Introducción a las Redes de Computador{ae}s y Comunicación de Datos

Solución – Marzo de 2012

(ref: sirc1203.odt)

Preguntas Teóricas

Respuesta Pregunta 1 (6 puntos)

a) aplicación, transporte, red, enlace, física.
Aplicación: DNS, HTTP
Transporte: TCP, UDP
Red: IP, ICMP
Enlace: Ethernet, PPP
b) El objetivo de la capa de transporte es comunicar datos entre **procesos**; el de la capa de red, es comunicar datos entre dos **equipos** (o **sistemas finales**) cualesquiera de la red mientras que la capa de enlace apunta a conectar dos **equipos adyacentes** de la red.

Respuesta Pregunta 2 (8 puntos)

- En principio, para enviar request HTTP al servidor web se necesita hacer una consulta al DNS porque expiró el caché. Sin embargo, dado que existe una conexión de transporte y por lo tanto el socket tiene regitrados las direcciones IP de los extremos, no es necesaria la consulta al DNS.
- Se debe entonces insertar el método HTTP GET en el stream provisto por el socket, que normalmente se convertirá (por parte del Sistema Operativo) en un segmento TCP, que es entregado a la capa de red para su procesamiento.
- Se pasa el paquete a la capa de enlace para enviar la trama correspondiente al gateway, pero como no se conoce la dirección MAC del mismo, se debe realizar un ARP request.
- Una vez obtenida la dirección MAC del gateway se envía la trama utilizando esa MAC como dirección destino de capa 2.
- De ahí en más se produce un proceso de enrutamiento del paquete por la Internet hasta que es recibido por el servidor Web, cuya capa de transporte procesa el segmento recibido y entrega los datos a la aplicación, en este caso, el HTTP GET.

Respuesta Pregunta 3 (10 puntos)

a) *(Datagram)* ***length****:* Longitud total del datagrama (incluido encabezado), expresada en cantidad de bytes. Los siguientes tres campos se utilizan para la gestión de la fragmentación y el reensablado posterior. *(Identifier)* ***ID***: Identificador único del datagrama, generado por el origen del paquete. Su valor debe ser igual en todos los datagramas en los que se divide el datagrama original.
(Fragmentation Flag bit; formalmente, *MF flag)* ***fragflag****:* bandera que indica cuál es el último fragmento de un datagrama y *(Fragmentation)* ***offset***: corrimiento, medido en cantidad de bytes, que permite al destino reensamblar correctamente todos los fragmentos en los que se dividió el datagrama original.

b)

i. Una manera conveniente de fragmentarlo (lo que se debería hacer según RFC) es "partirlo" en tres fragmentos; dos fragmentos del tamaño máximo posible (**1500**) y el tercero, del tamaño restante necesario. La cantidad máxima posible de datos de un fragmento en el enlace en cuestión es 1480 bytes (1500 – 20). Por lo tanto, el tercer y último fragmento contendrá 4000 – 20 – 2 x1480 = **1020** bytes de datos.

ii. Para cada fragmento, los valores de los campos indicados en el datagrama de la figura, son:

Fragmento	*Length*	*ID*	*Fragflag*	*Offset*	
1	1500	x	1	0	
2	1500	x	1	185	*porque 1480 / 8 = 185*
3	1040	x	0	370	*porque (2 x 1480) / 8 = 370*

Respuesta Pregunta 4 (6 puntos)

Las funciones de checksum tienen determinada capacidad de detección de errores, pero existen algunos que no pueden ser detectados. Luego, no existe seguridad de que no existan bits erróneos.

Respuesta Pregunta 5 (10 puntos)

a) *Flooding*: estamos en presencia de *flooding* cuando un nodo recibe un paquete y envía copias del mismo a todos sus vecinos (excepto al vecino de quien lo recibió) .
Flooding controlado: estamos en presencia de *flooding controlado* cuando el nodo solo hace broadcast de un paquete si no lo ha enviado antes .

b) *Reverse Path Forwarding* es una técnica para implementar flooding controlado en un entorno multicast. Consiste en lo siguiente: cuando un paquete multicast arriba a una interfaz, el router chequea la existencia de un camino de red desde la interfaz hacia el origen; si existe, el paquete es aceptado, y en caso contrario se descarta para prevenir loops.

c) IGMP: *Internet Group Management Protocol*. Permite gestionar la asociación de los hosts (sistemas finales) a los grupos de multicast intercambiando mensajes con los routers multicast:
Query: desde el router a los hosts
Membership report: desde los hosts a los routers
Leave: desde los hosts a los routers
El router envía mensajes **Query** genéricos (“todos los grupos”) o específicos por grupos, que son respondidos por los hosts con mensajes de asociación **Membership report** (también se pueden enviar mensajes de asociación sin esperar ser interrogados). Cuando un host decide abandonar un grupo de multicast puede enviar un mensaje **Leave** o simplemente dejar que venza la temporización. El router multicast detectará la desconexión en el próximo **Query**. A este comportamiento se le denomina “soft state”, es decir, se debe mantener la tabla de asociaciones en base a preguntas periódicas.

Problemas Prácticos

Respuesta Problema 1 (30 puntos)

Parte 1

```
routeStep findNextHop(ip_addr address){
        route retorno = null;
        int mejorMascara = 0;
        for(route iterador in routeList){
                if(andBitABit(iterador.netmask, address) == iterador.net){
                        // Ruta válida
                        int largoMascara = numberOfBits(iterador.netmask);
                        if( largoMascara > mejorMascara ){
                                // Longest prefix match, mejor ruta hasta ahora
                                retorno = iterador;
                                mejorMascara = largoMascara;
                         }
                }
        }
        return retorno;
}
```

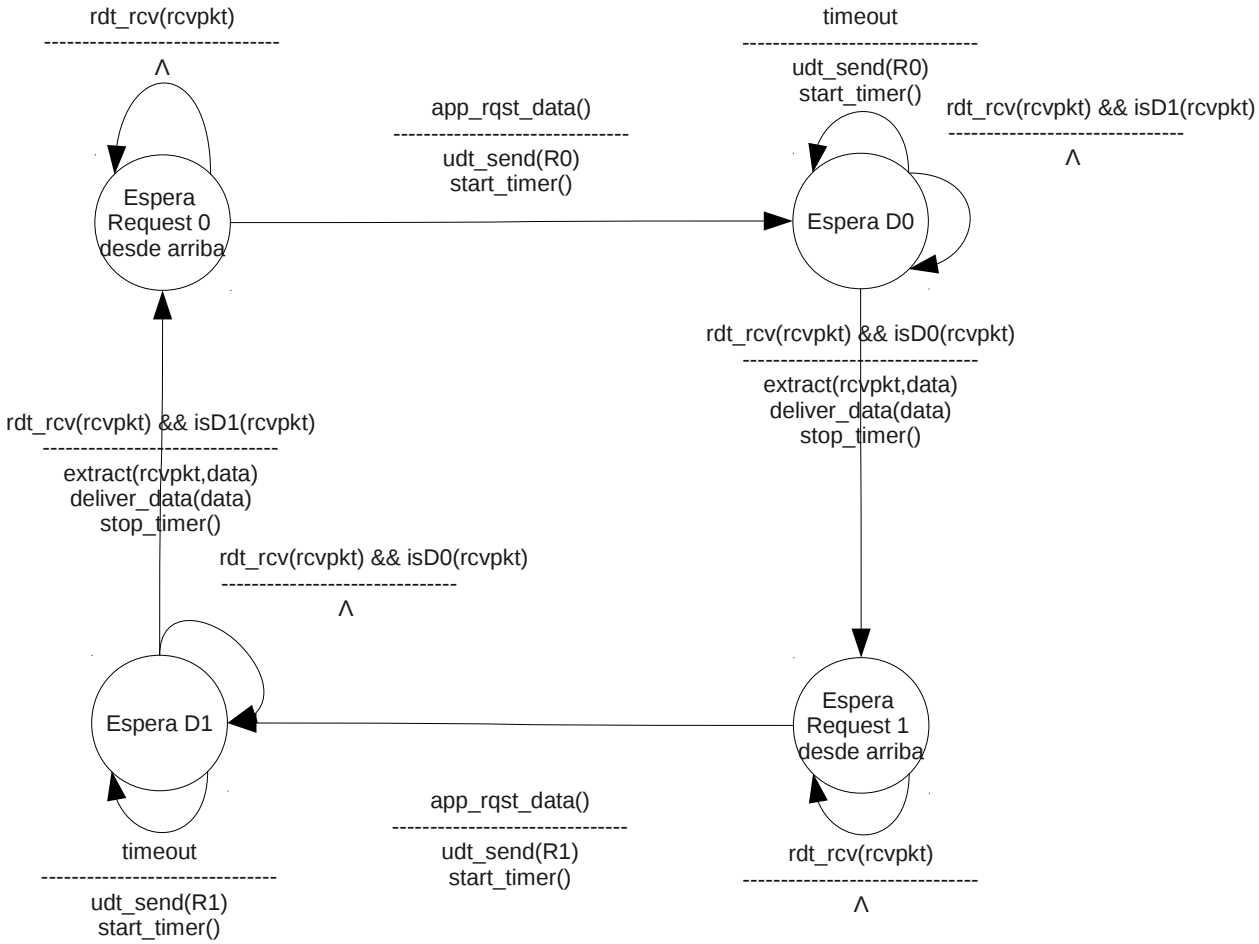
Parte 2

```
void forwardPacket(ip_pack packet){
        route routeToDest = findRoute(packet.dest);
         packet.ttl--;
        if((routeToDest != null) && (packet.ttl >0)){
                if(routeToDest.next_hop != null){
                        // Se lo envio al next hop ubicado en la interfa if.
                        sendTo(packet, routeToDest.next_hop, routeToDest.if);
                }
                else{
                        // El destinatario esta directamente conectado, se lo doy.
                        sendTo(packet, packet.dest, routeToDest.if);
                }
        }
        else {
                route routeToSrc = findRoute(packet.from);
                // Siempre hay una ruta al remitente.
                if(routeToSrc.next_hop != null){
                        sendTo((packet.ttl?createNoRouteError(packet.from):createTTLExpired(packet.from)),
                                routeToSrc.next_hop, routeToSrc.if);
                }
                else{
                        sendTo((packet.ttl?createNoRouteError(packet.from):createTTLExpired(packet.from)),
                                packet.from, routeToSrc.if);
                }
        }
}
```

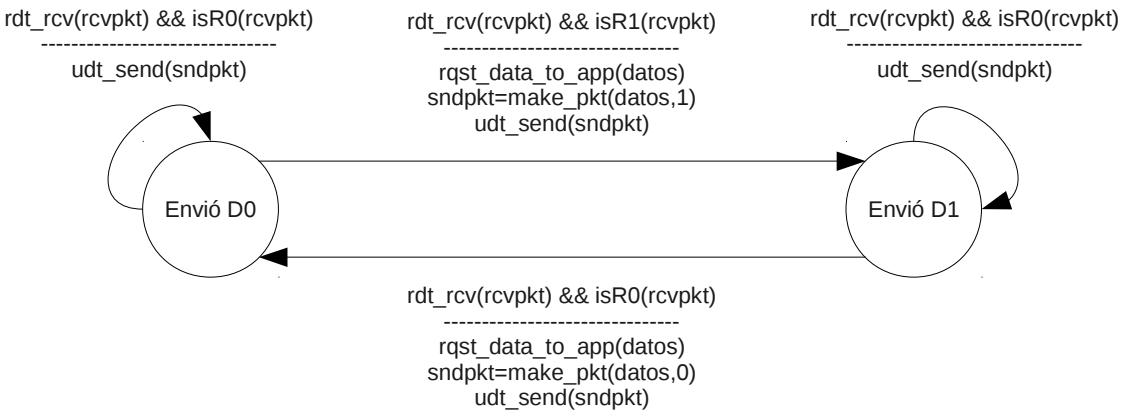
Parte 3

Esto se debe a que si existe un medio compartido, es necesario indicar expresamente a quién se le debe enviar un paquete dado en capa 2. Ya que el next_hop puede no tener que ver con la dirección destino del paquete, la capa 2 no podrá resolver a quién enviarle el paquete en cuestión.

Respuesta Problema 2 (30 puntos)



FSM de A



FSM de B