Examen – 1° de agosto de 2014

(ref: solredes20140801.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.**
- **Sólo 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.**
- **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 (10 puntos)

Mencione las características fundamentales de una red de conmutación de paquetes y de una red conmutación de circuitos. Proponga un ejemplo de cada tipo de red, y analice las diferencias fundamentales en cuanto a la asignación de recursos para cada tipo de red.

Solución

En una red de conmutación de circuitos, para iniciar un flujo de información entre dos nodos se debe cumplir una fase de establecimiento del circuito, que consiste en determinar un camino de extremo a extremo que permanecerá fijo durante la duración de la transferencia. En cambio, en las redes de conmutación de paquetes el flujo de información entre dos nodos se divide en unidades de largo variable (paquetes) que son encaminados en la red uno a uno, y donde cada nodo atravesado determina el próximo nodo al cual será enviado el paquete (enrutamiento *hop-by-hop*). Un ejemplo típico de red de conmutación de circuitos es la red telefónica, mientras que un ejemplo típico de red de conmutación de paquetes es internet (o más genéricamente, las redes TCP/IP). En conmutación de circuitos se asignan recursos para cada circuito y son de uso exclusivo mientras esté establecido (aún cuando momentáneamente no haya flujo de información); en cambio, en conmutación de paquetes, los recursos se asignan mediante multiplexado estadístico, es decir que cada paquete utiliza recursos (memoria en los routers, ancho de banda en los enlaces) solo cuando lo necesita, y de esta forma cuando no se utilizan, los recursos quedan liberados para otros paquetes.

Pregunta 2 (8 puntos)

Describa el algoritmo de Estado del Enlace (*Link State*) mediante un pseudocódigo. Especifique claramente el paso de inicialización y las iteraciones necesarias para encontrar los caminos más cortos desde un nodo **u** a todos los nodos de una red con **N** nodos.

Solución

```
1 Initialization:
2  N' = {u}
3  for all nodes v
4   if v adjacent to u
5   then D(v) = c(u,v)
6  else D(v) = ∞
7
8 Loop
9    find w not in N' such that D(w) is a minimum
10  add w to N'
11  update D(v) for all v adjacent to w and not in N' :
12   D(v) = min( D(v), D(w) + c(w,v) )
13  /* new cost to v is either old cost to v or known
14  shortest path cost to w plus cost from w to v */
```

```
15 until all nodes in N'
```

Pregunta 3 (6 puntos)

a) Dada la siguiente tabla de *forwarding* de un *router*, ¿por qué interfaz será enviado un paquete con dirección IP destino `172.31.0.126`? Justifique realizando la operación que ejecuta el *router* para tomar dicha decisión.

Prefijo	**Interfaz**
172.28.0.0/22	if0
172.30.0.0/23	if1
172.31.0.128/25	if2
0.0.0.0/0	if3

b) ¿Cuál es el nombre del algoritmo utilizado por el *router* para seleccionar una entrada en la tabla de *forwarding*?

Solución

a) 172.31.0.126 en notación binaria es 10101100.00011111.00000000.01111110

172.28.0.0/22 incluye las redes 172.28.0.0, 172.28.1.0, 172.28.2.0 y 172.28.3.0
172.30.0.0/23 incluye las redes 172.30.0.0 y 172.30.1.0

Analizamos cada entrada de la tabla de forwarding

Para la primera entrada

```
       10101100.00011111.00000000.01111110     (172.31.0.126 en notación binaria)
       AND bit a bit
       11111111.11111111.111111 00.00000000    máscara 255.255.252.0
       ----------------------------------
*1     10101100.00011111.000000 00.00000000    es en decimal 172.31.0.0
*2     10101100.00011100.000000                prefijo 172.28.0.0/22

       *1 y *2 no hay match
```

Para la segunda entrada

```
       10101100.00011111.00000000.01111110     (172.31.0.126 en notación binaria)
       AND  bit a bit
       11111111.11111111.1111111 0.00000000    máscara 255.255.254.0
       ----------------------------------
*3     10101100.00011111.0000000 0.00000000    es 172.31.0.0 en notación decimal
*4     10101100.00011110.0000000               prefijo 172.30.0.0/23

       *3 y *4 no hay match
```

Para la tercera entrada

```
       10101100.00011111.00000000.01111110     (172.31.0.126 en notación binaria)
       AND  bit a bit
       11111111.11111111.11111111.1 0000000    máscara 255.255.255.128
       ----------------------------------
*5     10101100.00011111.00000000.0 0000000    es en decimal 172.31.0.0
*6     10101100.00011111.00000000.1            es en decimal 172.31.0.128

       *5 y *6 no hay match
```

Por lo tanto la interfaz de salida es la "if3" (ruta por defecto). De no existir esta entrada en la tabla de forwarding del router, el paquete sería descartado.

b) Se llama "regla de coincidencia con el prefijo más largo" o *longest prefix match*.

Pregunta 4 (7 puntos)

Describa las características fundamentales del protocolo FTP (*File Transfer Protocol*).

Solución

Protocolo de la capa de aplicación utlizado para la transferencia de archivos desde/hacia un host remoto.
Trabaja de acuerdo al modelo cliente/servidor.
El cliente es el lado que inicia la transferencia.
Un servidor ftp escucha por defecto en el puerto 21 (conexión de control)
El puerto 20 del lado del servidor se destina para establecer la conexión de datos.
El protocolo de transporte en ambas conexiones es TCP.
El cliente inicia la conexión de control hacia el servidor, puerto 21.
El servidor antes de iniciar la transferencia inicia la conexión hacia el cliente, desde el puerto 20.

Pregunta 5 (9 puntos)

a) ¿Cuál es la utilidad del comando `traceroute`?

b) ¿En base a qué protocolos se implementa habitualmente?

c) Describa el funcionamiento del `traceroute` (secuencia de mensajes y sus principales características) en una de las implementaciones mencionadas en la parte anterior.

Solución

a) Permite determinar el camino de ida (a ser recorrido por los paquetes en un momento dado) entre un origen y un destino.

b) ICMP y UDP

c) Se describe la implementación basada en ICMP.

Al ejecutar el comando (por defecto el único argumento necesario es la dirección IP destino), se envían, por defecto, 3 paquetes IP CON TTL=1 conteniendo 3 mensajes ICMP de control del tipo "echo". Al llegar al primer salto y se decrementado el TTL, este pasa a valer 0 y si no es el destino de los paquetes, se devuelven al origen 3 paquetes IP con dirección IP origen quien los genera, dirección IP destino la dirección IP origen de los paquetes que los motivó y conteniendo cada uno un mensaje de error "time exceeded". El receptor de los mensajes refleja en una primera línea en consola la llegada de estos tres mensajes así como el tiempo que transcurrió entre que salió cada paquete y se recibió el mensaje de error correspondiente.

El siguente paso es generar los mismos paquetes que en el paso anterior pero ahora con TTL=2. Ellos atravesarán el primer salto y salvo que sean destinados al segundo salto, el comportamiento será similar el descripto en el párrafo anterior.

Cuando los paquetes arriban al destino final, en lugar de retornar un mensaje ICMP de error "time exceeded" como en todos los saltos anteriores, se generan mensajes ICMP de control "echo reply".

Siempre todos los mensajes que salen desde el host donde se ejecuta el comando lo hacen con la misma dirección IP destino.

La dirección IP destino de los paquetes IP que contienen las respuestas ICMP se obtienen de la dirección IP origen del paquete que lo motivó.

La dirección IP origen es la del nodo que genera el mensaje ICMP (payload del paquete IP)

El payload de los mensajes ICMP de error es una parte del paquete IP que lo originó (encabezado y algunos bits de su payload)

El traceroute basado en UDP se implementa a partir de enviar mensajes UDP dirigidos a puertos donde la probabilidad de respuesta es muy baja. En todos los nodos intermedios el camino se va construyendo a partir de mensajes "time exceeded". Al llegar al destino, se recibirá un mensaje ICMP de error "port unreachable".

Problemas Prácticos

Problema 1 (30 puntos)

Como se muestra en la Figura 1, se dispone de un servicio web que atiende en el puerto 80 de una única IP pública (88.88.88.88), accesible desde Internet. El mismo se atiende con un conjunto de 4 servidores iguales configurados con IP privadas, no ruteables en Internet y cuyo *default gateway* es la IP privada del equipo balanceador (10.0.0.1).

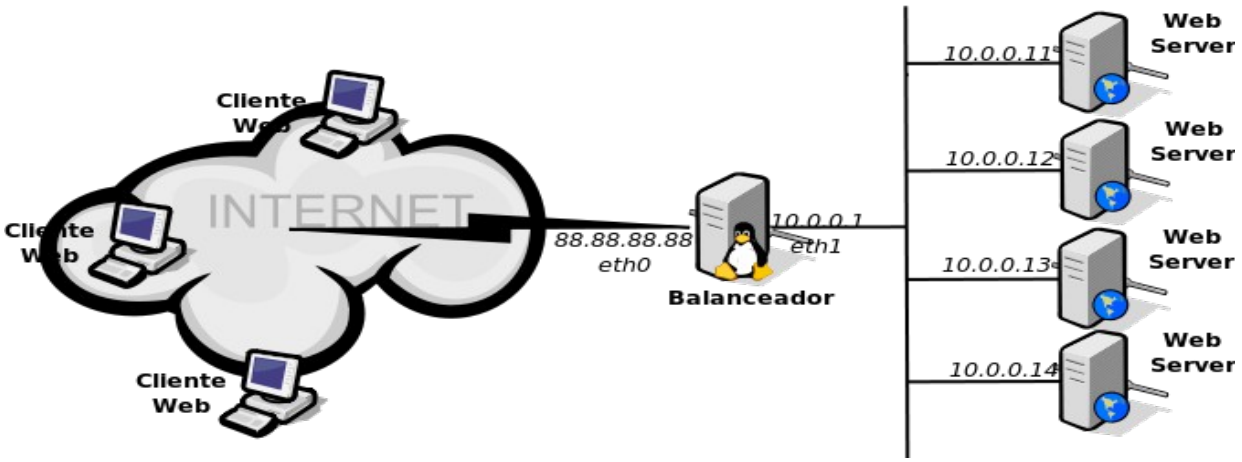

Figura 1

Se busca balancear las solicitudes de servicio, utilizando una metodología similar a NAT (*Network Address Translation*). Para esto a cada nueva conexión TCP se le asigna el servidor que se encuentre con menor cantidad de conexiones activas, que son las que se encuentran en la etapa de transferencia de datos. Además, se deberá controlar que un cliente no transmita datos por más *CNX_TIMEOUT* segundos (no contando el tiempo necesario para establecer la conexión). De superarse este tiempo la conexión se dará por terminada, dejando de reenviar datagramas al servidor asignado. Todo tráfico a otros puertos debe ser ignorado.

Para la solución se puede disponer y modificar la información disponible en el cabezal de capa 3 y 4 de los datagramas. Se cuenta dado un datagrama *d*, las siguientes funciones:

- *datagramaTCP(d): boolean*
  Devuelve TRUE si el datagrama contiene un segmento TCP en sus datos.
- *getTCPflags(d): byte*
  Devuelve un byte con el siguiente formato (0,0,URG,ACK,PSH,RST,SYN,FIN), donde cada posición contiene las diferentes flags del segmento TCP contenido en *d*.
- *getDatosFlujoTCP(d): (ipOrigen,puertoOrigen,ipDestino,puertoDestino)*
  Devuelve registro con los elementos que identifican el flujo TCP contenido en el datagrama *d*.
- *setDatosFlujoTCP(d,ipOrigen,puertoOrigen,ipDestino,puertoDestino): datagrama*
  Modifica los elementos que identifican el flujo TCP contenido en el datagrama, modificando además el *checksum* TCP al nuevo valor pasado por parámetro.
- *sendDatagrama(d,i)*
  Envía el datagrama *d* por la interfaz *i*.
- *readDatagrama(i): datagrama*
  Devuelve datagrama recibido en la interfaz *i*.

Se pide: implemente, en un lenguaje de alto nivel, los procesos que atienden las interfaces hacia Internet (*eth0*), hacia la red interna (*eth1*), y aquellos que sean necesarios para resolver el problema planteado, suponiendo que éstos procesos pueden acceder a estructuras de datos compartidas.

Solución:

```
connections: Table[iporigen, puertoorigen] -> ipserver
connection_time: Table[iporigen, puertoorigen] -> t

function listen_eth0()
  while true do
    d = readDatagrama('eth0')
    if datagramaTCP(d) then
      ipOrigen,puertoOrigen,ipDestino,puertoDestino = getDatosFlujoTCP(d)
      flags = getTCPflags(d)
      if puertoDestino==80 then
        ipserver = connections[ipOrigen,puertoOrigen]
        // es un SYN de algo que no tengop en la tabla
        if (flags && SYN) and !ipserver then
          ipserver = elegir_servidor() //busca el servidor c/menos ocurrencias en connections
          connections[ipOrigen, puertoOrigen] = ipserver
          setDatosFlujoTCP(d,ipOrigen,puertoOrigen,ipserver,80)
          sendDatagrama(d, 'eth1')
        else
          time = connection_time[ipOrigen,puertoOrigen]
          //empiezo a contar el tiempo en el primer ACK que veo pasar
          if (flags && ACK) and !time then
            connection_time[ipOrigen,puertoOrigen] = get_time()
          end
          setDatosFlujoTCP(d,ipOrigen,puertoOrigen,ipserver, puertoserver)
          sendDatagrama(d, 'eth1')
        end
      end
    end
  end
end

function listen_eth1()
  while true do
    d = readDatagrama('eth0')
    if datagramaTCP(d) then
      ipOrigen,puertoOrigen,ipDestino,puertoDestino = getDatosFlujoTCP(d)
      setDatosFlujoTCP(d,address_of('eth0'),puertoOrigen,ipDestino,puertoDestino)
      sendDatagrama(d, 'eth0')
    end
  end
end

function connection_janitor()
  while true do
    sleep(1)
    foe each iporigen, puertoorigen, t in connection_time do
      if get_time() - t > CNX_TIMEOUT then
            //eliminar entrada para conexion fallida
            connections[ipOrigen, puertoOrigen] = NULL
            connection_time[ipOrigen, puertoOrigen] = NULL
      end
    end
  end
end

run_thread(listen_eth0)
run_thread(listen_eth1)
run_thread(connection_janitor)
```

Problema 2 (30 puntos)

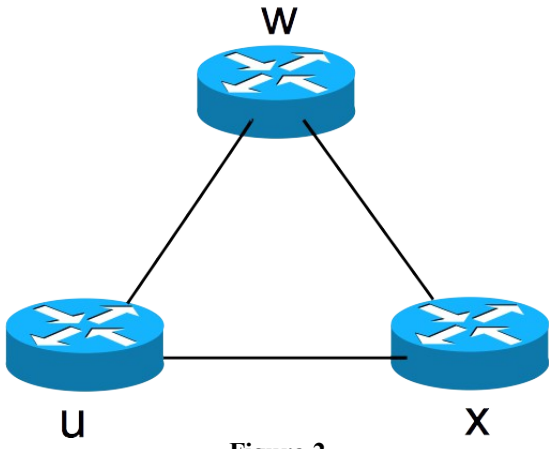

Figura 2

La red de la Figura 2 utiliza el algoritmo de Vector de Distancias (VD), tal como se describe en el curso, para calcular el camino que seguirán los mensajes que se envíen a través de ella. En este caso, los costos asociados a cada enlace son dinámicos y representan la cantidad de bits por segundo que atraviesan el enlace. Las tablas de *forwarding* de cada nodo se actualizan cada vez que el algoritmo VD se estabiliza (en este caso, cuando todos los nodos tienen una visión uniforme de la red).

Suponga que la red es inicializada con los costos de todos los enlaces en 1 (uno) y que luego se lanza el algoritmo de VD. Luego de estabilizado, un flujo constante de 3bits/s es enviado de **u** a **w**.

a) Escriba las tablas de vectores de distancias de cada uno de los *routers* de la red desde que es inicializada hasta que se estabilice y que comience el flujo de datos.
b) Escriba las tablas de vectores de distancias de cada uno de los *routers* de la red desde ese momento y por, al menos, 7 iteraciones más.
c) ¿Qué fenómeno observa?¿Es la forma elegida de asignar costos una buena idea?¿Por qué?

Solución:

a)

en u	u	w	x
Du	0	1	1
Dw	∞	∞	∞
Dx	∞	∞	∞

en w	u	w	x
Du	∞	∞	∞
Dw	1	0	1
Dx	∞	∞	∞

en x	u	w	x
Du	∞	∞	∞
Dw	∞	∞	∞
Dx	1	1	0

en u	u	w	x
Du	0	1	1
Dw	1	0	1
Dx	1	1	0

en w	u	w	x
Du	0	1	1
Dw	1	0	1
Dx	1	1	0

en x	u	w	x
Du	0	1	1
Dw	1	0	1
Dx	1	1	0

en u	u	w	x
Du	0	1	1
Dw	1	0	1
Dx	1	1	0

en w	u	w	x
Du	0	1	1
Dw	1	0	1
Dx	1	1	0

en x	u	w	x
Du	0	1	1
Dw	1	0	1
Dx	1	1	0

Los vectores en cada nodo no cambian por lo que el algoritmo se estabilizó.

b)

Cuando comienza el flujo de datos este toma el camino de costo mínimo entre u y w.
Este camino es por el enlace directo entre u y w.
Por lo tanto, los costos en la red quedan (no son los vectores de distancias):

	u	w	x
u	0	3	0
w	3	0	0
x	0	0	0

Al cambiar los costos de los enlaces cada nodo recalcula sus vectores distancia. Al cambiar los vectores distancia, estos son enviados a los vecinos y se generan las siguientes iteraciones:

en u	u	w	x
Du	0	1	0
Dw	1	0	1
Dx	1	1	0

en w	u	w	x
Du	0	1	1
Dw	1	0	0
Dx	1	1	0

en x	u	w	x
Du	0	1	1
Dw	1	0	1
Dx	0	0	0

en u	u	w	x
Du	0	0	0
Dw	1	0	0
Dx	0	0	0

en w	u	w	x
Du	0	1	0
Dw	0	0	0
Dx	0	0	0

en x	u	w	x
Du	0	1	0
Dw	1	0	0
Dx	0	0	0

en u	u	w	x
Du	0	0	0
Dw	0	0	0
Dx	0	0	0

en w	u	w	x
Du	0	0	0
Dw	0	0	0
Dx	0	0	0

en x	u	w	x
Du	0	0	0
Dw	0	0	0
Dx	0	0	0

Se estabiliza y las tablas de forwarding son actualizadas (por letra solo se actualizan luego de estabilizado el algoritmo). Ahora el camino de costo mínimo para ir de u a w es a través de x.
Por lo tanto el flujo cambia de ruta y los costos quedan:

	u	w	x
u	0	0	3
w	0	0	3
x	3	3	0

Los nodos ven un cambio en los costos de sus enlaces y actualizan su vector distancia.
En la primer iteración solo cambia el vector distancia de x, por lo tanto este nodo envía el vector a sus vecinos.

en u	u	w	x
Du	0	0	0
Dw	0	0	0
Dx	0	0	0

en w	u	w	x
Du	0	0	0
Dw	0	0	0
Dx	0	0	0

en x	u	w	x
Du	0	0	0
Dw	0	0	0
Dx	3	3	0

en u	u	w	x
Du	0	0	0
Dw	0	0	0
Dx	3	3	0

en w	u	w	x
Du	0	0	0
Dw	0	0	0
Dx	3	3	0

en x	u	w	x
Du	0	0	0
Dw	0	0	0
Dx	3	3	0

en u	u	w	x
Du	0	0	0
Dw	0	0	0
Dx	3	3	0

en w	u	w	x
Du	0	0	0
Dw	0	0	0
Dx	3	3	0

en x	u	w	x
Du	0	0	0
Dw	0	0	0
Dx	3	3	0

Se estabiliza y nuevamente el camino de costo mínimo entre u y w es por el enlace u-w.

Por lo tanto la tabla de forwarding cambia, cambia la ruta del flujo y los costos quedan:

	u	w	x
u	0	3	0
w	3	0	0
x	0	0	0

c) El fenómeno que se observa es que los costos de los enlaces, las tablas de vectores distancias y, por lo tanto, las tablas de *forwarding* de los nodos oscilan continuamente. Esto se debe a que los costos de los enlaces dependen de la cantidad de tráfico que los atraviesan, la cantidad de tráfico de depende de las tablas de *forwarding* y éstas de los costos de los enlaces, generándose una circulo de interdependencias que genera la anomalía observada. Se concluye que utilizar parámetros como tráfico o congestión sin otro tipo de medidas que eviten las oscilaciones, no es una buena idea para asignar costos a los enlaces en un algoritmo de ruteo. (ver Seccion 4.5 del Kurose)