**Capítulo 3 Cuestiones de repaso**

**SECCIONES 3.1–3.3**

R1. Suponga que la capa de red proporciona el siguiente servicio: la capa de red del host de origen acepta un segmento con un tamaño máximo de 1.200 bytes y una dirección de host de destino de la capa de transporte. La capa de red garantiza la entrega del segmento a la capa de transporte en el host de destino. Suponga que en el host de destino pueden ejecutarse muchos procesos de aplicaciones de red.

1. Diseñe el protocolo de la capa de transporte más simple posible que entregue los datos de la aplicación al proceso deseado en el host de destino. Suponga que el sistema operativo del host de destino ha asignado un número de puerto de 4 bytes a cada proceso de aplicación en ejecución.

Llamemos a este protocolo Protocolo Simple de Transporte (STP).

En el lado del emisor, STP acepta que el proceso de envío le pase la dirección del destino del host, un número de puerto de destino y un bloque de datos que no exceda los 1196 bytes. STP añade una cabecera de 4 bytes para cada bloque de datos y pone el número de puerto del proceso destino en esta cabecera. Luego STP le pasa a la capa de red la dirección destino del host y el segmento resultante.

Luego en el host de destino, la capa de red entrega este segmento al protocolo STP.

STP examina el número de puerto en el segmento y extrae los datos del segmento para luego pasar esos datos al proceso identificado por el número de puerto.

1. Modifique este protocolo de manera que proporcione una “dirección de retorno” al proceso de destino.

Se modifica STP para que el segmento tenga 2 campos de cabecera: un campo para el puerto de origen y otro para el puerto de destino.

En el lado del emisor, STP acepta un bloque de datos que no exceda de 1192 bytes, una dirección destino del host, un número de puerto de origen, y un número de puerto de destino. STP crea un segmento que contiene los datos, el número de puerto de origen y número de puerto de destino.

Luego, se entrega el segmento y dirección del destino del host a la capa de red.

En el host receptor, después de recibir el segmento, STP entrega al proceso identificado por el puerto destino, los datos que traía el segmento junto con el puerto origen.

1. En sus protocolos, ¿la capa de transporte “tiene que hacer algo” en el núcleo de la red de computadoras?

No, la capa de transporte no tiene que hacer nada en el núcleo, la capa de transporte "vive" en los sistemas finales.

R2. Imagine una sociedad en la que todo el mundo perteneciera a una familia de seis miembros, todas las familias vivieran en su propia casa, cada casa tuviera una dirección única y cada persona de cada casa tuviera un nombre único. Imagine que esa sociedad dispone de un servicio de correos que transporta las cartas desde una vivienda de origen hasta una vivienda de destino. El servicio de correos requiere que (i) la carta se introduzca en un sobre y que (ii) la dirección de la casa de destino (y nada más) esté claramente escrita en el sobre. Suponga también que en cada familia hay un delegado que tiene asignada la tarea de recoger y distribuir las cartas a los restantes miembros de la familia. Las cartas no necesariamente proporcionan una indicación acerca de los destinatarios.

1. Partiendo de la solución del Problema R1, describa un protocolo que el delegado de la familia pueda utilizar para entregar las cartas de un miembro de la familia emisora a un miembro de la familia receptora.

Para el envío de una carta, el miembro de la familia debe darle al delegado la carta, la dirección de la casa de destino y el nombre del destinatario. Luego, el delegado escribe el nombre del destinatario en la parte superior de la carta. El delegado luego pone la carta en un sobre y escribe la dirección de destino de la casa en el sobre, y le da el sobre al servicio de correo del planeta.

En el lado receptor, el delegado recibe el sobre del servicio de correo, saca la carta del sobre, y mira el nombre del destinatario escrito en la parte superior de la carta. Por ultimo, el delegado da la carta al miembro de la familia con este nombre.

1. En su protocolo, ¿el servicio de correos tienen que abrir el sobre y examinar la carta para proporcionar este servicio?

No, el servicio de correo no tiene que abrir el sobre, sino que sólo examina la dirección en el sobre.

R3. Considere una conexión TCP entre el host A y el host B. Suponga que los segmentos TCP que viajan del host A al host B tienen un número de puerto de origen x y un número de puerto de destino y. ¿Cuáles son los números de puerto de origen y de destino para los segmentos que viajan del host B al host A?

Numero de puerto de origen = y

Número de puerto de destino = x

R4. Describa por qué un desarrollador de aplicaciones puede decidir ejecutar una aplicación sobre UDP en lugar de sobre TCP.

Un desarrollador de aplicaciones puede no querer su aplicación para que utilice el control de congestión de TCP, que puede disminuir enormemente la tasa de envío de la aplicación en momentos de mucha congestión.

A menudo, los diseñadores de la telefonía IP y aplicaciones de videoconferencia IP optan por ejecutar sus aplicaciones a través de UDP porque quieren evitar el control de congestión TCP. Además, algunas aplicaciones no necesitan la transferencia de datos fiable proporcionada por TCP.

R5. ¿Por qué razón el tráfico de voz y de vídeo suele enviarse sobre TCP en lugar de sobre UDP en la Internet de hoy día? (Sugerencia: la respuesta que estamos buscando no tiene nada que ver con el mecanismo de control de congestión de TCP.)

Como hoy en día la mayoría de los servidores de seguridad están configurados para bloquear el tráfico UDP mediante firewalls, se puede utilizar TCP para enviar tráfico de voz y vídeo y asi evitar esos bloqueos de seguridad.

R6. ¿Es posible que una aplicación disfrute de una transferencia de datos fiable incluso si se ejecuta sobre UDP? En caso afirmativo, explique cómo.

Sí. El desarrollador de la aplicación puede poner transferencia de datos fiable en la capa de aplicación. Aunque esta no es una tarea fácil que requiere una cantidad significativa de trabajo y de depuración.

R7. Sea un proceso del host C que tiene un socket UDP con el número de puerto 6789. Suponga también que los hosts A y B envían cada uno de ellos un segmento UDP al host C con el número de puerto de destino 6789. ¿Serán dirigidos ambos segmentos al mismo socket del host C? En caso afirmativo, ¿cómo sabrá el proceso del host C que estos dos segmentos proceden de dos hosts distintos?

Sí, ambos segmentos serán dirigidos al mismo socket del host C.

Para cada segmento recibido, en la interfaz de socket, el sistema operativo le proporciona las direcciones IP (de los hosts A y B) al proceso para que pueda determinar el origen de los segmentos individuales.

R8. Suponga que un servidor web se ejecuta en el puerto 80 del host C. Suponga también que este servidor web utiliza conexiones persistentes y que actualmente está recibiendo solicitudes de dos hosts diferentes, A y B. ¿Están siendo enviadas todas las solicitudes al mismo socket del host C? Si están siendo pasadas a través de sockets diferentes, ¿utilizan ambos sockets el puerto 80? Explique y justifique su respuesta.

Para cada conexión persistente, el servidor Web crea por separado un "socket de conexión". Cada socket de conexión se identifica con una tupla de cuatro elementos: (dirección IP origen, número de puerto origen, dirección IP destino, número de puerto destino).

Cuando el host C recibe el datagrama IP, se examinan estos cuatro campos en el segmento para determinar a qué socket se debe enviar el payload del segmento TCP. Por lo tanto, las solicitudes de A y B van a diferentes sockets. El identificador de estos dos sockets tiene 80 para el puerto de destino, sin embargo, los identificadores para estos sockets tienen valores diferentes para direcciones IP de origen. A diferencia de UDP, cuando la capa de transporte envía el payload de un segmento TCP al proceso de la capa de aplicación, no se especifica la dirección IP de origen, ya que esto esta implícitamente especificado por el identificador del socket.

**SECCIÓN 3.4**

R9. En los protocolos rdt estudiados, ¿por qué necesitábamos introducir números de secuencia?

Los números de secuencia son necesarios para que un receptor pueda determinar si un paquete que le llega contiene nuevos datos o es una retransmisión.

R10. En los protocolos rdt estudiados, ¿por qué necesitábamos introducir temporizadores?

Para manejar las pérdidas en el canal. Si el ACK para un paquete transmitido no se recibe dentro de la duración del temporizador para el paquete, el paquete (o su ACK o NACK) se supone que se han perdido. Por lo tanto, el paquete es retransmite.

R11. Suponga que el retardo de ida y vuelta entre el emisor y el receptor es constante y conocido por el emisor. ¿Se necesitaría en este caso un temporizador en el protocolo rdt 3.0, suponiendo que los paquetes pueden perderse? Explique su respuesta.

Un temporizador seguiría siendo necesario en el protocolo rdt 3.0.

Si el tiempo de ida y vuelta es conocido, entonces la única ventaja será que, el remitente sabe con seguridad que, o bien el paquete o el ACK (o NACK) para el paquete se ha perdido, en comparación con el escenario real, donde el ACK (o NACK) todavía podría estar en el camino hacia el remitente, después de que expire el temporizador.

Sin embargo, para detectar la pérdida, para cada paquete, un temporizador de duración constante seguirá siendo necesario en el emisor.

R12. Visite el applet de Java Go-Back-N en el sitio web del libro.

1. Haga que el emisor envíe cinco paquetes y luego detenga la animación antes de que cualquiera de los cinco paquetes alcance su destino. A continuación, elimine el primer paquete y reanude la animación. Describa lo que ocurre.

El receptor de GBN solo reconoce paquetes en orden por lo que al eliminar el primer paquete, solo le llegaran los 4 siguientes los cuales no reconocerá. En este caso el protocolo indica que el receptor además de descartar estos 4 paquetes que le llegaron fuera de orden también debería enviar al emisor un ACK con el número de secuencia del último paquete reconocido. En este caso, esto no sucede en el applet ya que como es la primera prueba que hacemos, el receptor aun no ha reconocido ningún paquete.

1. Repita el experimento, pero ahora deje que el primer paquete alcance su destino y elimine el primer paquete de reconocimiento. Describa lo que ocurre.

Como los paquetes llegan todos y en orden al receptor, por cada paquete que llegue, se enviará el ACK correspondiente al número de secuencia del paquete y se moverá un lugar la ventana.

El emisor maneja ACK acumulativos, o sea que si recibe un ACK con número de secuencia X sabe que todos los paquetes enviados con número de secuencia <= X llegaron al receptor. Esto sucede debido a que el receptor reconoce en orden.

Por este motivo aunque se haya perdido el ACK0, al llegar el ACK1 el emisor sabe que el paquete con numero de seq 0 también llego. Por lo tanto mueve la ventana 2 lugares (por los paquetes 0 y 1) y luego al llegar los ACK con seq 2, 3 y 4 mueve 3 lugares más la ventana.

1. Para terminar, pruebe a enviar seis paquetes. ¿Qué ocurre?

El tamaño de la ventana del emisor en GBN es el máximo de paquetes que se pueden enviar simultáneamente, lo cual sirve para un control de flujo. En el applet utilizado este tamaño es 5 por lo cual es imposible enviar 6 paquetes a la vez.

R13. Repita el problema R12, pero ahora utilizando el applet de Java con repetición selectiva (SR). ¿En qué se diferencian los protocolos SR y GBN?

1. Haga que el emisor envíe cinco paquetes y luego detenga la animación antes de que cualquiera de los cinco paquetes alcance su destino. A continuación, elimine el primer paquete y reanude la animación. Describa lo que ocurre.

El receptor de SR reconoce paquetes en desorden por lo cual al eliminar el primer paquete, solo le llegaran los 4 siguientes los cuales, a diferencia de lo que sucedía en GBN, el receptor si los reconocerá.

Como ninguno de los paquetes reconocidos tiene numero de secuencia igual a la base de la ventana de recepción, a pesar de enviar ACKs la ventana del receptor no se mueve.

Luego, esos 4 ACK llegan al emisor el cual marcará los paquetes 1, 2, 3 y 4 como recibidos pero aun no puede avanzar su ventana ya que todavía no le llego al ACK0 que corresponde con el paquete cuyo número de secuencia es igual a la base de su ventana.

1. Repita el experimento, pero ahora deje que el primer paquete alcance su destino y elimine el primer paquete de reconocimiento. Describa lo que ocurre.

Al llegar todos los paquetes ahora se podrá mover la ventana con lo cual la base de la ventana quedará en la posición 5.

El protocolo SR no maneja ACK acumulativos sino que reconoce individualmente cada paquete. Por lo tanto, para el emisor el efecto de que se haya perdido el ACK0 es el mismo que si se pierde el paquete 0. O sea que cuando le lleguen esos 4 ACK marcará los paquetes 1, 2, 3 y 4 como recibidos pero aun no podrá avanzar su ventana ya que todavía no le llego al ACK0 que corresponde con el paquete cuyo número de secuencia es igual a la base de la ventana.

c. Para terminar, pruebe a enviar seis paquetes. ¿Qué ocurre?

Sucede lo mismo que para el protocolo GBN, ya que el tamaño de la ventana del emisor es el máximo de paquetes que se pueden enviar simultáneamente, lo cual sirve para un control de flujo. En el applet utilizado este tamaño también es 5 por lo cual es imposible enviar 6 paquetes a la vez.

COMPARACION (esto no lo piden pero sirve de resumen):

GBN:

- En el emisor los ACK se consideran acumulativos.

- En el receptor se reconoce en orden.

- Tamaño de ventana del receptor es 1.

- Cuando ocurre un error el emisor reenvía toda la ventana

SR:

- En el emisor los ACK no se consideran acumulativos.

- En el receptor se reconoce en desorden.

- Tamaño de ventana del receptor es mayor a 1.

- El emisor solo reenvía aquellos paquetes que tuvieron error y no toda la ventana.

**SECCIÓN 3.5**

R14. ¿Verdadero o falso?

1. El host A está enviando al host B un archivo de gran tamaño a través de una conexión TCP. Suponga que el host B no tiene datos que enviar al host A. El host B no enviará paquetes de reconocimiento al host A porque el host B no puede superponer esos reconocimientos sobre los datos.

**FALSO**:

TCP brinda un servicio full dúplex por lo que ambos extremos de la conexión pueden enviar y recibir simultáneamente.

1. El tamaño de la ventana de recepción de TCP nunca varía mientras dura la conexión.

**FALSO**:

El tamaño de la ventana de recepción de TCP varía con el control de flujo.

Podría ocurrir que la velocidad del proceso de aplicación para la obtención de datos del buffer de recepción sea menor que la velocidad a la cual llegan datos nuevos al buffer, con lo cual la ventana comenzaría a achicarse producto del mecanismo de control de flujo que posee TCP.

1. Suponga que el host A está enviando al host B un archivo de gran tamaño a través de una conexión TCP. El número de bytes no reconocidos que A envía no puede exceder el tamaño del buffer del receptor.

**VERDADERO:**

El número de bytes enviados no reconocidos es lo que el emisor coloca en su ventana de transmisión. El tamaño de la ventana de transmisión es el mínimo entre el tamaño de la ventana de congestión y la ventana de recepción. La ventana de recepción se establece, justamente a través del control de flujo, a partir del espacio remanente en el buffer de recepción del receptor. Los bytes excedentes no serán enviados por el emisor debido al control de flujo.

1. Suponga que el host A está enviando al host B un archivo de gran tamaño a través de una conexión TCP. Si el número de secuencia de un segmento en esta conexión es m, entonces el número de secuencia del siguiente segmento necesariamente tiene que ser m + 1.

**FALSO:**

En TCP el número de secuencia de un segmento es el número de secuencia del primer byte del segmento. Por lo tanto si el segmento enviado con numero de secuencia m, tenía un tamaño t, el siguiente segmento tendrá numero de secuencia m + t.

1. El segmento TCP contiene un campo en su cabecera para VentanaRecepcion.

**VERDADERO**:

Es un campo de 2 bytes que indica el tamaño de la ventana de recepción y el mismo se envía con cada segmento intercambiado entre los hosts.

1. Suponga que el último RTTMuestra en una conexión TCP es igual a 1 segundo. El valor actual del IntervaloFinDeTemporización para la conexión será necesariamente 1 segundo.

**FALSO**:

El TimeoutInterval necesariamente debe ser mayor al sampleRTT ya que sino se provocaran retransmisiones innecesarias.

1. Suponga que el host A envía al host B un segmento con el número de secuencia 38 y 4 bytes de datos a través de una conexión TCP. En este mismo segmento el número de reconocimiento necesariamente tiene que ser 42.

**FALSO**:

El número de reconocimiento de ese segmento tendrá el número de secuencia del siguiente byte que espera recibir A de B, lo cual no hay datos en la letra para poder saberlo ya que no conocemos el último segmento recibido por A.

R15. Suponga que el host A envía dos segmentos TCP seguidos al host B a través de una conexión TCP. El primer segmento tiene el número de secuencia 90 y el segundo tiene el número de secuencia 110.

1. ¿Cuántos datos hay en el primer segmento?

En TCP el número de secuencia de un segmento es el número de secuencia del primer byte del segmento. Por lo tanto si un segmento enviado con numero de secuencia m, tiene un tamaño t, entonces el siguiente segmento tendrá numero de secuencia m + t.

En este caso 90 + t = 110, por lo cual t = **20 bytes.**

1. Suponga que el primer segmento se pierde pero el segundo llega a B. En el paquete de reconocimiento que el host B envía al host A, ¿cuál será el número de reconocimiento?

TCP reconoce los segmentos en orden y cada vez que llega un segmento con un número de secuencia distinto al que estaba esperando, responde con un ACK cuyo número de reconocimiento es el número del segmento que espera. En este caso cuando A envió el primer segmento, B tendría que estar esperando un segmento con numero de secuencia 90. Como este se pierde y al host B solo le llega el segundo segmento que le envía A, entonces B responde con un segmento ACK cuyo número de reconocimiento es **90.**

R16. Considere el ejemplo de la conexión Telnet de la Sección 3.5. Unos pocos segundos después de que el usuario escriba la letra ‘C’, escribe la letra ‘R’. Después de escribir la letra ‘R’, ¿cuántos segmentos se envían y qué valores se almacenan en los campos número de secuencia y número de reconocimiento de los segmentos?

En el ejemplo un host A es el cliente que envía los segmentos de datos y un host B es el servidor que envía los segmentos de control.

En Telnet cada carácter atraviesa la red 2 veces en el instante que el cliente pulsa una tecla ya que cada carácter escrito por el cliente se envía al servidor y este responde con una copia del carácter que es mostrada luego en la pantalla del cliente. Ese “eco” se utiliza como reconocimiento.

Para el ejemplo suponer que TCP esta utilizando reconocimientos retardados (delayed ack) lo que permitiría a la respuesta del servidor viajar junto con el reconocimiento del carácter sobre el que se hace eco.

En el ejemplo:

1. A envía ‘C’: segmento A-B = [sec=42, ACK=79, datos=’C’]
2. B reconoce y envía ‘C’: segmento B-A = [sec=79, ACK=43, datos=’C’]
3. A reconoce y envía ‘R’: segmento A-B = [sec=43, ACK=80, datos=’R’]
4. B reconoce y envía ‘R’: segmento B-A = [sec=80, ACK=44, datos=’R’]
5. A reconoce ‘R’: segmento A-B = [sec=44, ACK=81, sin datos]

Despues de escribir ‘R’, se envían 3 segmentos (pasos 3, 4 y 5).

**SECCIÓN 3.7**

R17. Suponga que existen dos conexiones TCP en un cierto enlace de cuello de botella con una velocidad de R bps. Ambas conexiones tienen que enviar un archivo de gran tamaño (en la misma dirección a través del enlace de cuello de botella). Las transmisiones de los archivos se inician en el mismo instante. ¿Qué velocidad de transmisión podría proporcionar TCP a cada una de las conexiones?

Respuesta: **R/2**

Aclaracion:

Suponemos que las conexiones tienen el mismo RTT y el mismo MSS, por lo que tienen el mismo tamaño de ventana de congestion y por lo tanto la misma tasa de transferencia.

El algoritmo que utiliza TCP para el control de la congestion hace que el ancho de banda se reparta de forma equitativa entre las conexiones, o sea que la velocidad de transmisión de cada conexión sea R/2.

R18. ¿Verdadero o falso? En el control de congestión de TCP, si el temporizador del emisor caduca, el valor de umbral se hace igual a la mitad de su valor anterior.

**FALSO:**

Se establece en la mitad del valor actual de la ventana de congestión.