Solución del Examen de
Introducción a las Redes de Computadores y
Comunicación de Datos
(sirc0112.doc)
27 de diciembre de 2001

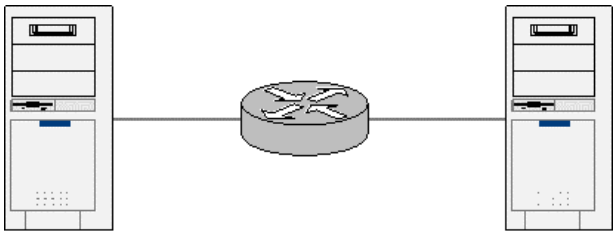
Problema 1.

Sean dos computadoras (A y B), interconectadas mediante una red TCP/IP a la que A está conectada (únicamente) mediante un router R. Desde A hay abiertas dos conexiones TCP a los ports 9001 y 9002 de B, que envían datos continuamente desde A hacia B. B únicamente contesta ACKs.

En la red se producen ocasionales períodos de congestión, durante los cuales la capacidad disponible es suficiente para una de las transmisiones, pero no para ambas. Se ha determinado que es preferible "interrumpir" transitoriamente los envíos al port 9002 manteniendo inafectada la transmisión hacia el port 9001, que experimentar retrasos en ambos.

La mencionada "interrupción" involucra pérdida de información (lo que no siendo que ambas transmisiones transportan voz digitalizada, estimándose aceptable que ocurran interrupciones, y en todo caso preferible a que en ambos canales haya retardos). No obstante, esta pérdida de información deberá resultar transparente a las capas de transporte de A y de B.

Se pide: diseñar y codificar un proceso que ejecutará en el router R interceptando todos los paquetes correspondientes a las conexiones mencionadas, y que cumpla con el objetivo deseado. Se sugiere alterar inteligentemente los campos de secuencia correspondientes a los datos transmitidos de A hacia B y a los ACKs de B hacia A.



Solución:

La solución planteada se basa en el siguiente esquema de funcionamiento:

- Al comienzo de la congestión, descartar paquetes hacia B, llevando la cuenta de la cantidad de bytes descartados. Insertar ACKs hacia A (como si provinieran de B) para esos datos, o bien sumar esa cantidad al campo de secuencia de los ACKs que lleguen de B.

- Al fin de la congestión, restar la cantidad de bytes descartados del campo de secuencia de los datos enviados hacia B (de modo que B no detecte que faltan datos), y coincidentemente sumar dicha cantidad a los ACKs de B hacia A.

Se supone la existencia de un a variable booleana *congestión*, que indica el momento de congestión. Además se cuenta con dos primitivas para envío y recepción de paquetes, desde el equipo A y desde el equipo B.

- `int send_A(const void *msg);`
  Envía a la red un paquete TCP del equipo A contenido en msg, Retorna la cantidad de bytes enviados o –1 (error).
- `int recv_A(void *buf);`
  Recibe de la red un paquete TCP del equipo A que almacena en buf. Retorna la cantidad de bytes recibidos o –1 (error).
- `int send_B(const void *msg);`
  Envía a la red un paquete TCP del equipo B contenido en msg, Retorna la cantidad de bytes enviados o –1 (error).
- `int recv_B(void *buf);`
  Recibe de la red un paquete TCP del equipo B que almacena en buf. Retorna la cantidad de bytes recibidos o –1 (error).

Una vez recibidos los paquetes, se procesarán según su contenido, considerando su puerto origen y destino. Como se modificarán los campos de Número de Secuencia *SN*, y número de ACK *ACK*, para estos campos se recalcula buscando que la diferencia que se mantenga.

El proceso en resumen es el siguiente:

- Si el paquete está originado en la red A, y si el port destino es 9001 se permite su tránsito, sin restricciones, si el port destino es 9002, entonces se procesa según el estado de la congestión. En caso de congestión, se descarta el paquete y se envía un ACK, en caso contrario se envía a la red B, falseando el número de secuencia del paquete *SN*, restándole el valor de la variable *dif_SN*.
- Si el paquete está originado en la red B, entonces si el port origen es 9001 se permite su tránsito sin interferencia, si el port origen es 9002, se pre-procesa para corregir el campo de ACK falseado. Esta corrección implica sumarle al valor del campo *ACK* y *SN*, el valor de la variable *dif_SN*.

Para el estudio de las entradas utilizo una sentencia del tipo *select*, que me permite determinar que entrada requiere ser procesada. La respuesta del *select* es un identificador (*RED_A, RED_B, TIME_OUT*) que permite determinar donde hay datos disponibles para lectura.

Para los paquetes recibidos supongo la existencia de los siguientes campos:

- Port_orig – Puerto origen
- Port_dest – Puerto destino
- IP_orig – IP origen
- IP_dest – IP destino
- SN – Número de secuencia del paquete
- ACK – Número de ACK del paquete.
- Head_length – Largo del cabezal

Se considerarán para simplificar:

- las conexiones ya establecidas, y se supondrá que no hay interrupciones en la conexión.
- que hay una sola conexión por puerto a operar (9001 y 9002).
- que solamente existen conexiones para los puertos mencionados.

```
int ProcesoServidor () {
   int next_SN;
   int dif_SN;
   int aux_SN;
   pack buff,buffsend;

   // Inicializo Variables
   dif_SN=0;
   // configuro select con RED_A, RED_B, TIME_OUT
   conf_select(sel);

   while (TRUE) {
      // Espero paquete de las distintas entradas, RED_A o RED_B.
      if ((status= select (sel)) <> -1 ){
         print("error en Select");
         exit (-1);
      }
      // recibo de la RED_A
      if (status == RED_A) {
         // recibo el paquete
         largo=recv_A(buff);
         // Analizo si port destino == 9001
         if (buff.por_dest == 9001 )
            send_B(buf);
         else {
            // port destino == 9002
            if ( congestion ){
               // Recalculo next_SN y envío ACK
               if ( next_SN >= buff.SN ){
                  largo= largo - buff.head_length;
                  aux_SN= buff.SN - largo;
                  dif_SN=dif_SN + largo;
                  next_SN=buff.SN + largo;
               }
               // Armo y envío ACK a equipo A
               buffsend.port_orig= buff.port_dest;
               buffsend.port_dest= buff.port_orig;
               buffsend.IP_orig= buff.IP_dest;
               buffsend.IP_dest= buff.IP_orig;
               buffsend.data="";
               buffsend.ACK= buff.SN + largo;
               buffsend.SN= buff.ACK;
               send_A(buffsend);
            }else{
               // Rearmo paquete campo SN y envío paquete a lan B
               if ( next_SN >= buff.SN ){
                  largo= largo - buff.head_length;
                  next_SN=buff.SN + largo;
               }
               buff.SN= buff.SN - dif_SN;
               send_B(buff);
            }
         }
      // recibo de la RED_B
      if (status == RED_B) {
         // recibo el paquete de red B
         recv_B(buff);
         // Analizo si port origen == 9001
         if (buff.por_orig == 9001 )
            send_A(buff);
         else {
            // port orig == 9002, Rearmo paquete campo SN y ACK
            // y envío paquete a lan A
               buff.SN= buff.SN + dif_SN;
               buff.ACK= buff.ACK + dif_SN;
               send_A(buff);
            }
         }
   }
}
```

Problema 2.

Es sabido que, al enviar mensajes mediante una conexión TCP, el mero uso del bit de *PUSH* no asegura que, en el destino, la capa de aplicación reciba un mensaje por vez. Se desea realizar una prueba de laboratorio que demuestre este concepto.

Se pide construir un bridge que interconecte los equipos de origen y destino, y manipule el orden de transmisión de los paquetes y eventualmente la segmentación de los mismos, (como podría ocurrir en una red real) de forma que en el destino se experimenten las siguientes situaciones:

a) Arribo simultáneo de dos mensajes a la capa de aplicación de destino.
b) Arribo simultáneo de un mensaje y de parte del siguiente a la capa de aplicación de destino.

Solución:

a) Cambiar el orden de los mensajes. Se envían el 1 y el 2, ambos con push, pero se recibe primero el 2. Al recibirse el 1 se enviarán ambos a la capa de aplicación.

b) Como en a) pero segmentando además los mensajes. Alcanza con que al equipo receptor llegue primero el comienzo del mensaje 2 y luego el mensaje 1 completo.

Problema 3.

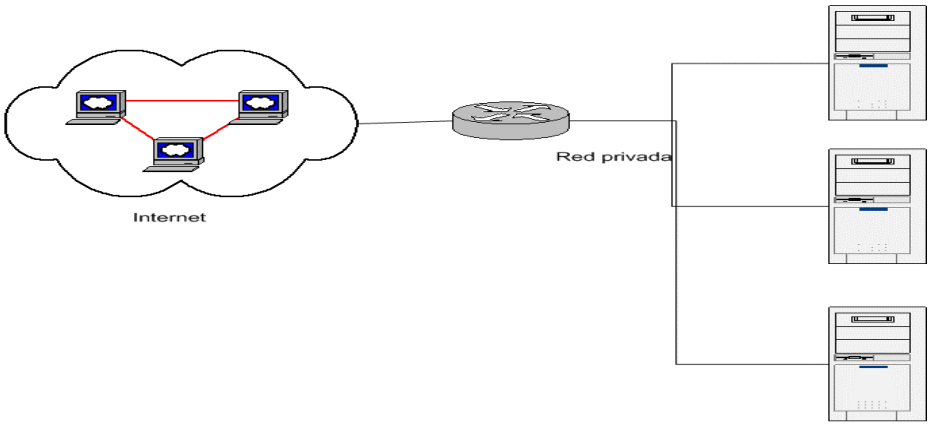
Se cuenta con un equipo que recibe solicitudes de consulta a una página web, (conexiones TCP puerto 80), y redirige las mismas a tres servidores independientes que se encuentran en una red privada no accesible directamente por los usuarios del servicio.

La política de redirección que se desea es round-robin, por lo que permuta en forma circular los requerimientos.

Se pide:

a) Especificar el proceso que se ejecuta en el equipo que recibe las solicitudes. Este debe además chequear en forma periódica el estado del servicio, y en caso de no funcionar suspender el envío conexiones hasta que el mismo se restablezca.

b) Suponga la necesidad de persistencia en la conectividad, lo que implica que una vez que un equipo se conectó una vez a un servidor, durante un tiempo los requerimientos deben ser enviados al mismo servidor (no a los equipos restantes). Explique, que modificaciones deberían realizarse al sistema.



Solución:

a)

Se pide implementar un proceso que redirija los paquetes hacia los servidores de páginas web. Para esto se debe reconocer cuando se inicia una conexión, verificar el cierre de las mismas, y en el intermedio transferir los paquetes de una red a la otra (internet <-> red privada).

Para poder cumplir con los requerimientos debo tener una tabla de redirección, ya que los paquetes TCP/IP todos llegan con la dirección de Internet del equipo. Esta tabla debe permitir saber cada una de las conexiones establecidas hacia donde deben ser redirigidas. La estructura sugerida de la tabla es la siguiente:

```
VAR T: TABLE OF RECORD
          IP_orig : direccionRed;
          Port_orig : integer;
          IP_dest : direccionRed;
          Port_dest : direccionRed;
          Estado : Estado_TCP;
          Tiempo_inicio: time;
       END;
```

Estado_TCP contiene el estado de la conexión con alguno de los siguientes valores: SYN_RCVD, ESTABLISHED, FIN_WAIT_1, FIN_WAIT_2, CLOSE.
Tiempo: contiene un time_stamp de inicio de la conexión. En la solución propuesta no se utiliza, pero se puede implementar un elemento de seguridad, que permita eliminar las conexiones que se han perdido, y el equipo las mantiene.

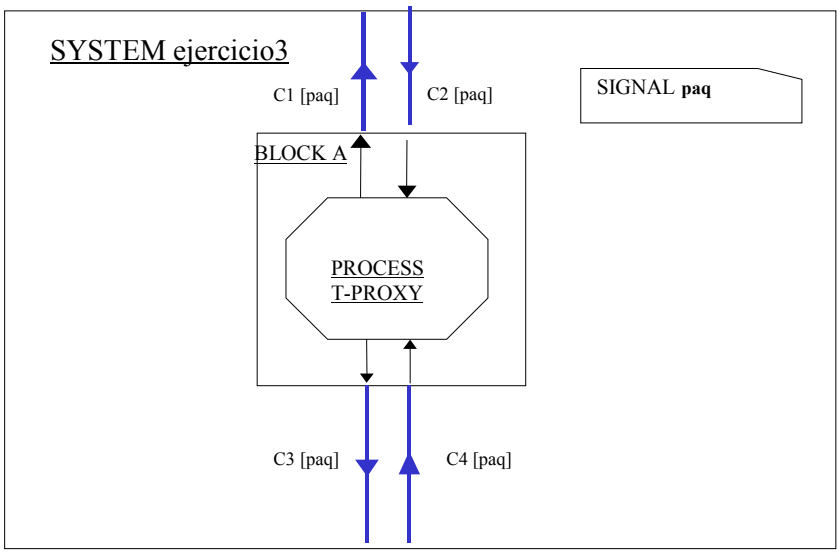
En la solución no se consideran los paquetes TCP/IP recibidos para otros puertos. Estos deberían rutearse, según políticas de routing del equipo.

Se propone modificar el paquete IP, para poder trabajar en forma transparente. El procedimiento resumidamente es siguiente:

- Si se recibe de Internet un paquete con bit SYN, se selecciona un servidor de paginas web destino, y se genera un registro en la tabla T, con la siguiente información, IP_orig del paquete, Port_orig del paquete, IP_dest, con la dirección IP del servidor seleccionado, y Port_dest, con valor 80 (puerto destino), además se agrega en Estado SYN_RCVD, y en Tiempo_inicio un timestamp con el tiempo en que se inicio.
- Si se recibe de la red privada un paquete *paq* con bit SYN, se busca en la tabla T por <*paq.Port_orig, paq.IP_dest, paq.Port_dest*> de la búsqueda se saca el IP_orig. En el paquete se remplaza la dirección *paq.IP_orig* por la dirección Internet del equipo y se envía hacia Internet. Además se modifica el estado a ESTABLISHED.
- Si se recibe un paquete *paq* de Internet con bit FIN, se busca en la tabla T por <*paq.IP_orig, paq.Port_orig, paq.Port_dest*> y de la búsqueda se saca el IP_dest del servidor que lo atiende. Esta IP es remplazada en el paquete en *paq.IP_dest* y se envía hacia la red privada. Si el estado es ESTABLISHED, se pasa a FIN_WAIT_1, si no se elimina el registro de la conexión en la tabla T.
- Si se recibe un paquete *paq* de la red privada con bit FIN, se busca en la tabla T por <*paq.Port_orig, paq.IP_dest, paq.Port_dest*> y de la búsqueda se saca el registro en la tabla T. Se reemplaza en *paq.IP_dest* con la IP del Internet. Si el estado es ESTABLISHED, se pasa a FIN_WAIT_2, si no se elimina el registro de la conexión en la tabla T..
- Si se recibe un paquete *paq* de Internet sin el bit SYN ni FIN, se busca en la tabla T por <*paq.IP_orig, paq.Port_orig, paq.Port_dest*> y de la búsqueda se saca el servidor que atiende la conexión. Esta es remplazada en el paquete en *paq.IP_dest* y se envía hacia la red privada.
- Si se recibe un paquete *paq* de la red privada sin bit SYN ni FIN, se modifica la *paq.IP_orig* del paquete con la dirección Internet del equipo y se envía hacia Internet.

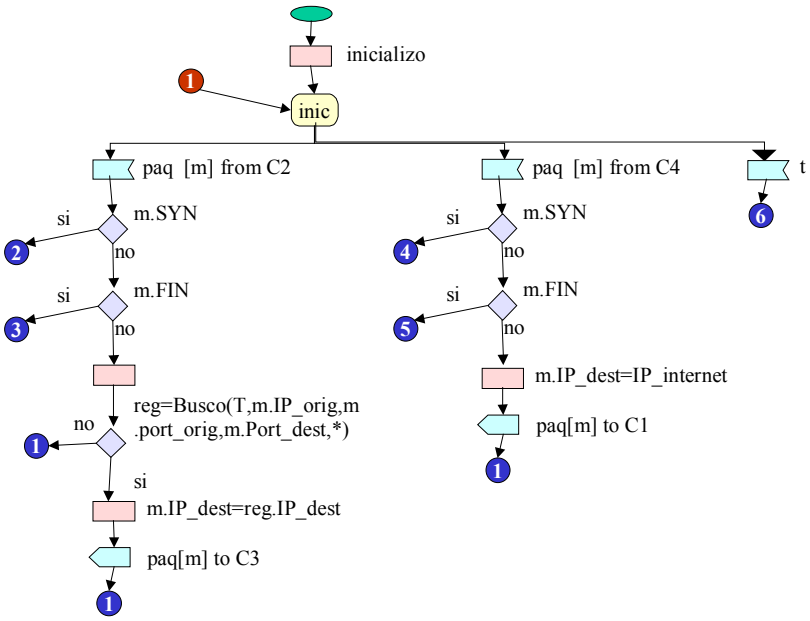
Además cada otro período de tiempo se deberá verificar que el servicio web en los servidores esté funcionando. Para esto se propone que se realice una conexión al puerto 80, lo que permite verificar el funcionamiento del servicio

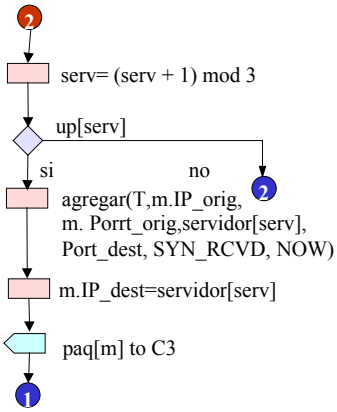
Internet

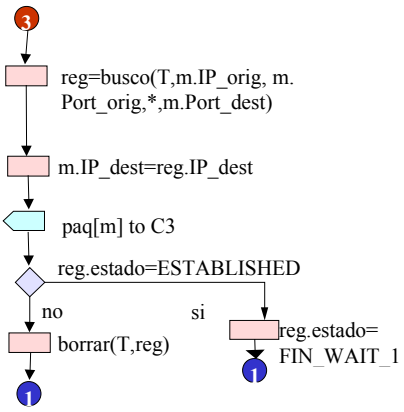


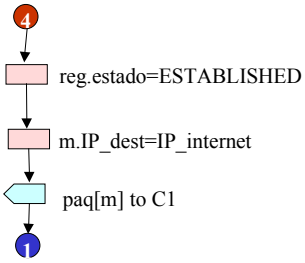
Red Privada

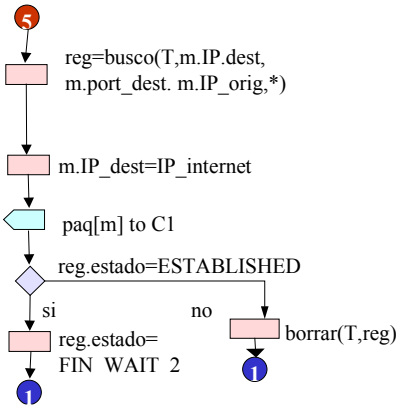
Process T-PROXY

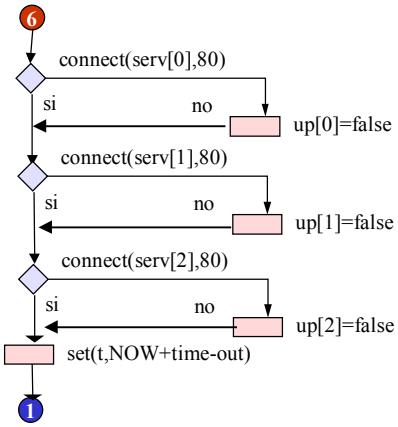



2
serv= (serv + 1) mod 3
up[serv]
si
no
2
agregar(T,m.IP_orig,
m. Porrt_orig,servidor[serv],
Port_dest, SYN_RCVD, NOW)
m.IP_dest=servidor[serv]
paq[m] to C3
1


3
reg=busco(T,m.IP_orig, m.
Port_orig,*,m.Port_dest)
m.IP_dest=reg.IP_dest
paq[m] to C3
reg.estado=ESTABLISHED
no
si
borrar(T,reg)
reg.estado=
FIN_WAIT_1
1


4
reg.estado=ESTABLISHED
m.IP_dest=IP_internet
paq[m] to C1
1


5
reg=busco(T,m.IP.dest,
m.port_dest. m.IP_orig,*)
m.IP_dest=IP_internet
paq[m] to C1
reg.estado=ESTABLISHED
si
no
reg.estado=
FIN WAIT 2
borrar(T,reg)
1


6
connect(serv[0],80)
si
no
up[0]=false
connect(serv[1],80)
si
no
up[1]=false
connect(serv[2],80)
si
no
up[2]=false
set(t,NOW+time-out)
1

Funciones auxiliares utilizadas:

agregar(TABLA,IP_DIR1,PORT1,IP_DIR2,PORT2,ESTADO,TIME_STAMP)
Realiza un alta en la tabla TABLA con la información pasada como parámetros.

busco(TABLA,IP_DIR1,PORT1,IP_DIR2,PORT2)
Realiza una búsqueda en la tabla TABLA, por los argumentos pasados. En caso de algún argumento ser *, implica que puede tener cualquier valor.

connect(IP_DIR,PORT)
Realiza una conexión al puerto PORT de la dirección IP_DIR, y si responde devuelve TRUE, en caso contrario devuelve FALSE.

b)

En la solución propuesta en la parte a), de darse una conexión desde un mismo equipo, podría pasar que se direccionen a diferentes servidores de web.

Para resolver esto se debe implementar algo que dado un cliente origen, siempre direccione al mismo servidor. Esto puede realizarse con una función que dada la IP origen, genere un número entre 0 y 2. Por ejemplo puede ser:
Sea la IP A.B.C.D -> serv = (A+B+C+D) mod 3

Esto asegura que siempre se redireccionará al mismo servidor.

Deben considerarse los casos en que un servidor no este atendiendo, por lo que debería enviarse por ejemplo a uno por defecto.