SOLUCION EXAMEN DE INTRODUCCIÓN A LAS REDES DE COMPUTADORAS
(ref: sirc0102.doc)

2 de Marzo de 2001

Problema 1.

Se dispone de un computador que se comunica con otros mediante TCP/IP. Para hacerlo posee dos tarjetas ethernet que están conectadas a diferentes LAN's.

El sistema conoce:

1) La dirección IP, la máscara de subred y la dirección IP del gateway por defecto (si ha sido especificado) para cada una de las dos tarjetas ethernet.

2) La tabla de ruteo estática, que contiene cero o más entradas del tipo
<dirección de red de destino, máscara de subred, gateway a utilizar>.

Se pide:

- Escribir en un lenguaje de alto nivel el proceso que determina por cual o cuales tarjetas ethernet, se debe enviar un paquetes IP dado la dirección IP destino del mismo.

Solución:

Considero la existencia de la siguientes estructura de datos:

```
Linea_ruteo record of
   Begin
      Dir_IP : IP_addr;     //Dirección IP Ej: 164.73.32.23
      Sn_mask : IP_add;     // Sub net mask Ej: 255.255.255.0
      Def_gw : IP_address   // defaul gateway Ej: 164.73.32.1
   End;
```

Para la información de las tarjetas ethernet:

```
Tarjeta_red: array 1..2 of linea_ruteo;
```

Para la tabla de ruteo estática:

```
Tabla_ruteo: array 1..MAX_ROUTE of linea_ruteo;
```

La función a implementar devuelve los siguientes valores:

0 en caso de no salir por ninguna tarjeta
1 en caso de salir por tarjeta ethernet 1
2 en caso de salir por tarjeta ethernet 2
3 en caso de ser enviado por ambas tarjetas.

```
function selec_tarjeta (dirIP: IP_addr): int
begin

   //Chequeo si están el alguna de las LAN de las tarjetas
   //Utilizo and bit a bit
   if (dirIP and tajeteta_red[1].sn_mask =
       tarjeta_red[1].dir_IP and tajeteta_red[1].sn_mask) then
         //esto implica que la IP pertenece a la LAN de la red 1
         return 1;
   if (dirIP and tajeteta_red[2].sn_mask =
       tarjeta_red[2].dir_IP and tajeteta_red[2].sn_mask) then
         //esto implica que la IP pertenece a la LAN de la red 2
         return 2;

   // chequeo si existe algo definido en tabla de ruteo estatica
   res:= 0;
   for lin=1 to MAX_ROUTE do
      if (dirIP and tabla_ruteo[lin].sn_mask =
          tabla_ruteo[lin].dir_IP and tabla_ruteo[lin].sn_mask) then
            //esto implica la IP se rutea al default gw
            res:= lin;
   if (res > 0 ) then begin
      //debo encontra por donde se sale a la default gateway
      if (tabla_ruteo[res].def_gw and tajeteta_red[1].sn_mask =
          tarjeta_red[1].dir_IP and tajeteta_red[1].sn_mask) then
            //esto implica que def_gw pertenece a la LAN de 1
            return 1;
      if (tabla_ruteo[res].def_gw and tajeteta_red[2].sn_mask =
          tarjeta_red[2].dir_IP and tajeteta_red[2].sn_mask) then
            //esto implica que def_gw pertenece a la LAN de 2
            return 2;
   endif;

   // estudio las default gateway de las tarjetas
   if (existe tajeteta_red[1].def_gw and
       existe tajeteta_red[2].def_gw) then
         //esto implica que envio por ambas tarjetas
         return 3;
   if (existe tajeteta_red[1].def_gw) then
         //esto implica que envio por tarjeta 1
         return 1;
   if (existe tajeteta_red[2].def_gw) then
         //esto implica que envio por tarjeta 2
         return 2;
   else
         //esto implica que no envio por ninguna tarjeta
         return 0;
end;
```

Para después hacer el envío de paquete debo realizar lo siguiente:

- Colocar como dirección IP origen la dirección de la tarjeta ethernet tabla_ruteo[n].dirIP
- Realizar el ARP de la dirección IP destino, en caso de direcciones locales a las redes, o de la dirección del default_gateway en caso de salir por una línea de la tabla de routeo o la default gateway de las tarjetas.

Problema 2.

Se desea hacer posible la comunicación entre equipos existentes que solo manejan UDP (que no se modificarán) y servidores existentes que solo manejan TCP (que no se modificarán).

A estos efectos se tendrá un equipo puente que interactúa con UDP con los equipos y con TCP con los servidores.

Los equipos envían datagramas UDP con los siguientes campos:

1. Servidor (dirección IP, puerto).
2. Datos (deben ser enviados al servidor).

Las conexiones TCP se abrirán en la medida de ser necesarias y serán cerradas por el equipo puente al transcurrir 5 minutos de inactividad en ellas.

Se asume que los paquetes UDP, no se pierden, no se repiten y que llegan en el mismo orden en que fueron generados.

Se pide:

- Escribir en un lenguaje de alto nivel el procesos que se ejecuta en el equipo puente, especificando las estructuras de datos utilizadas.

Solución:

A los efectos de abreviar la invocación a las rutinas, utilizaremos el tipo *dirT*, para describir las direcciones de TCP y UDP.

```
TYPE dirT: RECORD OF
                IP: dirIP,
                Port: long
END;
```

El proceso tendrá una tabla con información de las conexiones abiertas.

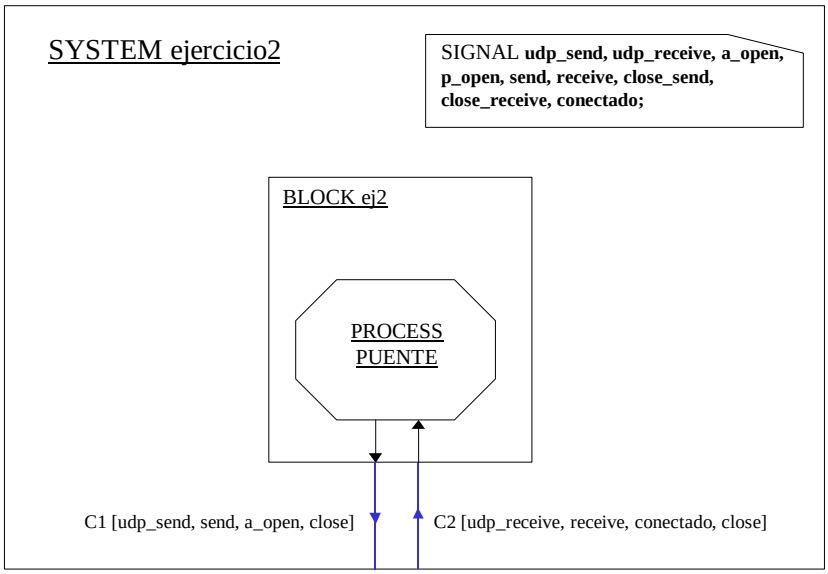
```
VAR tabla: TABLE OF
                equipo: dirT,
                servidor: dirT,
END;
```

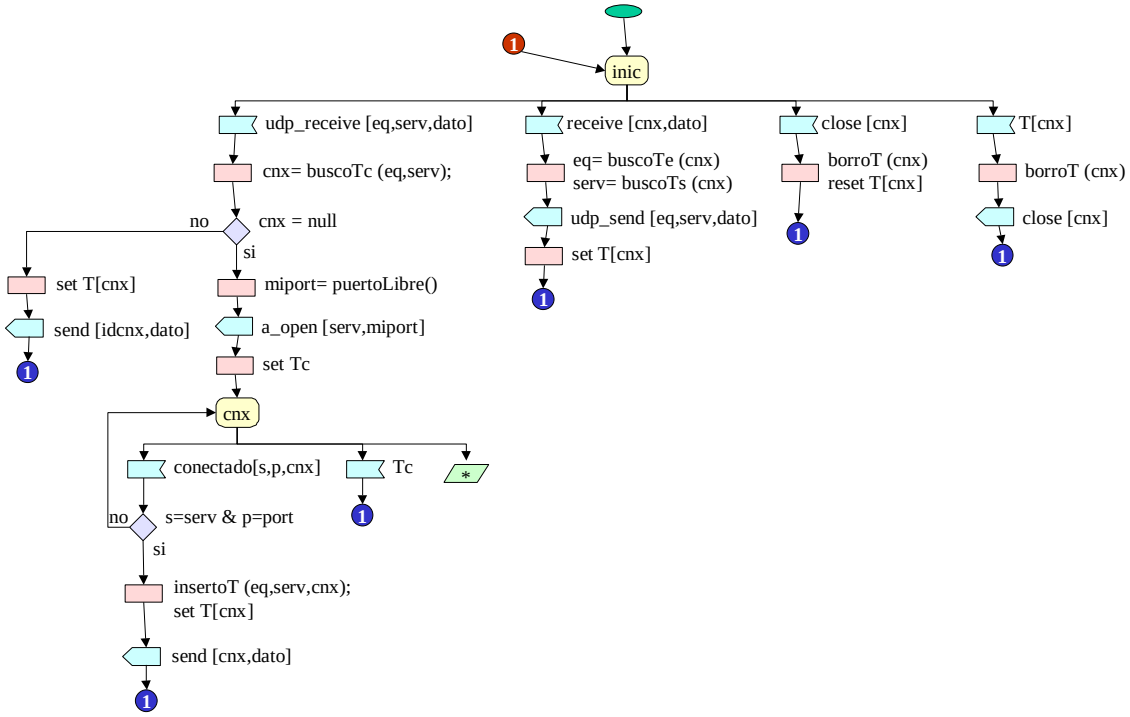
Se tendrán las siguientes funciones para el manejo de la tabla:

- insertoT (equipo: dirT, servidor: dirT, idconexion: long) → Agrega la tupla a la tabla.
- buscoTc (equipo: dirT, servidor: dirT): long → Devuelve el ideconexion asociado a la pareja <equipo, servidor> en la tabla. Si la pareja no está en la tabla devuelve null.
- BuscoTe (idconexion: long): dirT → Devuleve el equipo asociado a idconexión en la tabla.
- buscoTs (idconexion: long): dirT → Devuleve el servidor asociado a idconexión en la tabla.
- borroT (idconexion: long) → Borra la tupla con clave idconexión en la tabla.

La función *puerto_libre* devolverá un puerto libre del equipo puente.

Se tendrá también un array de timers, con un timer para cada conexión. Las función set, seteará el timer en 5 minutos y la función reset lo deshabilitará.


SYSTEM ejercicio2
SIGNAL udp_send, udp_receive, a_open, p_open, send, receive, close_send, close_receive, conectado;
BLOCK ej2
PROCESS PUENTE
C1 [udp_send, send, a_open, close]
C2 [udp_receive, receive, conectado, close]


1
inic
udp_receive [eq,serv,dato]
cnx= buscoTc (eq,serv);
no
cnx = null
si
set T[cnx]
send [idcnx,dato]
1
miport= puertoLibre()
a_open [serv,miport]
set Tc
cnx
conectado[s,p,cnx]
Tc
*
1
no
s=serv & p=port
si
insertoT (eq,serv,cnx);
set T[cnx]
send [cnx,dato]
1
receive [cnx,dato]
eq= buscoTe (cnx)
serv= buscoTs (cnx)
udp_send [eq,serv,dato]
set T[cnx]
1
close [cnx]
borroT (cnx)
reset T[cnx]
1
T[cnx]
borroT (cnx)
close [cnx]
1

Problema 3.

Se dispone de un dispositivo que cuenta con *N* puertos (prt1,.., prtN) conectados a una LAN ethernet cada uno. Los mencionados puertos, ven todos los frames que circulan por la LAN a las que están conectados.

Para cada uno de los puertos ethernet, se cuenta con un buffer de envío y otro de recepción, colocando un frame en el buffer de envío, este saldrá por el puerto, y todos los frames de la LAN son guardados en el buffer de recepción. Se dispone de las siguientes funciones:

- getFrame(prt,frame) que devuelve en frame y elimina del buffer de recepción del puerto especificado en prt, o devuelve NULL si no hay frames.
- putFrame(prt,frame) que coloca el frame en el buffer de envío del puerto especificado en prt.

Sobre esta base se desea implementar un equipo que interconecte las *N* LAN's, con las siguientes características:

- El equipo debe soportar distintos protocolos de capa 3 encapsulados en ethernet.
- Inicalmente solo conoce la dirección ethernet de sus puertos
- Tienda a un manejo optimo de las LAN's

Se pide:

a) Especificar la estructura de datos requerida por el equipo.
b) Especificar en un lenguaje de alto nivel el proceso que debe ejecutarse en cada puerto.

Solución:

Parte a:

El equipo debe tener una estructura de datos que permita determinar en que puerta ethernet se encuentra cada dirección ethernet perteneciente a alguna de las N LAN's.

```
linea_ruteo record of
   begin
      orig_eth_addr: eth_addr;   //Dirección ethernet origen
      port: integer;             //port ethernet de salida
   end;
```

Para resolver el problema debo contar con una tabla que contenga la información de por dónde se han reportado las diferentes direcciones ethernet.

```
tabla_equipo: array 1..MAX_LINES of linea_ruteo;
```

Parte b:

La idea es que el equipo aprenda las direcciones ethernet conectadas a cada una de las puertas ethernet, por medio de la inspección del campo origen de los frames ethernet. Si un frame llego a la puerta i, y su dirección ethernet origen es dir_eth_orig, entonces puedo saber que todos los frames cuyo destino sea dir_eth_orig, deben salir por el puerto i.

Se pide resolver los procesos que corren en cada una de los puertos, los cuales utilizan la estructura de datos definida en la parte a). El puerto que se está atendiendo se encuentra representado por la variable myport.

```
procedure manejo_puerto ()
begin
   while (true) do begin

      // recibo un frame del buffer
      getFrame(myport,buffer);
      if (buffer <> NULL) then begin

         // Busco si la dir ethernet origen esta en la tabla
         if (buffer.dir_eth_orig not in tabla_equipo) then
            // en caso de no estar la inserto
            insert (buffer.dir_eth_orig,myport);

         // si la dir ethernet destino esta en la tabla
         if (buffer.dir_eth_dest in tabla_equipo) then
         begin
            //Si no es para mi LAN local lo envío
            if (tabla_equipo(buffer.dir_eth.dest)<>myport)  then
               // envío frame por el puerto correspondiente
               putFrame(tabla_equipo(buffer.dir_eth_dest),buffer);
         end
         else begin
            // si la dir ethernet destino no esta en la tabla
            for i=1 to N do begin
               // envio por todos los que no sea mi puerto
               if (i <> myport) then
                  putFrame(i,buffer);
            end for;
         end;
      end;
   end while;
end.
```

Este proceso supone buffer como un registro, donde el campo buffer.dir_eth_orig contiene la dirección ethernet origen del frame, y buffer.dir_eth_dest, contiene la dirección destino del frame.

El proceso utiliza las siguientes funciones auxiliares:

`Tabla_equipo(dir_ethr: eth_addr):int`
que devuelve el puerto ethernet de salida indicado en la tabla.

Además utiliza la función `in`, suponiendo que la tabla_equipo se encuentra indizada por la entrada eth_addr. Esta función determina si una dirección ethernet se encuentra en la tabla.

Para un correcto funcionamiento en caso de que equipos pudiesen cambiar de LAN, se podría agregar un reseteo total de la tabla por tiempo, donde el sistema comenzaría a recabar nuevamente información deshenchado la que cuenta.