Redes de Computadoras

Solución Examen – 25 de julio de 2019
(ref: solredes20190725.odt)

Instrucciones

- **Indique su nombre completo y número de cédula en cada hoja.**
- **Numere todas las hojas e indique la cantidad total de hojas que entrega en la primera.**
- **Escriba las hojas de un solo lado y utilice una caligrafía claramente legible.**
- **Comience cada pregunta teórica y cada ejercicio en una hoja nueva.**
- **Solo se responderán dudas de letra. No se responderán dudas de ningún tipo los últimos 30 minutos del examen.**
- **El examen es individual y sin material. Apague su teléfono celular mientras esté en el salón del examen.**
- **Es obligatorio responder correctamente al menos 15 puntos en las preguntas teóricas y 20 de los problemas prácticos. Los puntos ganados en el curso se suman a los puntos de teórico.**
- **El puntaje mínimo de aprobación es de 60 puntos.**
- **Para todos los ejercicios, si es necesario, puede suponer 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).**
- **Justifique todas sus respuestas.**
- **Duración: 3 horas. Culminadas las 3 horas el alumno no podrá modificar las hojas a entregar de ninguna forma.**

Preguntas Teóricas

Pregunta 1 (8 puntos)

a) ¿Qué función cumplen el *broadcast* y el *flooding*?, ¿Qué diferencias existen entre ambos conceptos?
b) De ejemplos de protocolos vistos en el curso que utilizan estas técnicas.
c) Describa el mecanismo de flooding básico.

Solución:

a) Ambos mecanismos cumplen la misma función: enviar una copia de un mensaje a todos los nodos de una red. En el caso de broadcast típicamente se asume un medio compartido (por ejemplo un dominio de broadcast ethernet), donde el origen de la comunicación envía un mensaje a una dirección especial que representa a todos los nodos. En redes con topologías arbitrarias, el envío de un mensaje a todos los nodos se puede implementar mediante flooding.
b) Ethernet 802.3 tiene la dirección FF:FF:FF:FF:FF:FF definida como dirección de broadcast. OSPF utiliza flooding para el envío de información de estado-enlace entre routers; en este caso se utiliza como destino una dirección de multicast que representa a todos los routers.
c) El flooding no controlado es el caso más simple, aunque puede llevar a problemas cuando hay ciclos. El mecanismo implica que un nodo que recibe un mensaje hace copias idénticas y las reenvía por todos sus enlaces menos por el que lo recibió.

Pregunta 2 (10 puntos)

a) Describa el protocolo de transferencia de datos fiable *rdt 2.1*.
b) Mencione y describa la utilidad de todos los mensajes de control utilizados.
c) ¿Qué agrega el emisor *rdt 3.0* para afrontar la pérdida de paquetes?

Solución:

a) *Ver FSM emisor/receptor del rdt 2.1 vista en el curso.*

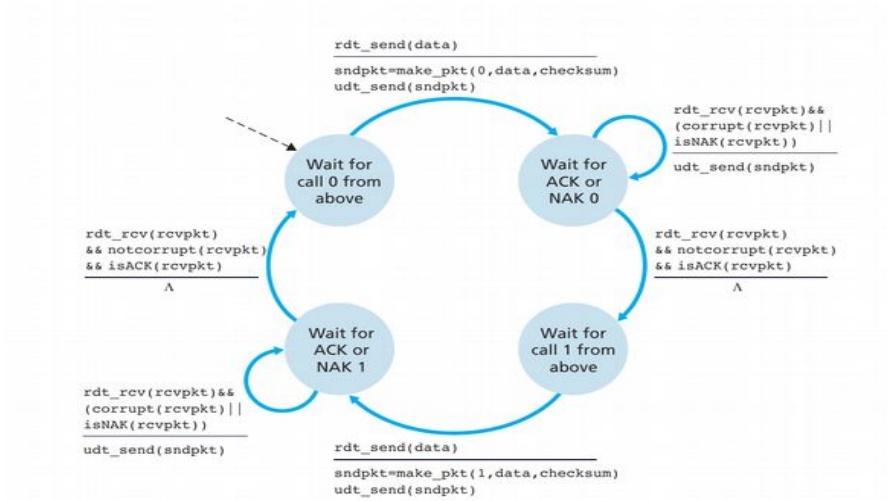


Figure 3.11 ♦ rdt2.1 sender

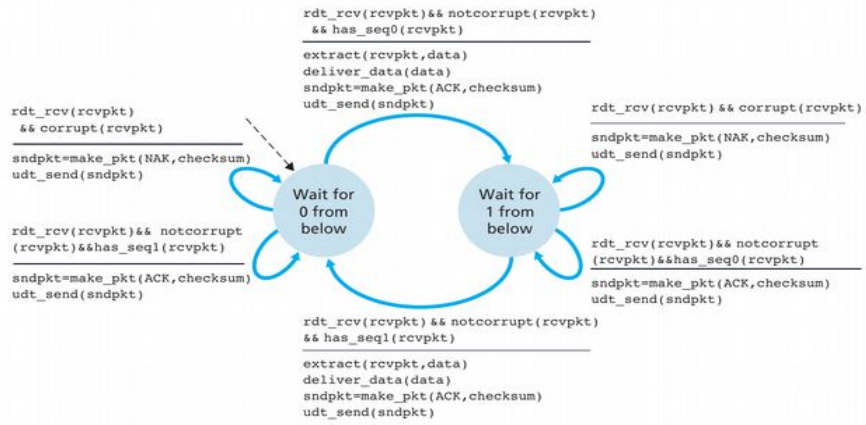


Figure 3.12 ♦ rdt2.1 receiver

b) Se utilizan ACK y NAK. El ACK es para que el receptor pueda confirmar al emisor que recibió correctamente el mensaje con la secuencia esperada, mientras que el NAK indica lo contrario, por ejemplo por haber recibido el paquete corrupto.
c) Lo que agrega es el timer, el cual activa la retransmisión ante la ausencia por un cierto período de tiempo de un ACK confirmando la misma secuencia del último paquete enviado. Este vencimiento del timer dispara la retransmisión automática del último paquete enviado no confirmado.

Pregunta 3 (6 puntos)

Comente brevemente los conceptos de control de congestión y control de flujo en TCP, especificando en particular la existencia o no de mecanismos de señalización explícitos para ambos casos.

Solución:

El control de congestión refiere a una condición o estado de la red como un todo, en contraposición al control de flujo en el que lo que se controla es el flujo de datos entre un emisor y un receptor. El control de flujo es notificado explícitamente mediante el encabezado de TCP (campo "Receive window"), en el que el receptor notifica en todo momento cuantos datos es capaz de recibir antes de que el emisor lo sature. Por otra parte el control de congestión es un mecanismo no explícito, donde el emisor infiere el estado de la red en base a algunos eventos relevantes tales como: la recepción de ACK, la recepción de ACKs duplicados y/o timeouts. Frente a estos eventos el control de congestión modifica la ventana de congestión del emisor, que indica la cantidad de paquetes sin confirmar que pueden estar "en viaje" en la red. Cuanto mayor es el grado de congestión, dicha ventana tiende a valores menores,

entendiendo que en ese caso el estado de la red se recompondrá de manera gradual, y a medida que eso suceda también ir{a en aumento los valores de la ventana de congestión. Se asume que todos los nodos se comportan de esta manera (no hay "free riders").

Pregunta 4 (6 puntos)

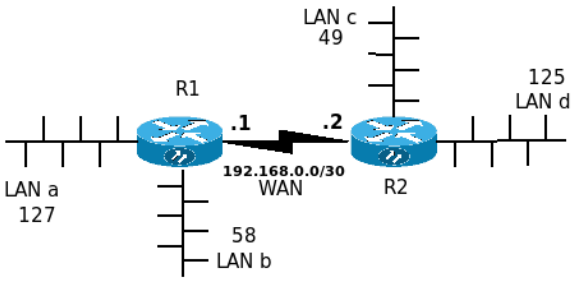
Dado un Sistema Autónomo ASx conectado a otros Sistemas Autónomos utilizando BGP, describa el concepto de "enrutamiendo de papa caliente" (*hot-potato routing*) para routers internos al ASx.

Solución:

BGP es utilizado intra-dominio para propagar a los routers internos la información de alcanzabilidad de prefijos externos al sistema autónomo. Si para un determinado prefijo existen múltiples opciones de "next-hop", el enrutamiento de "papa caliente" eligirá como router de borde aquel que tenga el menor costo según el protocolo de enrutamiento interior (IGP).

Pregunta 5 (10 puntos)

Considere la red de la figura.



Nota 1: el número junto a cada red LAN indica la cantidad de hosts que requieren la asignación de una dirección IP
Nota 2: el enlace WAN ya se encuentra numerado, según figura

a) Proponga un plan de numeración para las 4 redes LAN. Se dispone de los prefijos 4.0.4.0/24, 4.0.5.0/24 y 4.0.6.0/24,
b) Explique los conceptos de enrutamiento (*routing*), y reenvío (*forwarding*). Construya un ejemplo de estos conceptos, considerando el tráfico entre un *host* en LAN a y otro en LAN d.

Solución:

a) Para implementar un plan de numeración en las redes LAN de la figura, consideremos el requerimiento de cantidad de direcciones IP:
LAN a: 127 hosts + 1 router + 2 (red y broadcast) -> total: 129
LAN b: 58 hosts + 1 router + 2 (red y broadcast) -> total: 61
LAN c: 49 hosts + 1 router + 2 (red y broadcast) -> total: 52
LAN d: 125+ hosts + 1 router + 2 (red y broadcast) -> total: 128

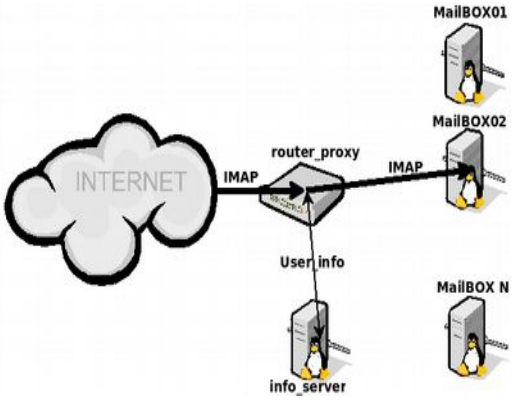
Por lo tanto, para cumplir con dichos requerimientos, los mínimos prefijos necesarios en cada caso serán:
LAN a: /24- LAN b: /26- LAN c: /26- LAN d: /25

Se dispone de los siguientes prefijos: 4.0.4.0/24, 4.0.5.0/24 y 4.0.6.0/24
Del lado de R1, se requiere un prefijo /24 y un prefijo /26. Necesariamente implica la asignación de dos prefijos /24.
Del lado de R2, se requiere un prefijo /26 y un prefijo /25. Con un prefijo /24 es suficiente para ambas redes LAN.
La asignación más adecuada es aquella que signifique la menor cantidad de entradas en las tablas de routing de cada router.
La decisión está entonces en cómo seleccionar los dos prefijos /24 a utilizar en las redes LAN a y b. Si se seleccionan los prefijos 4.0.4.0/24 y 4.0.5.0/24 es posible resumirlos en uno solo: 4.0.4.0/23.
Por lo tanto, la asignación a cada red LAN definida es:
LAN a 4.0.4.0/24
LAN b 4.0.5.0/26
LAN c 4.0.6.128/26
LAN d 4.0.6.0/25
Donde en cada LAN, por ejemplo, se puede asignar la dirección más alta (permitida) del prefijo, a la interfaz de red correspondiente del router, o sea:
4.0.4.254, 4.0.5.62, 4.0.6.254 y 4.0.6.126 respectivamente.

b)
La capa de red realiza dos tareas fundamentales: enrutamiento y reenvío de paquetes. El concepto de enrutamiento tiene sentido a nivel global de la red, donde mediante el intercambio de información entre los *routers* a través de algún protocolo de enrutamiento, es posible determinar al menos un camino entre cualquier par de *hosts* origen y destino conectados a dicha red. La selección de caminos que realizan los algoritmos de enrutamiento se utiliza para determinar las tabla de renvío (*forwarding*).
El concepto de reenvío tiene significado local en cada router e implica que, a partir de la información contenida en la tabla de *forwarding* se determina, para cada paquete que ingresa por una interfaz, cuál es la interfaz de salida y cuál es el siguiente salto para dicho paquete, utilizando por ejemplo el algoritmo de *longest prefix matching*.

Si por ejemplo tenemos un paquete IP con dirección origen 4.0.4.1 (asignada a un host A en LAN a) y dirección destino 4.0.6.1 (asignada a un host D en LAN d),
- cuando el paquete alcance el router R1, su tabla de *forwarding* indicará que el mismo debe salir por la interfaz WAN (enlace punto a punto) y con destino a la interfaz con dirección IP 192.168.1.2.
- cuando el router R2 reciba el paquete, a partir de su tabla de *forwarding* determinará que la interfaz de salida a utilizar es aquella donde está conectada la red LAN d, y que deberá ser enviado al destino 4.0.6.1.

Problema 1 (30 puntos)



Se desea implementar un servicio proxy para IMAP, como se muestra en la figura, que reciba las conexiones TCP solicitadas desde internet, y comunique con el servidor interno (*MailBOX*) que contiene los correos recibidos para éste.

Para conocer a que servidor debe ser redirigido el usuario, se debe enviar un datagrama UDP, al puerto 5555 del equipo *info_server*, con el siguiente texto:

<id_solicitud> <usuario>

y se devuelve:

- **<id_solicitud> OK <MailBOX>** , en caso de éxito
- **<id_solicitud> ERROR**, en caso que el usuario no exista

donde *id_solicitud* es un identificador aleatorio para reconocer la consulta realizada, *usuario* es usuario que ha realizado el login (IMAP) y *MailBOX* es la dirección IP del servidor encargado de atender la sesión. La respuesta se realiza al puerto origen de la misma. En caso de no recibir respuesta del equipo *info_server* en 5 segundos, se cerrará la conexión con el cliente.

El proxy es ejecutado en el servidor *router_proxy.*

A continuación se presenta una sesión del protocolo IMAP, utilizado para la consulta de correos recibidos para un usuario por un sistema de correo electrónico. El mismo atiende en el puerto TCP/143, e intercambia mensajes (*login*, *select*, *fetch* y *logout*, entre otros), identificados al inicio con una etiqueta (ej, `a001`), que permite identificar la respuesta devuelta por el servidor:

```
C: <open connection>
S:   * OK Bienvenido IMAP4rev1 Fing \n
C:   a001 login usuario@fing.edu.uy password \n
S:   a001 OK LOGIN completed \n
C:   a002 select inbox \n
S:   * 18 EXISTS \n
….....
S:   a002 OK [READ-WRITE] SELECT completed \n
C:   a003 fetch 12 full \n
S:   * 12 FETCH (FLAGS (\Seen) INTERNALDATE "17-Jul-1996 02:44:25 -0700"
      RFC822.SIZE 4286 ENVELOPE ("Wed, 17 Jul 1996 02:23:25 -0700 (PDT)"
….....
C:   a006 logout \n
S:   * BYE IMAP4rev1 server terminating connection \n
S:   a006 OK LOGOUT completed \n
```

Se pide:
Implemente en un lenguaje de alto nivel, utilizando las primitivas de la API de *sockets*, el programa que ejecuta el *router_proxy.* Su solución debe cumplir que el mensaje de bienvenida (mostrado en el ejemplo) no llegue más de una vez al cliente.

Solución:

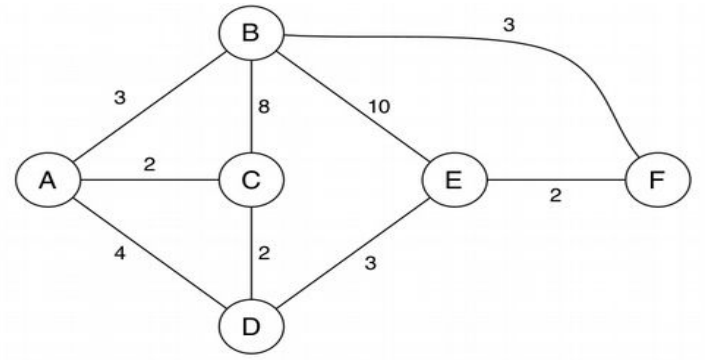
```
void proxy(){
    master = socket.tcp();
    master.bind (*, 143);
    serverSock = master.listen();
    while (true){
        clientSock, err = serverSock.accept(); //espero conexiones
        if (err == 'timeout')
          break;
        thread.new (atenderCliente, clientSock); //nuevo hilo para el cliente
    }
    serverSock.close();
}
```

```
void atenderCliente (clientSock){

    //envio el primer mensaje del protocolo
     clientSock.send("* OK Bienvenido IMAP4rev1 Fing \n")
     data = “”
     repeat
         datarec, err = clientSock.receive(); //Recibo
         data = data + dararec;
     until ((err == 'closed') || (find(data, '\n')) //recibí el login completo

     if (data){

         id_solicitud, usuario = extraer_datos_solicitud(data);
         datagram = id_solicitud +' '+usuario; //armo la solicitud
         consultaSock = socket.udp();
         consultaSock.sendto(datagram, info_server, 5555);
         respuesta, ip_o, port = consultaSock.receive(5); //timeout en 5 segundos
         close(consultaSock); //cierro el socket de la consulta
         if (ip_o==info_server){
             id, resultado, ip = particionar_respuesta(respuesta)
             if (resultado == 'OK'){
                 master = socket.tcp();
                 proxyclientSock = master.connect(ip,143);
                 skip = “”;
                 repeat
                   //Recibo y descarto la bienvenida
                   databienv, err = clientSock.receive();
                   skip = skip + databienv;
                 until ((err == 'closed') || (find(skip, '\n'))
                 if (skip){
                     //hilos para enviar datos y
                     //esperar respuesta de los servidores y del cliente
                     thread.new (pushData, clientSock, proxyclientSock, data);
                     thread.new (pushData, proxyclientSock, clientSock, "");
                 }
             }
         }

     }
     clientSock.close(); //cierro el socket cliente
}


void pushData(Socket src, Socket dst, String initial){
     String data = initial;
     while (true){
         remain, err = dst.send(data) //bloqueante
         if (err == 'closed')
             break;
         data, err = src.receive();
         if (data, err == nil, 'closed')
             break;

     }
     src.close();
     dst.close();
}
```

Problema 2 (30 puntos)

Considere la red de la Figura.

a) Muestre detalladamente en una tabla todos los pasos de la ejecución del algoritmo de Dijkstra para calcular los caminos más cortos desde E al resto de los nodos.
b) Dibuje la estructura que queda definida una vez que se encontraron estos caminos, ¿qué tipo de estructura es?
c) Muestre la tabla de forwarding de E.
d) ¿Qué cambia si se corta el enlace E-B?



Solución:

a)

Iter 1	E
D(E)	0
D(A)	9999
D(B)	(10,E)
D(C)	9999
D(D)	(3,E)
D(F)	(2,E)
N'	{E}
min	F

Iter 2	E
D(E)	0
D(A)	9999
D(B)	(5,F)
D(C)	9999
D(D)	(3,E)
D(F)	(2,E)
N'	{E,F}
min	D

Iter 3	E
D(E)	0
D(A)	(7,D)
D(B)	(5,F)
D(C)	(5,D)
D(D)	(3,E)
D(F)	(2,E)
N'	{E,F,D}
min	B

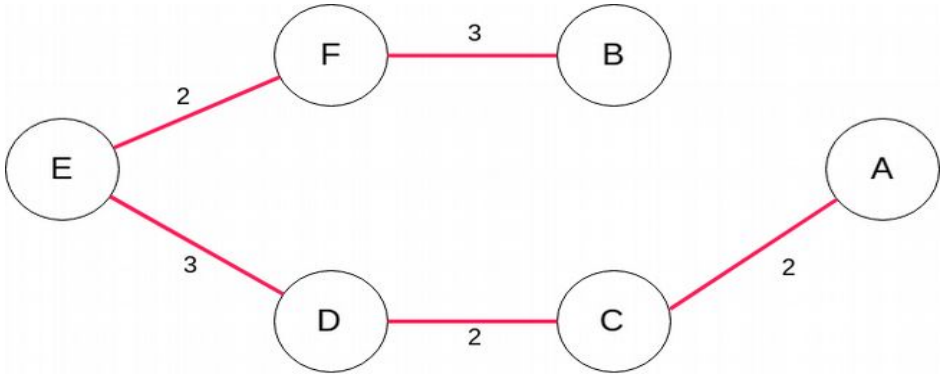
Iter 4	E
D(E)	0
D(A)	(7,D)
D(B)	(5,F)
D(C)	(5,D)
D(D)	(3,E)
D(F)	(2,E)
N'	{E,F,D.B}
min	C

Iter 5	E
D(E)	0
D(A)	(7,D)
D(B)	(5,F)
D(C)	(5,D)
D(D)	(3,E)
D(F)	(2,E)
N'	{E,F,D,B,C}
min	A

Iter 5	E
D(E)	0
D(A)	(7,D)
D(B)	(5,F)
D(C)	(5,D)
D(D)	(3,E)
D(F)	(2,E)
N'	{E,F,D,B,C, A}

N' : Conjunto de los recorridos
min: vértice fijado en la iteración

b) Es un árbol de cubrimiento desde E:



c)

Destino	Link salida	Costo
A	D	7
B	F	5
C	D	5
D	D	3
E	E	0
F	F	2

d) Respecto al nodo E no cambia nada, este link no es parte del cubrimiento mínimo. E llega a B por F y no reenvía nada por B.