Solución Examen de Introducción a las Redes de Computadoras
(ref: sirc0008.doc)

12 de Agosto de 2000.

Problema 1

En una red de datagramas, se desea implementar un algoritmo de ruteo basado en aprendizaje reverso (backward learning).
Los routers utilizarán el algoritmo de *aprendizaje reverso* para decidir por donde encaminar los datagramas; pero en caso de ambigüedad (se conoce más de un camino hacia el mismo destino) utilizarán la estrategia de *papa caliente* para decidir entre esos caminos.
Cuando no sabe como llegar al destino, se utilizará también *papa caliente*.
Se partirá de un estado inicial en el que los routers tienen sus tablas vacías.
Los routers conocen las direcciones ethernet de sus enlaces.

Se cuenta con las siguiente funciones:

- enviar (destino: IN direccionEthernet, dtg: IN datagrama)
  que envía el datagrama *dtg* por el enlace hacia *destino*.
- recibir (origen: OUT direccionEthernet, dtg: OUT datagrama)
  que recibe el datagrama *dtg* proveniente del enlace hacia *origen*.
- tamaño_cola (vecino: IN direccionEthernet): integer
  que devuelve el tamaño de la cola del enlace hacia *vecino*.

Se pide:

a- Definir la estructura de las tablas de ruteo.
b- Implementar el protocolo de ruteo que debe correr en los routers.
c- Proponer un mecanismo para contemplar cambios en la topología. (No se pide implementar).

Parte a:

Cada router tendrá una tabla, que indica para cada host, las salidas conocidas. En una implementación más eficiente se debería rutear a subredes, no a cada host individualmente.

```
VAR tabla_ruteo: TABLE OF
      host: dirRed,
      salida: direccionEthernet;
END;
```

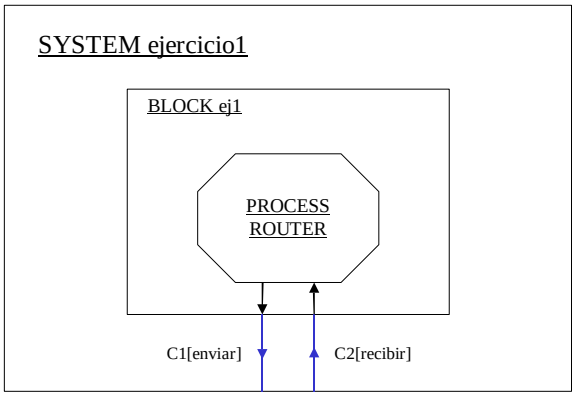
Se guarda también una tabla con las salidas que posee el router hacia otros routers, o redes locales.

```
VAR vecinos: LIST OF direccionEthernet;
```

Parte b:

Funciones auxiliares:

- Para el manejo de la tabla de ruteo:
  - AgregoTablaRuteo (H: DirRed, d: direccionEthernet) → Agrega la pareja <H,d> en la tabla. Si ya estaba, no la agrega nuevamente.
  - BuscoTablaRuteo (H: DirRed, lista: LIST OF direccionEthernet): boolean → Devuelve verdadero si H está en la tabla, y en la variable lista devuelve todas las salidas conocidas para H.
- Para el manejo de listas se usarán las funciones clásicas: vacia, primero, resto, crear, agregar.




```
PROCESS Router
BEGIN
    while true loop
        Recibir (origen, dtg):                      /* recibe un datagrama */
        AgregoTablaRuteo(dtg.origen, origen);       /* guardo info sobre el origen del datagrama */
        esta= BuscoTablaRuteo (dtg.destino, salidas);   /* ruteo el datagrama */
        if (esta) then
            destino= PapaCaliente (salidas);
        else
            destino= PapaCaliente (vecinos);
        endif
        Enviar (destino, dtg);                      /* envía el datagrama */
    endloop
END;

FUNCTION PapaCaliente (salidas: LIST OF direccionEthernet)
BEGIN
    minimo= infinito
    while not vacia(salidas)
        vecino= primero(salidas);
        salidas= resto(salidas);
        largo= TamañoCola (vecino);
        if (largo < minimo) then
            minimo= largo;
            salida= vecino;
        endif
    endloop
    return salida;
END;
```

Parte c:

Al cambiar la topología, puede pasar que un host ya no esté alcanzable por alguna de las salidas que están registradas en la tabla. La solución es tener un mecanismo de envejecimiento para las entradas en la tabla (para cada pareja <host, salida>).
Proponemos mantener una fecha asociada a cada pareja, que registre la fecha de último acceso. La función AgregarTablaRuteo, registra la fecha del sistema al insertar una nueva pareja, y modifica la fecha (la cambia por la del sistema) cuando la pareja ya estaba en la tabla.
Periodicamente debería correrse un proceso que borrara de la tabla, las parejas que no han sido accedidas desde hace x tiempo (fecha del sistema – fecha registrada > x).

```
VAR tabla_ruteo: TABLE OF
        host: DirRed,
        salida: direccionEthernet;
        fecha: date;
END;
```