$$Num\ of\ packets = \frac{(1 * 10^6\ bytes)}{5000\ bytes} = 200$$

Total Bytes send:

$$(Total\ bytes\ per\ packer * Number\ of\ packets) + Lost\ packet\ retransmit$$

$$5100\ bytes * 200\ packets + 5100\ bytes$$

$$= 1025100\ bytes$$

10000 bytes per packet

Total bytes per packet:

$$Overhead\ bytes + packet\ bytes = 100 + 1000 = 10100\ bytes$$

Number of packets:

$$Num\ of\ packets = \frac{(1 * 10^6\ bytes)}{10000\ bytes} = 100$$

Total Bytes send:

$$(Total\ bytes\ per\ packer * Number\ of\ packets) + Lost\ packet\ retransmit$$

$$10100\ bytes * 100\ packets + 10100\ bytes$$

$$= 1020100\ bytes$$

20000 bytes per packet

Total bytes per packet:

$$Overhead\ bytes + packet\ bytes = 100 + 20000 = 20100\ bytes$$

Number of packets:

$$Num\ of\ packets = \frac{(1 * 10^6\ bytes)}{20000\ bytes} = 50$$

Total Bytes send:

$$(Total\ bytes\ per\ packer * Number\ of\ packets) + Lost\ packet\ retransmit$$

$$20100\ bytes * 50\ packets + 20100\ bytes$$

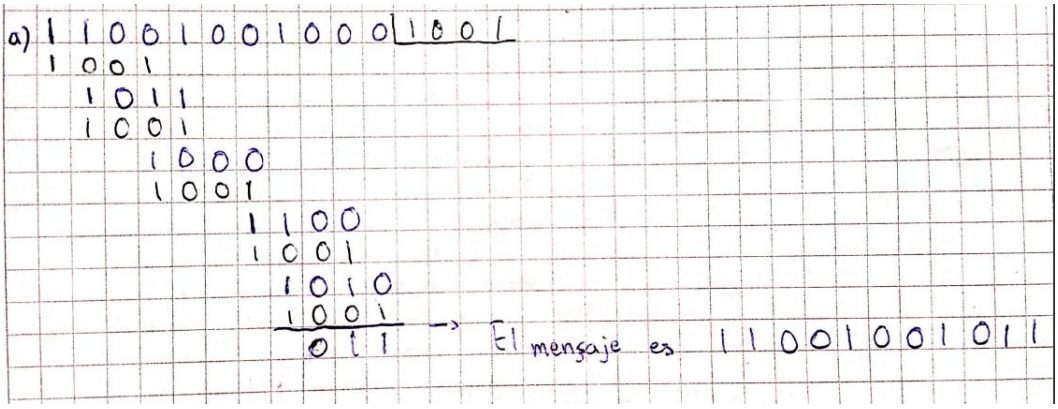
$$= 1025100\ bytes$$

El tamaño de paquete en el que se envían menos bytes es: 10000 bytes. Ya que se envían 1020100 bytes.

5) Suppose we want to transmit the message 11001001 and protect it from errors using the CRC polynomial x3 + 1

   a) Use polynomial long division to determine the message that should be transmitted.

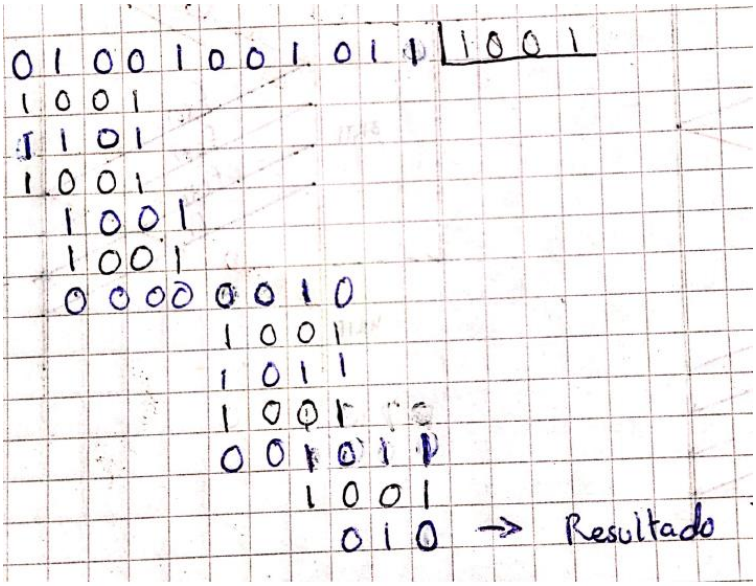
   Se colocan 3 ceros al final del mensaje que se quiere enviar y se realiza la división para el polinomio x3+1 es 1001



El remainder de la división XOR es 011 entonces el mensaje que debe ser transmitido es 110001001011

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

Al invertir el bit más significativo se tiene que: 01001001011

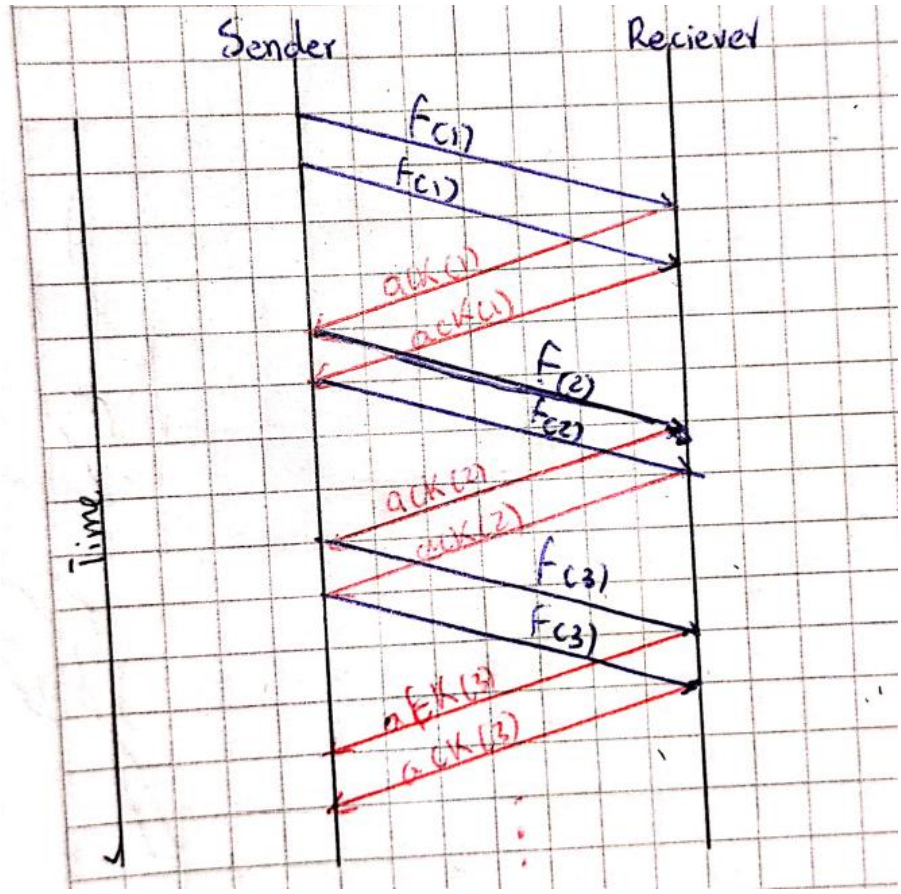


Y como el remainder no es 0, el Reciever sabe que el mensaje llegó corrupto.

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.

a) 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.



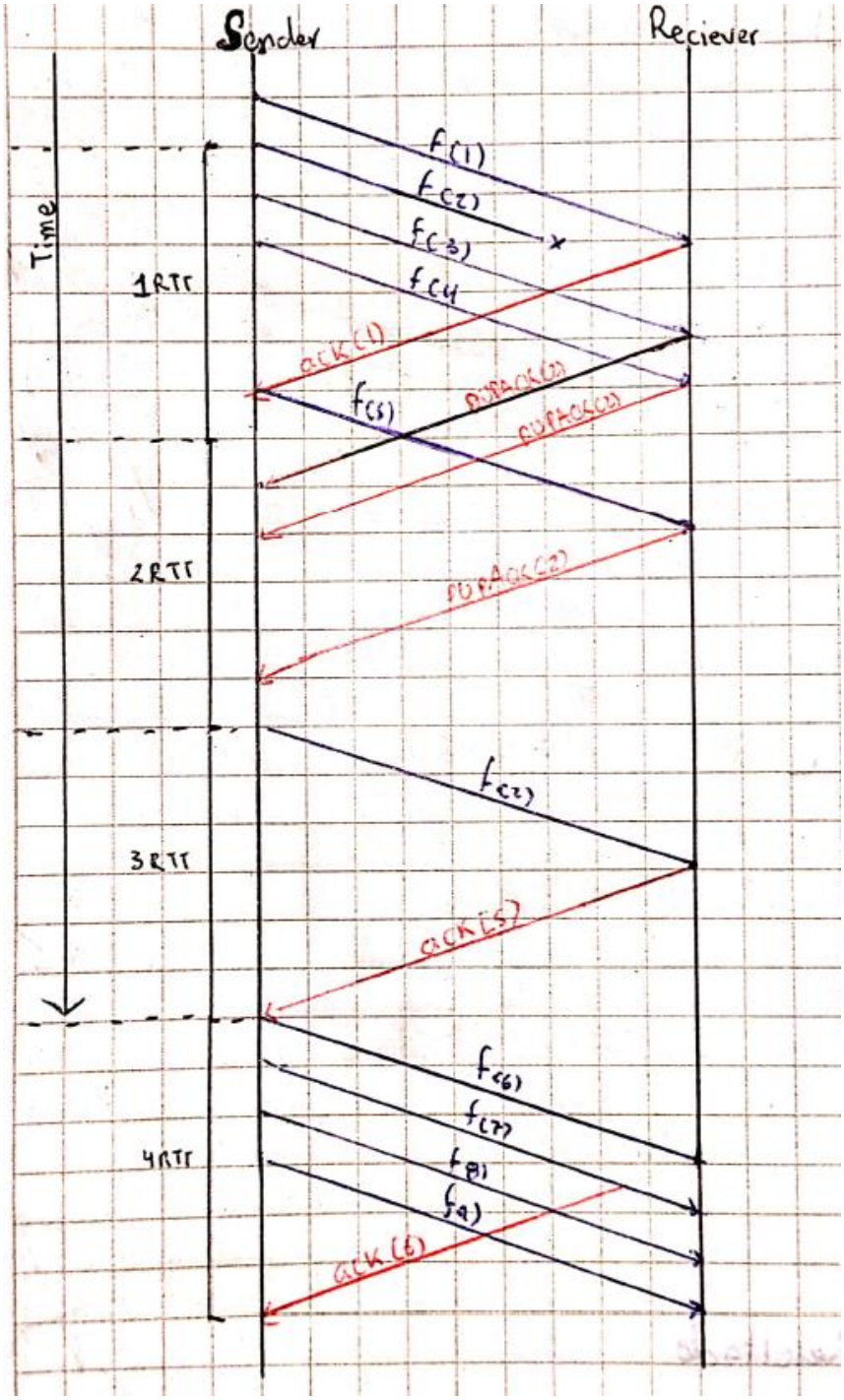
Las duplicaciones van a continuar indefinidamente, lo que hace que la red sea ineficiente.

b) 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.

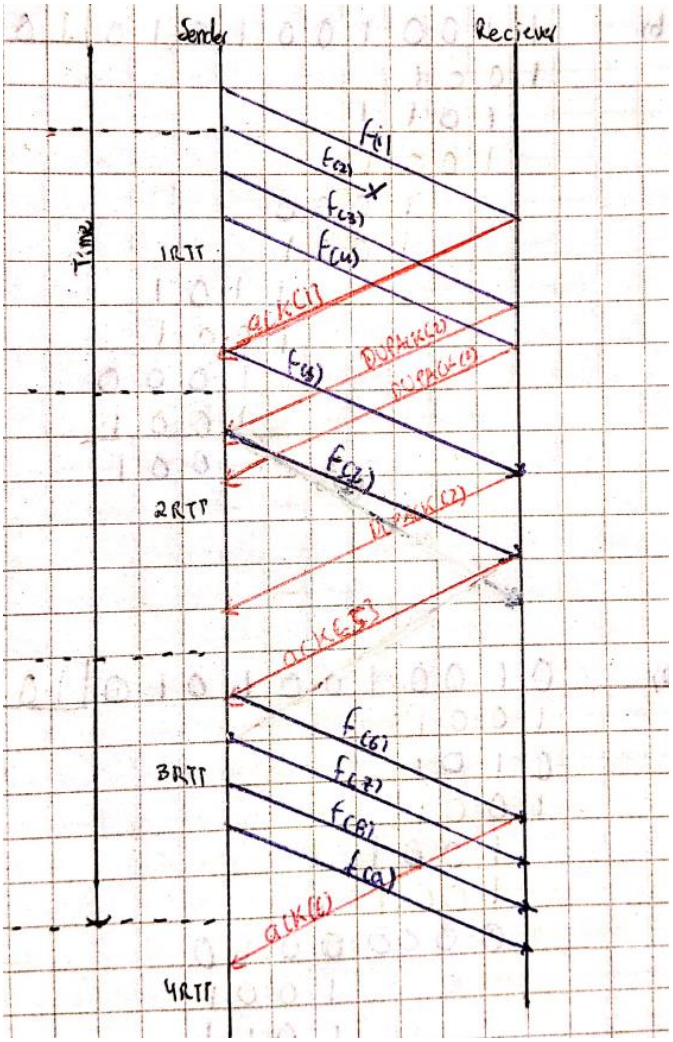
Un escenario posible es que el Sender envía un frame [1] al Reciever, el frame llega y el Reciever envía un ACK [1] al Sender. Pero en el ACK se ve retrasado por condiciones de la red. Este retraso podría sobrepasar el tiempo de espera del Sender, por lo cual vuelve a enviar el frame [1]. Al Reciever le llega el frame [1] y envía el ACK [1] de nuevo y el Sender lo recibe. Pero el ACK [1] retrasado llega de todos modos al Sender, por lo cual ahora transmite el frame [2] dos veces. Entonces en al Reciever le llega el frame [2] y su duplicado. Entonces envía el ACK [2] dos veces, luego el Sender recibe 2 ACK y envía doblemente el frame [3] y las duplicaciones continúan indefinidamente.

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.

a) Frame 2 is lost. Retransmission takes place upon timeout (as usual).



b) 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.



Sí, este approach reduce el tiempo para renviar un paquete perdido. Ya que en este caso luego de 2 RTTs ya se envían los siguientes 4 frames. Mientras que en el anterior literal se los envía después de 3 RTTs.