Redes de Computadoras

Solución Examen – 13 de agosto de 2020

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 (10 puntos)

Considere una aplicación que transmite datos a una velocidad constante. Además, cuando esta aplicación se inicia, se ejecutará durante un periodo de tiempo relativamente largo.

a) ¿Qué sería más apropiado para esta aplicación, una red de conmutación de circuitos o una red de conmutación de paquetes? ¿Por qué?

b) Suponga que se utiliza una red de conmutación de paquetes y el único tráfico que existe en la misma procede de la aplicación que acabamos de describir. Además, suponga que la suma de las velocidades de datos de la aplicación es menor que las capacidades de cada uno de los enlaces. ¿Será necesario algún mecanismo de control de congestión? ¿Por qué?

Solución Pregunta 1

a) La red de conmutación de circuitos tiene como desventaja el costo inicial para establecer el circuito, pero dado que la hipótesis es que la aplicación se ejecutará por un período largo de tiempo, ese costo fijo inicial se diluye y tiene poco peso respecto al peso que tiene en el global. Dado que la aplicación transmite a una velocidad constante, con conmutación de circuitos podemos reservar los recursos necesarios para ese tráfico y además no estaremos desperdiciando esos recursos, ya que van a estar siempre usados.

b) Dado que los datos viajan a velocidad constante, que la suma de todos no logra llenar la capacidad de ninguno de los enlaces de la red, y que no existe en la red ningún otro tráfico que no sea el de la aplicación bajo estudio, podemos decir que ningún router debería presentar problemas para manejar el enrutamiento del mismo y por ende no debería esperarse una situación de congestión en la red.

Pregunta 2 (10 puntos)

Considere el protocolo DHCP.

a) Describa su utilidad.

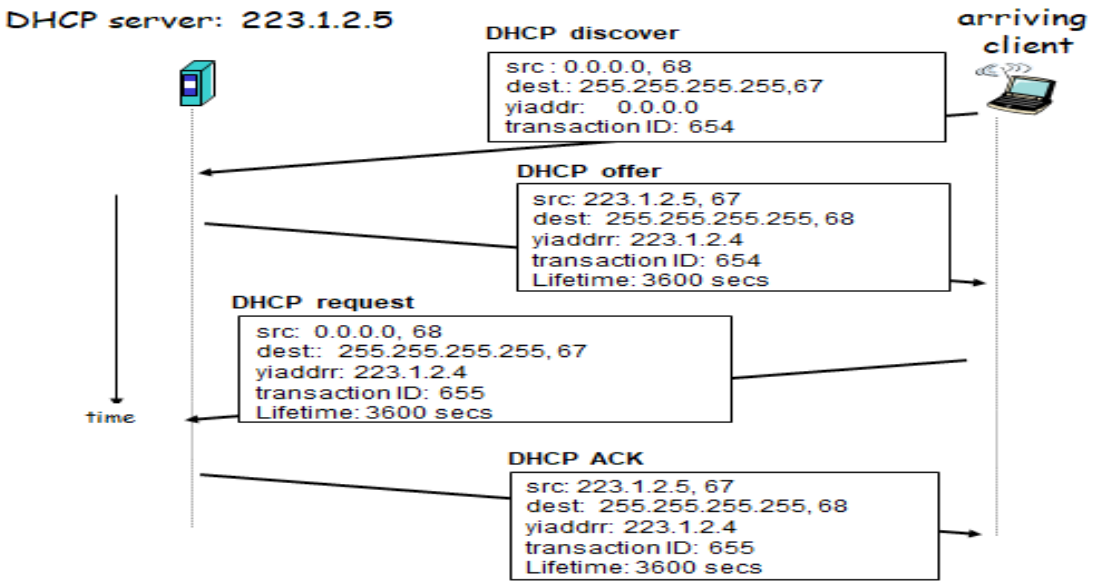
b) Describa en detalle los mensajes que se intercambian en el protocolo y entre que entidades. Indique el protocolo de transporte que utiliza.

Solución Pregunta 2

a) El protocolo DHCP, Dinamic Host Configuration Protocol, es una de las formas disponibles para que un nodo conectado a una red IP obtenga una dirección IPv4 desde un servidor DHCP, el que gestiona un pool de direcciones IP a tales efectos. Con este protocolo logramos mayor flexibilidad en el uso del rango de direcciones IP disponibles para numerar los nodos de una red, pudiendo reutilizarlas. DHCP brinda de forma automática un conjunto de parámetros de red, para el correcto funcionamiento de los dispositivos. Mínimamente uno espera que los dispositivos obtengan una dirección y máscara de red, pero pueden obtener otros parámetros como ser default gateway, rutas estáticas, servidores DNS entre otros.

b) Utiliza el protocolo UDP. Los mensajes intercambiados son los siguientes:

- El equipo cliente solicita la configuración enviando un mensaje de difusión de tipo “DHCP DISCOVER”.
- Cada servidor DHCP responde con un mensaje de tipo “DHCP OFFER” que contiene una dirección IP y otras opciones de configuración. Este mensaje se envía al cliente por unicast si es posible o por broadcast en otro caso (RFC 1531 - http://tools.ietf.org/html/rfc1531).
- El cliente acepta los parámetros ofrecidos por alguno de los servidores respondiendo con un mensaje de difusión de tipo “DHCP REQUEST” indicando en él el identificador de servidor que ha elegido. El mensaje DHCPREQUEST debe ser enviado a todos los servidores que recibieron el DHCPDISCOVER para que puedan reutilizar la dirección que habían ofrecido. También se usa DHCPREQUEST para extender en el tiempo la validez de una dirección IP.
- El servidor elegido envía un mensaje de confirmación de tipo “DHCP ACK” indicando que aprueba esa concesión y confirmando la configuración, así como indicando el tiempo máximo en que la dirección IP es válida.



Pregunta 3 (10 puntos)

Considere un protocolo de transferencia de datos fiable que sólo utiliza paquetes de reconocimiento negativo (NAK).

a) Imagine que el emisor envía datos con muy poca frecuencia. ¿Sería preferible un protocolo que solo emplea paquetes NAK a uno que utilice paquetes ACK? ¿Por qué?

b) Suponga ahora que el emisor tiene muchos datos que transmitir y que la conexión terminal a terminal experimenta muy pocas pérdidas. En este segundo caso, ¿sería preferible un protocolo que solo emplee paquetes NAK a otro que utilice paquetes ACK? ¿Por qué?

Solución Pregunta 3

Se plantea una solución posible, dependiendo de la justificación dada hay otras posibles respuestas correctas.
Se asume un protocolo que envía paquetes NAK ante casos de recibir un paquete corrupto o recibir un número de secuencia no esperado.

a) En este caso, sería preferible un protocolo que utiliza paquetes ACK. En un protocolo que utiliza NAK exclusivamente, la pérdida de un paquete x es detectado al recibir el paquete x+1, es decir, recibió el x-1 y luego el x+1, momento en el que detecta que no recibió el x y entiende que se perdió. Al recibir el x+1 enviará el NAK y luego recibirá el x, luego de al menos 4 RTT de la transmisión original. Dado que hay una demora grande entre un paquete de datos y el siguiente (por letra) entonces pasará un tiempo muy largo hasta que la situación vuelva a la normalidad y se recupere el paquete perdido. En un protocolo con ACK, el propio emisor puede reenviar automáticamente ante la finalización de los timers, incluso sin recibir nada del receptor.

b) En este caso sería preferible un protocolo que utiliza solo NAKs. Dado que en este caso los datos se envían a mayor frecuencia, la recuperación en un protocolo basado en NAKs ocurriría mas rápido. Mas aún, como los errores son poco frecuentes, se generarán muy pocos NAKs a diferencia de un protocolo basado en ACKs que por cada paquete generaría un ACK. Con lo cual podríamos decir que, en este caso, hay un consumo menor de ancho de banda por menos consumo en la mensajería de reconocimiento (ACK y NAK).

Pregunta 4 (10 puntos)

Considere el algoritmo de enrutamiento de vector-distancia.

a) Explique el fenómeno de conteo a infinito.

b) Ejemplifique el fenómeno anterior en una red de cuatro nodos.

Solución Pregunta 4

a) El problema del conteo infinito se genera por el propio mecanismo utilizado por los protocolos vector-distancia, ya que si bien los nodos reciben los vectores-distancias de sus vecinos, que ellos luego utilizan para recalcular el propio, no tienen conocimiento de si alguno de esos costos corresponden a caminos que pasan por ellos y por ende es posible que se generen bucles de enrutamiento. Cuando uno tiene visión global de la red es un problema sencillo de detectar y evitar, pero en los algoritmos vector-distancia solo se maneja información parcial.

b)

Este ejemplo presenta el problema del conteo a infinito. A partir de una solución estable con los costos iniciales unitarios, cuando el costo del enlace c-d cambia a un valor K>>1,los nodos c y d van a comenzar una iteración que arrastrará al resto de los nodos.

- Al inicio c informará del cambio y luego b le enviará su vector, en el que tiene un camino de costo 2 para llegar a b que evidentemente es mejor de lo que c tiene (por el cambio de costo) pero lo que no sabe es que ese camino ya lo incluye.
- En este paso de la iteración, c va a tener costo 3 hacia d (via b! Que a su vez era via c!).
- En la iteración siguiente b va a tener costo 4 hacia d (via a!), mejor camino que la información que recibe de c.
- Este proceso va a continuar hasta que c finalmente llegue al costo K hacia d, y con K grande, el proceso puede ser arbitrariamente largo.

Problemas Prácticos

Problema 1 (30 puntos)

Considere tres redes LAN, cada una con un *switch* (S1, S2 y S3), que están interconectadas mediante dos *routers* (R1 y R2), como se muestra en la Figura 1.

Se pide:

a) Asigne prefijos a cada subred sabiendo que la Subred 1 puede contener hasta 30 hosts, la Subred 2 hasta 23 hosts y la Subred 3 hasta 125 hosts.

b) Asigne direcciones IP y direcciones MAC a todas las interfaces cuando corresponda. Las direcciones MAC deben respetar el formato de 48 bits visto en el curso.

c) Suponiendo que se ejecuta algún algoritmo de *routing* interno, escriba la tabla de *forwarding* del *router* R2.

d) Considere el envío de un datagrama IP conteniendo un segmento UDP, desde el host E al host B. Suponga que todas las tablas ARP están actualizadas, salvo la del router R1, que se encuentra vacía. Describa todas las tramas intercambiadas, en la secuencia en que ocurren, completando para ello una tabla con el siguiente formato:

Nro. trama	Dir. MAC origen	Dir. MAC destino	Dir. IP origen	Dir. IP destino	Tipo y Contenido	Subred

De existir mensajes ARP, se debe indicar su contenido fundamental y no simplemente su objetivo.

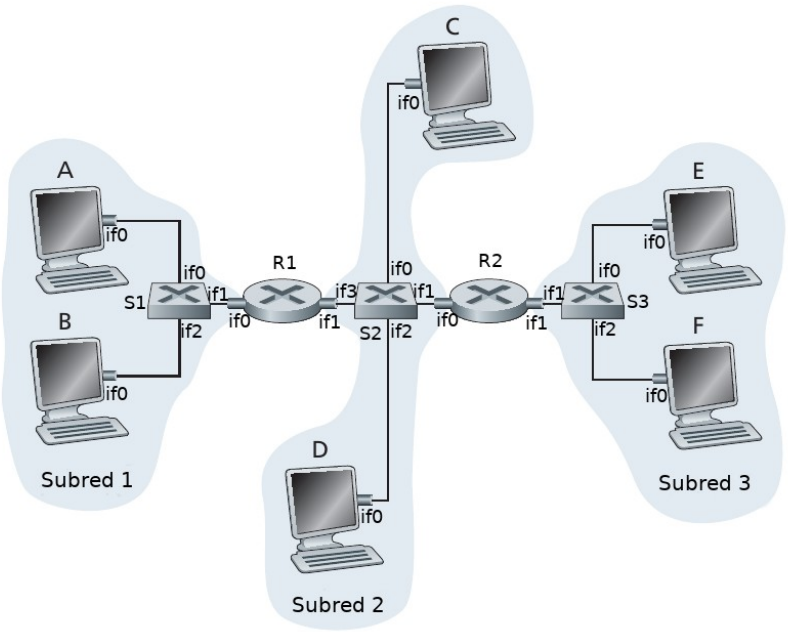


Figura 1

Solución Problema 1

a)

Subred 1 Requiere (30+1) direcciones IP → requiere un prefijo /26 → 172.16.1.0/26

Subred 2 Requiere (23+2) direcciones IP → requiere un prefijo /27 → 172.16.2.0/27

Subred 3 Requiere (125+1) direcciones IP → requiere un prefijo /25 → 172.16.3.0/25

Nota 1: se define el uso de direcciones IP privadas, en este caso, del rango 172.16.0.0/12
Nota 2: por no explicitarse en la letra del problema y de acuerdo a la "Nota 1", se define un plan de numeración donde el tercer octeto de cada subred coincide con el número de la misma, para facilitar la comprensión. De todas formas, si se quisiera, las 3 subredes podrían pertenecer al mismo /24.

b)

Subred 1
Host A – if0 172.16.1.2 255.255.255.192 00:4F:49:11:AB:1A
Host B – if0 172.16.1.3 255.255.255.192 00:4F:49:11:AB:1B
Router 1 – if0 172.16.1.1 255.255.255.192 00:00:0C:11:AB:10

Subred 2
Host C – if0 172.16.2.2 255.255.255.224 00:4F:49:22:CD:2C
Host D – if0 172.16.2.3 255.255.255.224 00:4F:49:22:CD:2D
Router 1 – if1 172.16.2.1 255.255.255.224 00:00:0C:22:CD:11
Router 2 – if0 172.16.2.4 255.255.255.224 00:00:0C:22:CD:20

Subred 3
Host E – if0 172.16.3.2 255.255.255.128 00:4F:49:33:EF:3E
Host F – if0 172.16.3.3 255.255.255.128 00:4F:49:33:EF:3F

Router 2 – if1 *172.16.3.1 255.255.255.128* *00:00:0C:33:EF:21*

c)

Tabla de forwarding de R2

Prefijo	Siguiente salto	Interfaz
172.16.1.0/26	172.16.2.1	if0
172.16.2.0/27	Directamente Conectada	if0
172.16.3.0/25	Directamente Conectada	if1

d)

Nro	***MAC origen***	***MAC destino***	***IP origen***	***IP destino***	***Tipo y Contenido***	***Subred***
1	*00:4F:49:33:EF:3E*	*00:00:0C:33:EF:21*	*172.16.3.2*	*172.16.1.3*	*Segmento UDP*	*3*
2	*00:00:0C:22:CD:20*	*00:00:0C:22:CD:11*	*172.16.3.2*	*172.16.1.3*	*Segmento UDP*	*2*
3	*00:00:0C:11:AB:10*	*FF:FF:FF:FF:FF:FF*	*- -*	*- -*	*Tipo: ARP request* *Contenido coloquial: "¿dir MAC de 172.16.1.3?"* *Contenido fundamental:* *IP origen: 172.16.1.1* *MAC origen: 00:00:0C:11:AB:10* *IP destino: 172.16.1.3* *MAC destino: 00:00:00:00:00:00*	*1*
4	*00:4F:49:11:AB:1B*	*00:00:0C:11:AB:10*	*- -*	*- -*	*Tipo: ARP response* *Contenido coloquial: "la dir MAC de 172.16.1.3 es 00:4F:49:11:AB:1B"* *Contenido fundamental:* *IP origen: 172.16.1.3* *MAC origen: 00:4F:49:11:AB:1B* *IP destino: 172.16.1.1* *MAC destino: 00:00:0C:11:AB:10*	*1*
5	*00:00:0C:11:AB:10*	*00:4F:49:11:AB:1B*	*172.16.3.2*	*172.16.1.3*	*Segmento UDP*	*1*

Problema 2 (30 puntos)

Sea la red mostrada en la Figura 2, donde un equipo *analizador* analiza los *logs* recibidos por los equipos *servidores web* y en caso de encontrar actividad sospechosa genera una alerta. Esta alerta debe ser recibida por **todos** los *administradores* de la red (Admin1, …, Admin N). Cada *servidor web* envía las entradas de *log*, a medida que se producen, al *analizador*. Esto se hace mediante una conexión TCP al puerto 2562 que permanece abierta mientras el servidor web está activo.
Los *administradores* escuchan las alertas por UDP en el puerto 8523.

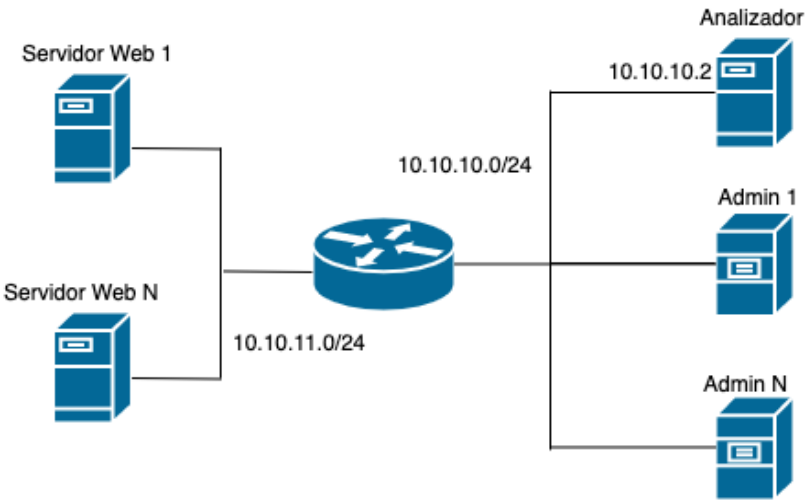


Figura 2

El archivo de *log* sigue el formato de los Apache *access log*, un ejemplo del contenido de este archivo son las siguientes entradas (líneas):

```
177.38.32.248 - - [04/May/2019:00:00:02 -0300] "GET /centos/repomd.xml HTTP/1.1" 200 3445\n
2620:113:80c0:8::20 - - [04/May/2019:00:00:02 -0300] "GET /opensuse/ HTTP/1.1" 200 1190\n
2001:41c9:1:3ce::1:10 - - [04/May/2019:00:00:02 -0300] "GET /raspbian/ HTTP/1.1" 200 1660\n
```

Se pide:

a) Implemente en un lenguaje de alto nivel, utilizando las primitivas de la API de *sockets*, el programa que ejecuta el *analizador*.
Deberá ser capaz de atender múltiples conexiones de los servidores simultáneamente.
Para analizar las entradas de *log* cuenta con la función `analizar_solicitud(solicitud_http):boolean` que recibe como entrada una solicitud HTTP (línea del *log*) y retorna `true` en caso de que sea sospechosa.
Además, se cuenta con la función `crear_alerta(solicitud_http):string` que crea la carga útil de una alerta para dicha solicitud.
Puede asumir implementadas las funciones de manejo de *strings* que considere necesarias, pero debe explicar su funcionamiento.

b) Implemente en un lenguaje de alto nivel, utilizando las primitivas de la API de *sockets*, el procedimiento `enviar_log(archivo_log)` que utilizan los *servidores web* para enviar las entradas de *log* al *analizador*.
Para esto dispone de la llamada bloqueante `read_line(file):string`, que consume y devuelve una línea completa de un archivo.

Solución Problema 2

```
void analizador () {
  master = socket.tcp();
  master.bind (*, 2562);
  serverSock = master.listen();
  while (true){
     clientSock, err = serverSock.accept(); //espero conexiones
     if (err == 'timeout')
       break;

     //nuevo hilo para la conexion de un servidor que enviará su log
     thread.new (atenderServidor, clientSock);
  }
  serverSock.close();
}

void atenderServidor(clientSock) {
  buff = ""
  // recibo datos hasta que se cierre la conexión
  repeat
    fragment, err = clientSock.receive();
    buff = buff + fragment

    // busco primer \n en datos recibidos
    pos = string.pos(buff, "\n")
    if (pos > 0) {
      // si hay \n entonces obtengo la linea
      line = string.sub(buff, 1, pos-1)    // Recorto la línea
      buff = string.sub(buff, pos+1, size(buff))  // Guardo el resto

      // analizo la linea
      if (analizar_solicitud(line)) {
        // envio alerta a administradores
        data = crear_alerta(line)
        sockAlerta = socket.udp()
        sockAlerta.sendto(data, 10.10.10.255, 8523)
        sockAlerta.close()
      }
    }
  until (err == 'closed')
  clientSock.close(); //cierro el socket cliente
}

void enviar_log (log) {
  master = socket.tcp();
  clientSock, err = master.connect(10.10.10.2, 2562)
  while (true){
    line = read_line(log)
    repeat
      remain, err = clientSock.send(line)
    until (remain == "")
  }
```

```
  clientSock.close ()
}
```