Guía de Ejercicios - 3

Nivel de Transporte

1. Responder las siguientes preguntas:
    a. ¿Cómo se identifica un socket UDP? ¿Y uno TCP? ¿Hay alguna diferencia en la forma de identificación de ambos protocolos?
    b. ¿Es posible que una aplicación transfiera datos de manera confiable a través del protocolo UDP? Si la respuesta es negativa, explicar las razones. Si la respuesta es positiva, describir de qué manera se lograría.
    c. Cierto host calcula el *checksum* para un segmento UDP recibido y encuentra que coincide con el valor que se incluye en el campo correspondiente. ¿Se puede estar absolutamente seguro de que no se han producido errores?
2. Responder las siguientes preguntas, justificando cuidadosamente cada una de las respuestas:
    a. ¿Qué utilidad tiene la ventana de recepción `rwnd` en TCP? ¿Por qué no existe tal concepto en UDP?
    b. Si tuviésemos la garantía de que ningún router en Internet es susceptible a experimentar pérdida de paquetes, ¿sería provechoso dejar de utilizar TCP como protocolo de transporte en favor de UDP?
    c. En el marco del control de congestión de TCP, ¿qué impacto tendría transicionar a Fast Recovery cuando se observa sólo un ACK duplicado en vez de tres? (Es del próximo tema)

3. Se tienen dos hosts que desean intercambiar datos. La capa de red en el host de origen acepta segmentos provenientes de la capa de transporte de hasta 1200 bytes de tamaño. Además, sabemos que la implementación de la capa de red nos garantiza entrega confiable del segmento de transporte a la capa de transporte en el host de destino.
    a. Describir un protocolo de capa de transporte que sea lo más simple posible para permitir que los hosts puedan comunicarse. Asumir números de puerto de 4 bytes (tener en cuenta que pueden existir múltiples procesos de aplicación ejecutándose en el host de destino).
    b. Modificar este protocolo de forma tal que proporcione un "puerto de retorno" al proceso destino.

c. En estos protocolos, ¿cuál es el rol de la capa de transporte en el núcleo de la red?

4. Supongamos que dos hosts A y B quieren iniciar una conexión con un servidor https S mediante TCP. Dar ejemplos de posibles sockets involucrados en los siguientes:
   - Segmento enviado de A a S
   - Segmento enviado de B a S
   - Segmento enviado de S a A
   - Segmento enviado de S a B

b. ¿Puede ocurrir que los valores de puertos en los sockets que se crearon para la conexión de de A a S y de B a S utilicen los mismos valores ? ¿Por qué?

5. Modelar el *Receptor* del protocolo `rdt3.0` utilizando máquinas de estados finitos (FSMs).

6. Supongamos un escenario en el que el RTT es constante y conocido por el host de origen. ¿Sería necesario en este caso preservar el *timer* que utiliza el protocolo `rdt 3.0` visto en las clases teóricas? ¿Por qué?

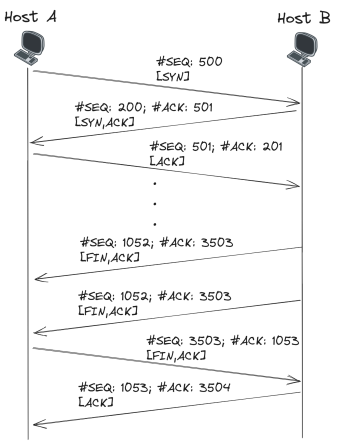
7. Consideremos un canal en el que es posible perder paquetes, pero la demora máxima es conocida. Modificar el protocolo `rdt 2.1` visto en las clases teóricas para incluir el tiempo de espera del remitente y la retransmisión. Argumentar informalmente por qué este protocolo permite comunicarse correctamente, puntualmente ante la pérdida de un paquete de datos o ACK.

8. Supongamos que el host A quiere enviar paquetes simultáneamente a los hosts B y C. A está conectado a B y C a través de un canal tipo *broadcast*: un paquete enviado por A es transportado por el canal tanto a B como a C. Supongamos que, en una transmisión, el canal puede perder o corromper paquetes de forma independiente (por ejemplo, un paquete enviado desde A podría ser recibido correctamente por B, pero no por C). Diseñar un protocolo similar a *stop-and-wait* que permita transmitir paquetes desde A hacia B y C de manera confiable y de manera que A no reciba nuevos datos de la capa superior hasta que no sepa que tanto B como C han recibido correctamente el paquete actual. Describir el protocolo utilizando FSMs para A y B (la FSM para C debe ser esencialmente la misma que para B.

9. Se tienen dos hosts A y B que se encuentran comunicándose a través de una conexión TCP. El host B ya recibió de A todos los bytes hasta el byte número 126. Supongamos que el host A envía dos segmentos consecutivos al host B conteniendo 80 y 40 bytes de datos, respectivamente. En el primer segmento, el número de secuencia es 126, el puerto de origen es 302 y el puerto de destino es 80. El host B envía un ACK cada vez que recibe un segmento del host A.
    a. En el segundo segmento enviado desde el host A al B, ¿cuál es el número de secuencia, el puerto de origen y el puerto de destino?
    b. Si el primer segmento llega antes que el segundo segmento, ¿cuál es el número de ACK, el puerto de origen y el puerto de destino en el segmento de reconocimiento enviado por B?
    c. Si el segundo segmento llega antes que el primero, ¿cuál es el número de ACK en el segmento de reconocimiento enviado por B?
    d. Supongamos que los dos segmentos enviados por A llegan en orden a B. El primer ACK se pierde y el segundo llega después del primer intervalo de tiempo de espera. Suponiendo que no hay otras pérdidas de paquetes, dibujar un diagrama que muestre la transmisión de estos segmentos y los ACKs enviados. Para cada segmento en la figura, indicar el número de secuencia, el número de ACK y el número de bytes de datos.

10. Supongamos que A y B están conectados de manera directa con un enlace de 100 Mbps. Hay una conexión TCP entre los dos hosts, y A le envía a B un archivo muy grande a través de esta conexión. El host A puede enviar los datos de su aplicación a su socket TCP a una velocidad de hasta 120 Mbps, pero el host B puede leer su *buffer* de recepción TCP a una velocidad máxima de 50 Mbps. Describir cómo impacta esto en el control de flujo de TCP.

11. Considerar un intercambio de segmentos TCP entre dos hosts A y B de acuerdo a la figura de la derecha y responder las siguientes preguntas:
    a. ¿En qué estado debía encontrarse el socket de B al comienzo del intercambio?
    b. ¿Qué estados TCP atravesó el socket de A?
    c. ¿Cuántos bytes envió en total el host B?
    d. Identificar un segmento retransmitido y describir a qué puede deberse dicha retransmisión.



12. Cierto proceso en un host A desea establecer una conexión TCP con un servidor B, que se encuentra activo y esperando conexiones.
    a. ¿En qué estado TCP debe estar el socket del servidor que recibe las nuevas conexiones?
    b. Asumiendo que el MSS de ambos interlocutores es de 1 KB, proponer un intercambio de segmentos TCP que satisfaga los siguientes requerimientos (indicar para cada segmento las direcciones IP origen y destino, los puertos origen y destino, los *flags* TCP activados, #SEQ, #ACK y cantidad de bytes del *payload*):
        - Uno de los sockets involucrados debe pasar por los estados `SYN_SENT` y `FIN_WAIT_1`.
        - El proceso del host A debe enviar un total de 3150 bytes.
        - Ambos sockets deben finalizar en estado `CLOSED`.

13. La siguiente captura de segmentos TCP corresponde a un programa corriendo en un host A comunicándose con el servicio corriendo en el puerto 5900 del host B. Los segmentos se *sniffearon* en un hop intermedio entre ambos hosts:

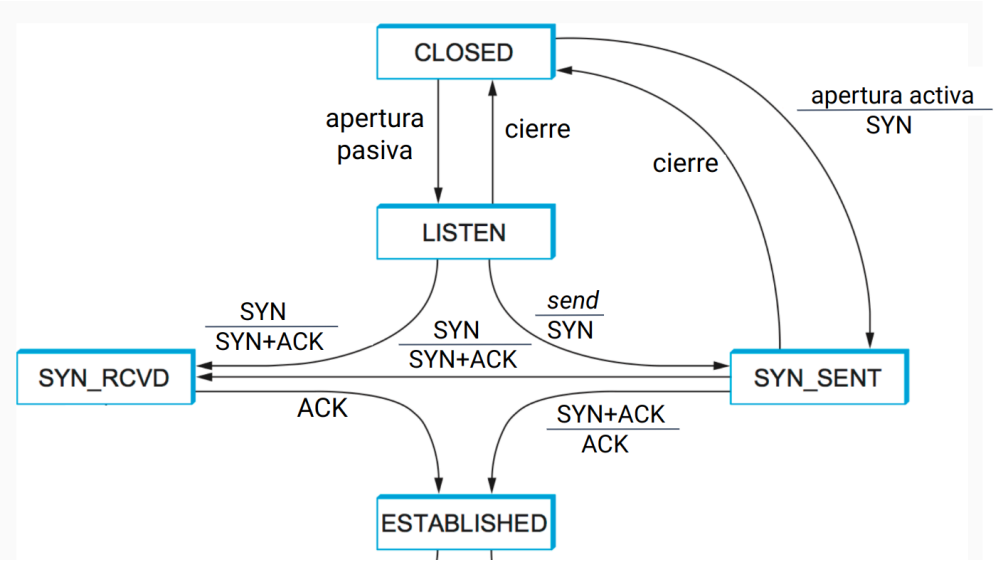
```
No.  Source  Dest  Info
1      A       B    26574 > 5900  [SYN]     Seq=823  Ack=0    Len=0
2      B       A    5900  > 25674 [SYN,ACK] Seq=11   Ack=824  Len=0
3      B       A    5900  > 25674 [SYN,ACK] Seq=11   Ack=824  Len=0
4      A       B    26574 > 5900  [ACK]     Seq=824  Ack=12   Len=0
5      A       B    26574 > 5900  [ACK]     Seq=824  Ack=12   Len=1024
6      B       A    5900  > 25674 [ACK]     Seq=12   Ack=1848 Len=200
7      A       B    26574 > 5900  [ACK]     Seq=1848 Ack=212  Len=0
```

    a. Explicar cuáles segmentos conforman el *three-way handshake* de TCP, describiendo en detalle qué eventos pueden haber disparado cada uno de dichos segmentos.
    b. Extender la captura de forma tal que el proceso en el host A envíe un total de 2000 bytes, el proceso en el host B envíe un total de 200 bytes y ambos finalicen su extremo de la conexión al terminar de enviar sus datos.

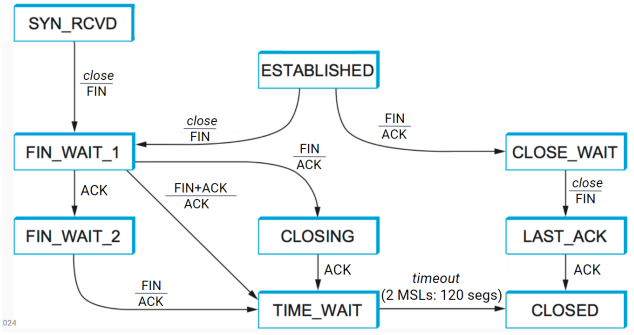
14. A partir de la máquina de estados de establecimiento de conexión TCP de la Figura, describir 3 secuencias de apertura de conexión diferentes entre un Host A y un Host B. Se deberán indicar los estados por los que pasa cada host en cada momento y los segmentos enviados entre los extremos (utilizando flechas). Además, es

necesario indicar #SEQ, #ACK y flags según corresponda. A la hora de diseñar las secuencias, se deberán cumplir con los siguientes requisitos en cada una:

a. La primera debe ser un caso de apertura activa simultánea.
b. La segunda debe contemplar la pérdida de (al menos) un segmento con el flag ACK.
c. La última debe contemplar la pérdida de (al menos) uno con el flag SYN.



15. La figura de la derecha muestra la máquina de estados que modela el cierre de una conexión TCP. Como vimos en clase, el TCP que inicia primero el cierre de la conexión debe permanecer durante cierto tiempo en el estado `TIME_WAIT` antes de que la conexión quede definitivamente cerrada. Explicar qué problema resuelve la inclusión de dicho estado en el protocolo, sustentando la explicación con un intercambio de segmentos entre dos interlocutores TCP que evidencie dicho problema en caso de que el estado no existiese (i.e. si desde `FIN_WAIT_1`, `FIN_WAIT_2` y `CLOSING` se transicionase directo hacia `CLOSED`).



Ejercicios *hands-on*

1. *netstat* es una herramienta de línea de comando que permite mostrar estadísticas y detalles relacionados con la red y las conexiones de red activas en un sistema.
    a. Utilizar esta herramienta para encontrar todos los puertos TCP abiertos en tu computadora y los IDs de los procesos con los sockets respectivos.
    b. Repetir el punto anterior para los puertos UDP.
    c. Identificar qué conexiones TCP están establecidas en tu computadora.
    d. Investigar para qué sirve la herramienta de Linux *ss* (*socket statistics*). Utilizarla para analizar cuántos bytes se enviaron y se recibieron por cada conexión TCP de tu computadora.

2. Investigar cómo ensamblar segmentos TCP en Scapy y cómo es posible enviarlos a la red y recibir potenciales respuestas. A partir de esto,
    a. Enviar un segmento TCP con el flag de SYN al puerto 80 del host del dominio `www.utdt.edu`. ¿Se recibe una respuesta? ¿Qué pasa si ahora enviamos el mismo segmento al puerto 20 del mismo host?
    b. Analizar la respuesta obtenida en el punto anterior. ¿Qué flags tiene el paquete recibido? ¿Cuál es el número de secuencia inicial escogido por el servidor?
    c. Enviar segmentos TCP con el flag de SYN a 1000 puertos diferentes de algún servidor web a elección. ¿Siempre se obtiene una respuesta? ¿A qué se debe?
    d. Repetir el punto anterior pero enviando segmentos UDP en vez de TCP. Analizar los resultados.