

CC51C Comunicación de Datos

Control 2

Profesor: Jens Hardings Perl

Prof. Aux: Sebastián Castro

15 de Junio de 2001

Responda dos de las tres primeras preguntas (1 a la 3). La Px es opcional (bonus), y la Pregunta 4 es obligatoria.

Pregunta 1

- i. En TCP se trata de adivinar la capacidad de la red en base a cuándo se comienzan a perder paquetes. ¿Si se conoce la red local y su comportamiento estudiado en laboratorio, no se podría mejorar ese esquema?

Lo primero que es necesario notar es que la capacidad de la red depende entre otras cosas del uso en el momento que se mide, cosa que no se puede saber a priori en un laboratorio. Si se tratara de una red bajo uso constante, se podría mejorar el comportamiento localmente, pero se pierde generalidad porque la capacidad entre dos puntos de una inter-red depende de muchas redes, y no es siempre posible estudiarlas y modelarlas todas.

- ii. Un implementador de TCP/IP decide incluir un algoritmo que cierra automáticamente conexiones que no están siendo usadas. ¿Qué consecuencias positivas y negativas tiene esto? ¿Qué debiera pasar en una implementación que se adhiere al standard?

Por el lado positivo, se pueden volver a usar los recursos reservados por la conexión en los extremos (los routers no reservan nada en este caso, y tampoco se reserva ni usa ancho de banda), como file descriptors, buffers, etc. Pueden haber casos en los cuales nunca se finalice la conexión, porque por ejemplo el lado que debiera cerrar la conexión está inaccesible, o bien se cayó el sistema y se reinició, haciendo que ya no exista rastro de esa conexión al otro lado. En estas situaciones, la única forma de recuperar los recursos sería la descrita en el problema.

Por el lado negativo, pueden haber conexiones que están activas, a pesar de que no se transmite nada por períodos de tiempo prolongados. Estas conexiones se verían interrumpidas sin motivo si se cierran automáticamente, y la aplicación que la usa no podría seguir funcionando correctamente.

En una implementación que se adhiere estrictamente al standard, las conexiones no tienen un 'idle' time-out, sino que se mantienen activas a menos que se detecte que por ejemplo el otro punto ya la cerró o bien no se puede alcanzar el otro punto para transmitir datos. Los casos críticos debieran poder ser detectados a nivel de aplicación, cerrando la conexión a ese nivel si corresponde.

Pregunta 2

Se define un protocolo para que un cliente pueda leer mensajes de texto desde el servidor. Tanto el cliente como el servidor deben autenticarse el uno frente al otro para que el intercambio sea exitoso. Si el cliente no se autentifica, el servidor no le envía los mensajes, y si el servidor no es autenticado, el cliente no acepta los mensajes. El protocolo es como sigue:

1. Cliente envía a servidor par (username,passwd) y un "Challenge".
 2. Servidor verifica (username,passwd) y responde el Challenge.
 3. Cliente pide mensajes.
 4. Servidor envía mensajes.
- i. ¿Por qué este protocolo no es adecuado para usarlo sobre UDP, pero sí es efectivo usando TCP?

En UDP no existe el concepto de conexión. Por lo tanto, no se puede determinar si los distintos mensajes pertenecen a la misma "sesión". Por lo tanto, no se puede separar la autenticación y los comandos que la necesitan, a menos que se identifique que pertenecen a la misma sesión.

- ii. Modifique el protocolo (simplificándolo) para que pueda usarse sobre UDP. Especifique las restricciones que supone para que funcione correctamente.

- (1) *Cliente envía a servidor par (username,passwd) y un "Challenge". En el mismo paquete pide los mensajes.*
- (2) *Servidor envía el Challenge y los mensajes. El cliente ignora los mensajes si el Challenge no es correcto. Restricción: Todos los mensajes deben caber en un paquete UDP.*

Pregunta 3

Parte I

- i. Para las consultas DNS se usa UDP como protocolo de transporte. Discuta ventajas/desventajas de utilizar TCP en su lugar, para efectos de resolución (consultas) de nombres.

Las consultas DNS son muy cortas, y hay una sola respuesta (que además es idempotente). En este caso, TCP no aporta mucho, y el overhead de establecer y luego cerrar una conexión no se justifica.

- ii. ¿Qué pasaría si se decidiera usar UDP en las transferencias de zona?

Con zonas que son más grandes que el máximo tamaño de paquete UDP, sería necesario dividir la zona en varios paquetes y enviarlos, asegurándose que lleguen todos y en orden. Esto no es trivial (retransmisiones, etc.) y si TCP ya lo implementa bien, para qué arriesgarse con re-inventar la rueda?

Parte II

Determine si para los siguientes casos es recomendable usar RPC para solucionar los asuntos relativos a la transmisión de los datos. Fundamente su respuesta.

- i. En una empresa de servicios, se desea implementar un sistema de facturación, en el cual el servidor de bases de datos tiene toda la información de servicios prestados, y el proceso cliente simplemente hace consultas, realiza los cálculos para determinar cuánto facturar, imprime la factura y finalmente avisa al servidor el resultado del proceso.

Es una buena idea usar RPC, puesto que el objetivo del sistema es generar facturas, lo cual no tiene nada que ver con la transmisión de los datos. Resulta útil poder independizar la transmisión de los datos del funcionamiento general del programa. En este caso, se puede usar algún sistema como ODBC, que es básicamente otra forma de hacer RPC.

- ii. Un sistema que mantiene sincronizados dos jerarquías de directorio entre servidores (mirror). Este sistema actualiza los cambios realizados

en uno de los servidores, y optimiza la copia de archivos para transmitir la menor cantidad de información por la red (por ejemplo, copiar solamente los archivos modificados desde la última vez y no el resto).

En este caso, el objetivo es lograr transferir ciertos datos entre los servidores. Probablemente RPC no sirva demasiado en este caso, y puede más bien interferir en el trabajo de copiado, al no permitir ajustes finos (timeouts, tamaños de paquetes) para optimizar el esquema según el estado o topología de la red, entre otros.

RPC además no está construido para que ambas partes trabajen al mismo tiempo, sino que detiene la ejecución en un lado para continuar en otro.

Px

- ¿Cómo es posible que dos procesos manejen dos conexiones independientes en forma concurrente sin problemas, si van dirigidas a la misma dirección IP y port TCP (como es el caso de servicios donde por cada conexión entrante se crea un nuevo proceso que la atienda)?

Porque una conexión entre a y b se identifica con la tupla (dirección a, port a, dirección b, port b), y si cualquiera de esos números cambia, se sabe que corresponde a otra conexión.

- ¿Por qué es necesaria una capa XDR en un esquema como RPC/RMI?

Esa capa se preocupa de “serializar” los parámetros y el resultado, además de preocuparse de la codificación (network byte order, codificaciones de 7 bits, etc).

- ¿Por qué la llamada “accept” devuelve un nuevo socket en vez de asociar la nueva conexión con el socket ya existente (y que recibe como parámetro)?

Si se usara el mismo socket para enviar y recibir datos, no sería posible obtener una nueva conexión al mismo port al cual está asociado ese socket.

Pregunta 4

Se tiene una ventana de transmisión de 32K, las MTU señaladas, RTT=25ms. Además, debido a las tablas disponibles, los datagramas desde 'host A' a

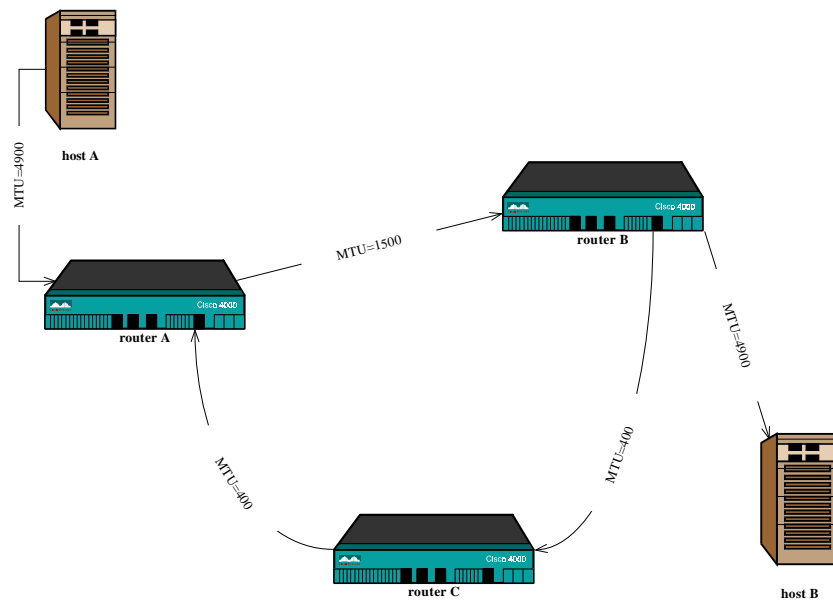


Figure 1: Diagrama de conexión

'host B' pasan por 'router A' a 'router B'. Pero los datagramas desde 'host B' a 'host A' pasan de RA a RB via RC.

- i. Calcule el throughput máximo en una conexión desde 'host A' a 'host B'

*Calcular el throughput máximo era muy fácil, igual a la auxiliar.
RTT=25ms, Ventana de 32K. En un segundo se transmiten efectivamente $1000/25 = 40$ ventanas.*

$$40 * 32K = 1280K$$

Un detalle. Esta situación se cumple debido a que suponemos que todos los segmentos salen juntos, y llegan juntos. Todos los ACK se generan juntos y llegan juntos.

- ii. A su juicio, ¿Cuál será el tamaño del segmento en la ventana? Justifique su respuesta. *La respuesta no era única, y como lo mostraba el diagrama habían tres alternativas. La fórmula de la clase auxiliar decía $MSS=MTU-20$ (20 bytes de header TCP+header IP). Ese valor no es exacto, pero sirve para nuestros efectos.*

Los valores de MTU eran

4900 , si decían que no usan MTU Path Discovery

1500 , si decían que usan MTU Path Discovery

400 si elegían la menor MTU. Esta respuesta no es la mejor, pues por error mío puse esa MTU en el enunciado, lo que en realidad podría causar problemas.

- iii. Supongamos que ahora tenemos delay en cada router de 2ms por cada datagrama IP, recalculé el throughput. *Un delay de 2ms en cada router implicaba un total de 10ms agregados al RTT (de ida pasa por RA y RB, de vuelta por RB, RC, RA).*

Ahora, el error más común fue aplicar el RTT a la ventana. Como los routers agregan delay por datagrama, se produce un fenómeno de

serialización de los segmentos (un segmento TCP se encapsula en un datagrama). Por lo que ya no se cumple que los segmentos salen juntos y llegan juntos (esencial en la parte i).

*Por tanto, el cálculo dice que cada 35ms se envía un segmento. El número de segmentos que se envía en un 1s es 1000/35=29. Dependiendo del valor del segmento que hayan elegido en ii, era el valor del throughput. La fórmula general es MSS*29*

- iv. ¿Cómo modelaría la pérdida de paquetes en su cálculo?

Habían dos respuestas que me parecieron válidas, aunque una era mejor que la otra. La primera decía que si tenemos un a% de pérdida, entonces el throughput real sería

$$T_{real} = T_{ideal} * \frac{1 - a}{100}$$

La otra alternativa era considerar que una red con a% de pérdida produce que a% de los datagramas se retransmiten. A su vez, de ese a% retransmitido, a% se pierde.

$$P_{real} = P_{ideal} - P_{ideal} * \frac{a}{100} - P_{ideal} * \frac{a}{100} * \frac{a}{100} - \dots$$
$$P_{real} = P_{ideal} * \left(1 - \sum_{i=1}^{\infty} \frac{a^i}{100^i}\right)$$

- v. ¿Cómo incluiría en el cálculo un costo de procesamiento en cada punta de 1ms?

El costo de procesamiento en cada punta se le agrega al RTT. El valor agregado es 4ms, pues en meter un datagrama se toma 1ms, en recibirlo al otro lado 1ms, al enviar el ACK se come 1ms, al recibirlo al otro lado otro ms.

*Nuevamente aquí, con ese RTT=29ms, se toma el número de segmentos. 1000/29 = 34 El throughput es 34*MSS*

- vi. ¿El throughput desde 'host A' a 'host B' será el mismo que desde 'host B' a 'host A'?

La respuesta era relativa. El RTT entre A y B es el mismo que entre B y A. Si el ancho de banda era infinito, entonces es igual. Si el ancho de banda no era infinito, no es igual, pues debido a la MTU=400 hay más fragmentación de B hacia A que al revés.