Examen de Introducción a las Redes de Computadoras y Comunicación de Datos (ref: eirc0302.doc) 26 de febrero de 2003

***Atención**: para todos los ejercicios, suponga que dispone de los tipos de datos básicos (p.ej. lista, cola, archivo, string, etc.) y sus funciones asociadas (ej: tail(lista), crear(archivo), concatenar(string, string).*

Ejercicio 1

Sea un equipo en una LAN, con IP sobre Ethernet, donde se cuenta con una tabla de ruteo expresada según el siguiente formato, no estático dado como ejemplo:

```
================================================
IP routing table
Destination     Gateway         Netmask         Iface
200.40.20.192   *               255.255.255.255 eth0
200.40.20.0     *               255.255.255.0   eth0
127.0.0.0       *               255.0.0.0       lo
default_GW      200.40.20.1     0.0.0.0         eth0
================================================
```

Donde los campos indican los siguiente:

- *Destination*, y *Netmask* informa la LAN que se accede a través del *Gateway* indicado por su IP, o * cuando pertenece a la LAN conectada a la tarjeta. La interface *Iface* marca por donde se debe enviar el frame para llegar al Gateway (Ej. eth0).
- La tabla de ruteo se considera comenzando por la primera opción, y finalizando por la última.

Se cuenta además con:

- ARP estandar, que dada una IP perteneciente a una LAN conectada, devuelve su dirección MAC (Ethernet).
- La primitiva *frame_send(dest: MAC_address,interface: Iface, paq: IP_buff)*, que encapsula el contenido de IP_buff, y lo coloca en la LAN a través de la interfase especificada en Iface, con destino MAC_address.

Se pide:

Implementar el proceso *envioIP(IP_buff)*, que dado un paquete IP, expresado en el buffer de memoria IP_buff, lo envía según la dirección IP destino que contenga el paquete.
El proceso devuelve el entero:

0 – en caso de envío exitoso
1 – en caso de ruta inexistente para el envío
2 – en caso de no obtener MAC_address destino
3 – en caso de otro error.

Solución:

El problema busca la implementación del proceso de envío de un paquete, conociendo la tabla de ruteo, a través de una red tipo Ethernet. Se busca analizar los diferentes resultados de la misma, ya sea error en el envío o éxito. Se supone que el paquete se encuentra en el buffer direccionado por la variable *IP_buff*.

Se supondrá que la tabla de ruteo, similar a la expresada en la letra del ejercicio, está presentada en la siguiente estructura de datos:

```
tabla_ruteo: record of {
                    destination: IP_address;
                    gateway: [IP_address,*]
                    netmask: address_mask;
                    iface:  string;
                    siguiente: ^tabla_ruteo;
              }
```

La idea básica es:

1. Sacar la IP_destino del paquete
2. ir recorriendo la tabla de ruteo, y analizando si la línea resuelve el ruteo del paquete, si lo resuelve voy al paso 3, en caso contrario voy al 4.
3. Si está conectado directamente a una LAN que se accede sin necesidad de un Gateway, resuelvo la dirección ethernet, a través de ARP, y envío, en caso de requerir de un Gateway, resuelvo la dirección ethernet del gateway y envío. Si se envía retorno 0, si no se puede resolver el ARP, devuelvo 2.
4. Retorno 1, por no existir ruta.

Se cuenta con las siguientes funciones auxiliares:

- `IP_dest(paq_IP); Función que recibe un paquete IP, y devuelve la IP destino especificada en el mismo.`

```
Boolean Busco_tabla_ruteo(in  IP:IP_address, out gway:[*,IP_address],
                  out iface:string);
{
      encontre:=FALSE;
      tabla_ruteo:=primer_registro;
      repeat {
            if ( (IP and tabla_ruteo.netmask ) ==
                  (tabla_ruteo.destination and tabla_ruteo.netmask) )
                  encontre:=TRUE;
                  gway:=tabla_ruteo.gateway;
                  iface:=tabla_ruteo.iface;
            } else {
                  tabla_ruteo:=tabla_ruteo.siguiente;
            }

      } until (tabla_ruteo.siguiente == null or encontre)
      return (encontre);
}
```

```
int envioIP (in paq_IP: *byte);
{
      IP_destino:=IP_dest (paq_IP);
      if ( Busco_tabla_ruteo(IP_destino,gateway,iface ){
            if (gateway == “*” ){
                  eth_address:=ARP(IP_destino,iface);
                  if ( eth_address == null ){
                        return (2);             // Error ARP
                  } else {
                        stat:=frame_send(eth_address,iface, paq_IP);
                        if (stat == 0){
                              return (0);       // Envío exitoso
                        } else {
                              return (3);       // Error en envío
                        }
                  }
            } else {
                  eth_address:=ARP(gateway,iface);
                  if ( eth_address == null ){
                        return (2);             // Error ARP
                  } else {
                        stat:=frame_send(eth_address,iface, paq_IP);
                        if (stat == 0){
                              return (0);       // Envío exitoso
                        } else {
                              return (3);       // Error en envío
                        }
                  }
            }
      }
      return (1);       // no existe ruta definida
}
```

Ejercicio 2

Se desea programar un pequeño navegador de internet. Este navegador debe tomar como entrada una página como la siguiente (la cantidad de imágenes es variable):

```
<HTML>
<BODY>
<IMG SRC=”http://host1/directorio1/imagen1.jpg ORDER=1>
<IMG SRC=”http://host2/directorio2/imagen2.jpg ORDER=2>
<IMG SRC=”http://host3/directorio3/imagen3.jpg ORDER=2>
<IMG SRC=”http://host4/directorio4/imagen4.jpg ORDER=3>
</BODY>
```

Nótese que a los tags IMG se ha agregado un parámetro ORDER, que indica el orden en el que se debe realizar la transferencia: primero (y simultáneamente) todas las imágenes con ORDER=1, luego (una vez finalizadas todas las anteriores) todas las de ORDER=2, y así sucesivamente.

Se dispone de un procedimiento *GET (servidor, path_y_archivo, finalizado)*, no bloqueante, del que puede haber más de uno simultáneamente en ejecución, que realiza la operación de DNS (si hace falta), la transferencia del archivo y su presentación en pantalla, indicando el fin de la operación asignando el valor true a la variable “finalizado”.

Se pide:

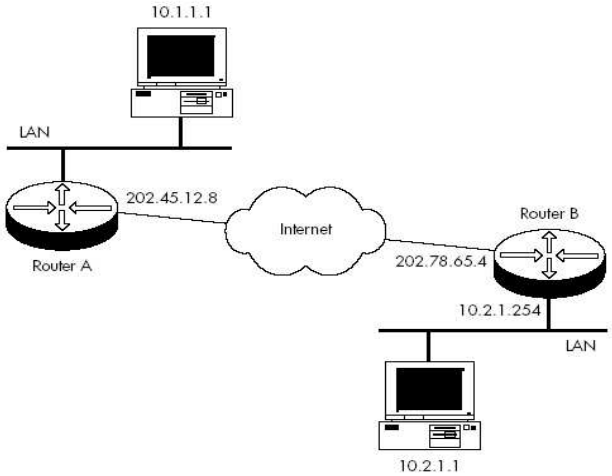
Programar el control de la transferencia de los archivos desde sus servidores de origen. No se considera ningún tipo de cache.

Solución:

Ejercicio 3

Se desea implementar un mecanismo para interconectar las dos subredes privadas físicamente separadas, donde ambas están conectadas a Internet, a través de las direcciones IP especificadas en la figura. Lo que se busca es que para los equipos las redes 10.1.1.X y 10.2.1.X se perciban como interconectadas.

Debe tenerse en cuenta que cualquier dirección privada (10.X.X.X) no puede rutearse a través de Internet, por lo que los equipos mostrados en la figura no son ubicables por internet. Las direcciones válidas son las de las salidas de Router_A y Router_B a internet (202.45.12.8 y 202.78.65.4 respectivamente).



Se pide:

> Diseñar un protocolo que debe ejecutarse en los equipos Router_A y Router_B, que permita solucionar el problema propuesto.

Solución:

Los equipos situados en las LAN, tienen como default gateway el router correspondiente en cada uno de los extremos (Router_A y Router_B).

La idea de las solución es establecer una conexión entre los dos routers, y enviar los paquetes IP generados en la red origen, hacia la red destino, enviándolo dentro de la conexión TCP/IP establecida, para asegurar que no exista pérdida, ni cambio de orden en los paquetes. Al ser recibidos, los paquetes se colocarán en la red destino y llegarán al equipo destino.

Se utilizará un formato de paquete con los siguientes campos:

```
Typedef paquete: record of {
                    largo: integer;
                    dato: string,
                }
```

Donde tendremos:

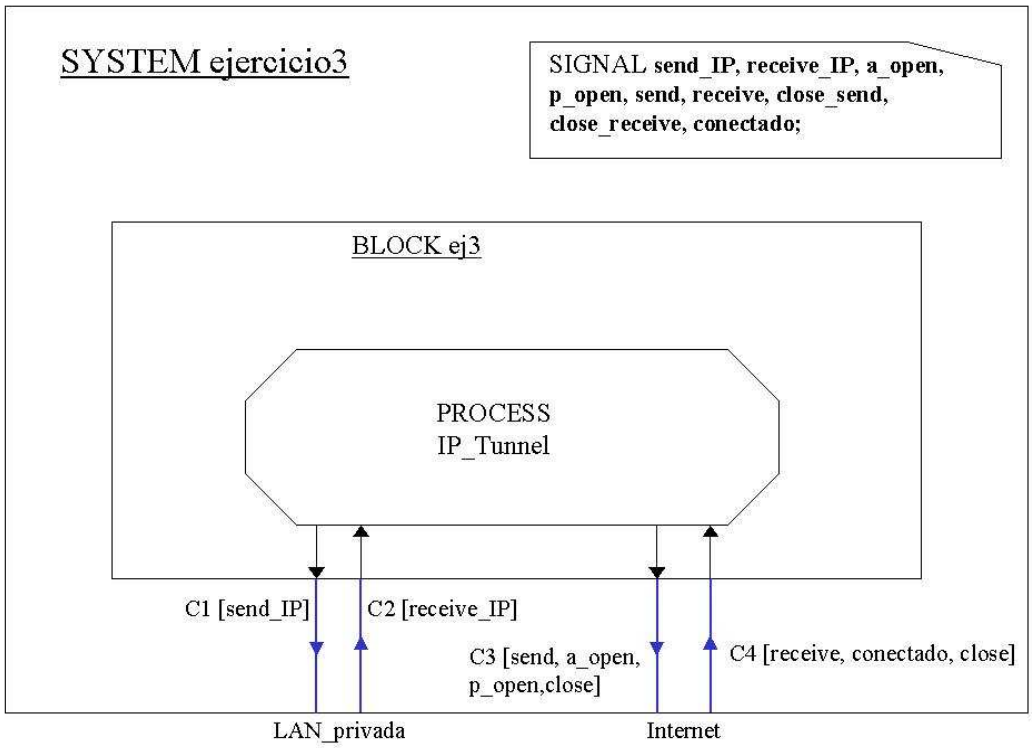
```
Int_paq: paquete;
```

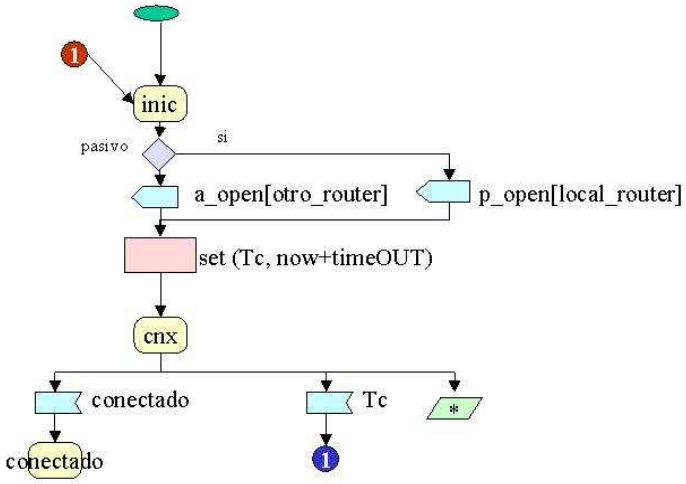
Se cuenta con las siguientes funciones auxiliares:

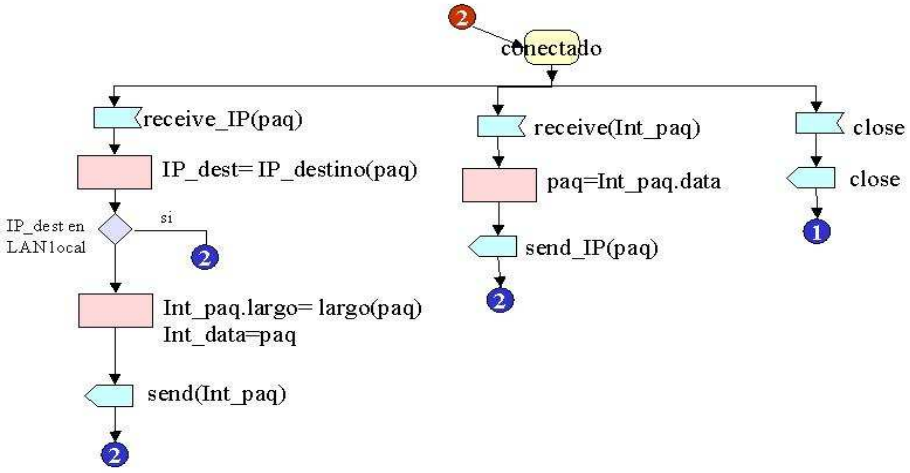
- `IP_dest(paq_IP); Función que recibe un paquete IP, y devuelve la IP destino especificada en el mismo.`
- `largo(paq_IP); que devuelve un entero conteniendo el largo en bytes del paquete contenido.`

Se supondrá que el protocolo define a través de variables locales que equipo realiza una conexión pasiva del TCP/IP, y cual una activa.

El SDL del protocolo es el siguiente:




1
inic
pasivo
si
a_open[otro_router]
p_open[local_router]
set (Tc, now+timeOUT)
cnx
conectado
Tc
*
conectado
1


2
conectado
receive_IP(paq)
IP_dest= IP_destino(paq)
IP_dest en LAN1ocal
si
2
Int_paq.largo= largo(paq)
Int_data=paq
send(Int_paq)
2
receive(Int_paq)
paq=Int_paq.data
send_IP(paq)
2
close
close
1