Redes de Computadoras

Solución de Examen – 29 de julio 2016

(ref: solredes20160729.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. 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 tipo de dispositivos son los responsables de definir el camino por el que es enrutado un paquete IPv4 a través de Internet, cuál es la tabla y algoritmo que utiliza para tomar la decisión y en qué parte del cabezal del datagrama IPv4 se basa dicha decisión?

b) Dado un intercambio de paquetes entre los *hosts Origen* y *Destino*, identifique al menos dos causas que justifiquen el hecho de que un paquete que envía *Destino* hacia *Origen* pueda no recorrer necesariamente el mismo camino *Origen-Destino* pero en orden inverso.

c) Describa la relación entre la MTU del enlace y la fragmentación IPv4, mencionando los campos del encabezado IPv4 definidos para este fin.

Solución

a) Los dispositivos para enrutar el tráfico IPv4 son los enrutadores (routers) de capa 3, quienes en base al valor de la dirección destino del encabezado IP origen toman la decisión de reenviar el paquete actual hacia el siguiente *hop* según lo indique el resultado de aplicar el algoritmo *Longest Prefix Match* sobre su *tabla de forwarding*.

b) Dado que la red IPv4 es una red de datagramas esto implica que las decisiones son distribuidas y descentralizadas; además, a diferencia de una red de conmutación de circuitos, no implica necesariamente que el camino de ida y vuelta sean los mismos. Como la decisión de enrutamiento es tomada de forma distribuida y por el carácter dinámico de la red, es posible que las condiciones de la misma cambien con el tiempo. De esta forma pueden existir condiciones como congestión, pérdida de paquetes en enlaces o incluso la caída de uno. Cualquiera de estos factores puede influir sobre el protocolo de ruteo dinámico (o incluso también cuando el ruteo sea estático) y por consiguiente generar diferente información en cada uno de los nodos de la red.
Incluso los prefijos de las redes pueden estar publicados de forma de forzar caminos de ida y vuelta distintos (en este caso el motivo es la política del proveedor de servicio, no un fallo en la red).

c) La MTU del enlace limita el tamaño que puede tener un paquete IPv4 incluyendo los encabezados. Cuando se tienen datos a transmitir y el tamaño de los datos + 20 bytes (cabezal IPv4) es superior a la MTU, entonces se deberá hacer algo o de lo contrario se descartará el paquete, eventualmente enviando un mensaje ICMP de error. Al fragmentar un paquete se deben configurar correctamente (al menos) 4 campos del encabezado, que son: largo, DF, MF y Fragment Offset. Con la flag DF se indica si ese paquete se puede o no fragmentar, con MF se indica si el fragmento actual es o no el último de la secuencia y el Offset indica la posición que deberá ocupar en el paquete original el actual, contado en unidades de 8 bytes.

Pregunta 2 (8 puntos)

a) Suponga que en una red se tiene un *hub* y a éste le ingresa una trama mal formada por un puerto, ¿qué hace el hub con ella?

b) Explique por qué se afirma que un *switch* al ser encendido se comporta como un *hub* y luego evoluciona al comportamiento típico de un *switch*.

c) ¿Es posible tener colisiones en una red implementada sólo con *hubs*? ¿Qué sucede en una red switcheada pura?

Solución

a) Dado que un *hub* es un dispositivo de capa física y no maneja el concepto de *frame* sino que replica bit a bit todo el tráfico que ingresa a sus interfaces, cuando algún dispositivo conectado a él le envía tráfico mal formado, el mismo será replicado.

b) Al encender un *switch* el mismo tiene su *tabla de switching* vacía. Al llegarle una trama por una de sus interfaces el *switch* deberá hacer un *flooding controlado* y reenviarlo por todas las interfaces, dado que la dirección MAC Destino presente en el encabezado de la trama no se encuentra en su tabla. Al mismo tiempo, el *switch* ahora sabe que al dispositivo que tiene la dirección MAC que está en el encabezado MAC Origen de la trama lo puede alcanzar a través del enlace por el que llegó esa trama. De esta forma el *switch* comienza a completar la *tabla de switching* con ese dato, agregando una fila que contiene: Dirección MAC (la que viene en el MAC Origen de la trama), Interfaz (por la que llegó la trama) y TTL (tiempo durante el cual es válida la entrada). Este mecanismo es lo que llamamos *self-learning*. Esto es fundamental para que el *switch* pueda ser *plug&play* y no sea necesario realizarle ninguna configuración adicional. A su vez, dado que los dispositivos pueden cambiar de lugar o que la red es dinámica y puede cambiar también, el campo TTL mencionado antes es importante, y permite a su vez que este tipo de cambios también sea soportado por los *switches* sin la necesidad de configuración manual.

c) Sí, porque son dispositivos de capa física y lo único que hacemos al conectar muchos *hubs* es agrandar el mismo dominio de colisión. Al utilizar *switches*, y mediante su mecanismo de *store & forward*, los dominios de colisión se limitan ahora a uno por cada enlace y si estos enlaces son *FullDúplex* entonces podemos decir que no habrá colisiones (si fueran *HalfDúplex* podría colisionar cuando dos estaciones quieren transmitir al mismo tiempo).

Pregunta 3 (6 puntos)

a) Describa brevemente las tecnologías FDM (*Frecuency Division Multiplexing*) y TDM (*Time Division Multiplexing*).

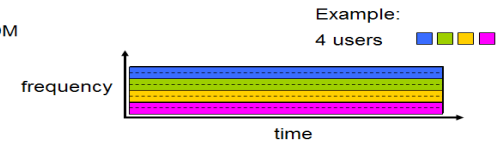
b) Mencione las principales diferencias entre los medios físicos fibra óptica y UTP.

Solución

a)

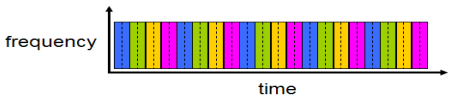
En Multiplexación por División de la Frecuencia (FDM), el espectro de frecuencias disponible en un enlace se reparte todo el tiempo entre las conexiones establecida en el mismo.

FDM



En Multiplexación por División en el Tiempo (TDM), el tiempo se divide en ranuras de duración fija preestablecida (“marcos”) y éstas en otras ranuras (con una cantidad preestablecida), también de duración constante (”particiones”), y cada una de ellas es utilizada periódicamente por una conexión, con todo el espectro de frecuencias disponible en ese lapso de tiempo para ella.

TDM



b)

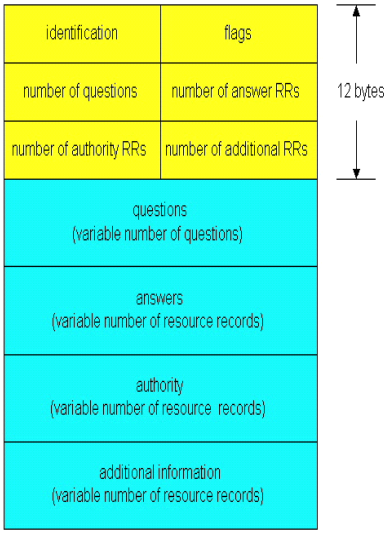
	Fibra óptica	**UTP**
Material utilizado para llevar la información	hilo de vidrio o plásticos	hilo/s de cobre trenzados
Principio de funcionamiento	la información se transmite como pulsos de luz	la información se transmite como pulsos eléctricos
Inmunidad a interferencias	si	no
Atenuación de la señal	menor	mayor
Dirección de la comunicación	unidireccional	bidireccional
Velocidades de operación (según distancias)	mayor (“decenas de Gbps”)	menor (“Gbps”)
Distancias de operación	mayor (“kilómetros”)	menor (“metros”)
Costo	mayor	menor

Pregunta 4 (8 puntos)

a) Describa el formato de los mensajes de DNS (*Query* y *Reply*).

b) ¿Es posible que un servidor recursivo funcionando correctamente consulte por determinado registro, reciba la respuesta e inmediatamente después repita la misma consulta?

Solución



a) Ambos tipos de mensajes tienen el mismo formato: encabezado de largo fijo (12 bytes) y carga útil de largo variable. En el encabezado viaja la siguiente información:

Identificador de 16 bits, para asociar parejas de mensajes *Query-Reply*.

Indicadores o Flags (señalizando por ejemplo, si es un *Query* o un *Reply*, si se desea realizar una consulta recursiva, si la recursión está disponible, si la respuesta es autoritativa, el tipo de respuesta, entre otras) y, el Número de Consultas, el Número de RRs (*Resources Records*) de Respuesta (respuesta/s a la/s consulta/s realizada/s), el Número de RRs de Autoridad (información proporcionada por servidores autoritativos) y el Número de RRs de (información) Adicional (información que podría ser útil para quien recibirá el *Reply*, a juicio de quien lo generó). Los últimos 3 Números pueden ser 0.

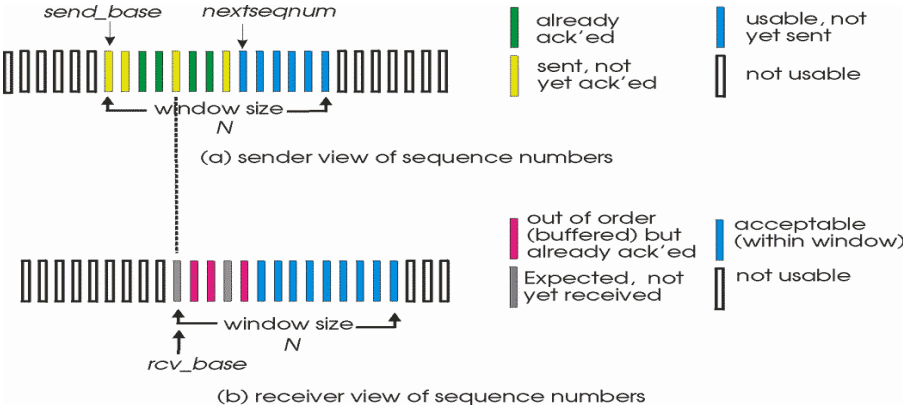
En la carga útil viene la información referenciada en el encabezado (*questions, answers, authority, additional*).

b) Sí. Por ejemplo en el caso que los RRs de la primera respuesta tuvieran 0 como valor de TTL.

Pregunta 5 (10 puntos)

Utilizando dos diagramas de mensajes y sus números de secuencia como los vistos en el curso, uno para el emisor y otro para el receptor, explique el principio de funcionamiento de la técnica denominada Repetición Selectiva.

Solución



sender

data from above :
- if next available seq # in window, send pkt

timeout(n):
- resend pkt n, restart timer

ACK(n) in [sendbase,sendbase+N]:
- mark pkt n as received
- if n smallest unACKed pkt, advance window base to next unACKed seq #

receiver

pkt n in [rcvbase, rcvbase+N-1]
- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

pkt n in [rcvbase-N,rcvbase-1]
- ACK(n)

otherwise:
- ignore

Problema 1 (30 puntos)

Se desea implementar la aplicación `redireccion` que permita implementar la redirección de un puerto en el host que la ejecuta a un puerto en otro host (*port-forwarding*).

La aplicación será ejecutada, a modo de ejemplo, de la siguiente forma:

```
> redireccion 8080:www.debian.org:80
```

En este ejemplo, mientras se ejecuta la aplicación, las conexiones TCP al puerto 8080 del host local serán redireccionadas al puerto 80 del host `www.debian.org`. La implementación debe permitir conexiones simultaneas, y no debe asumir cambios en la aplicación que inicia la conexión.

Asuma que se cuenta con primitivas para el manejo de *sockets* (establecimiento de conexión, envío y recepción, etc.), hilos y funciones de *parsing* de los comandos.

Se pide:

a) Implemente en un lenguaje de alto nivel el comando `redireccion`.
b) ¿Que utilidad encuentra para el comando implementado?

Solución

a)

```
int redireccion(arg){

// genero socket que atiende conexiones
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0){
print("ERROR abriendo socket");
return 1;
}

// almacena en listenPort puerto donde se escuchan solicitudes
// redirectHost el equipo al que debe redireccionarse
// redirectPort el puerto al que se debe redireccionar
(listenPort,redirectHost,redirectPort)= parseoArgumentos(arg);

serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = listenPort;

// Realizo bind
if (bind(sockfd, serv_addr) < 0){
       print("ERROR en binding");
       return 1;
}

// realizo accept
while((client_sock = accept(socketfd){
       print("Connexion aceptada");
       if( pthread_create(connection_handler,(void*)&client_sock) < 0){
              print("No se pudo crear el thread”);
       }
}
return 0;
}
```

```
/*
* thread que maneja conexiones para cada cliente
*/
void *connection_handler(void *socket){

// establezco conexión al servidor destino
proxysocket = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0){
       print("ERROR opening socket");
       return 1;
}

serv_addr.sin_family = AF_INET;
serv_addr.sin_addr = redirectHost;
serv_addr.sin_port = redirectPort;

// realizo conexion
if (connect(proxysocket,serv_addr) < 0){
       print("ERROR en conexion");
       close(proxysocket);
       return 1;
}

// configuro select
FD_ZERO(&readset);
FD_SET(socket, &readset);
FD_SET(proxysocket, &readset);
while (select(&readset) >= 0 ){
       if (FD_ISSET(socket, &readset)) {
              // envio hacia servidor remoto
              recv(socket, buffer);
              send(proxysocket, buffer);
       }
       if (FD_ISSET( proxysocket &readset)) {
              // envio a cliente del servicio
              recv(proxysocket, buffer);
              send(socket, buffer);
       }
}
close(proxysocket);
close(socket);
return 0;
}
```

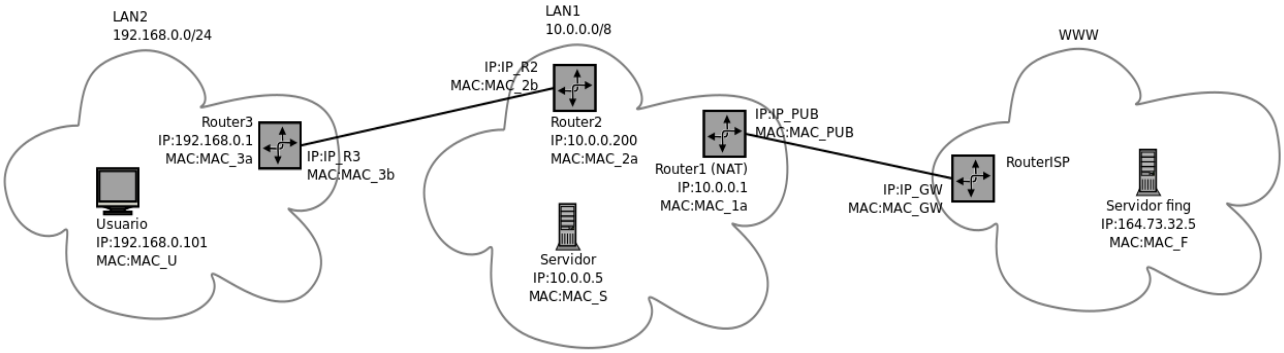
b)

La aplicación de la parte a) se denomina proxy, y permite conectarnos a un equipo de forma indirecta.

El uso más común del servicio proxy es el anonimato dado que el equipo que acepta la conexión, piensa que el cliente se encuentra en la dirección IP del proxy, elemento que funciona únicamente como intermediario.

Un ejemplo de uso es, suponga que existe una limitante para acceder a recursos que indica que solamente pueden ser accedidos desde ciertas direcciones IP's pertenecientes a organizaciones conocidas. Esto genera que por ejemplo desde mi casa no pueda acceder a la información. Por lo tanto instalando la aplicación en un equipo de las organizaciones autorizadas, podré acceder a la información de forma transparente.

Problema 2 (30 puntos)



Una institución posee una red con la topología de la figura. Su red consiste en dos LANs enrutadas (LAN1 y LAN2), y la salida hacia internet está provista por un router NAT en la LAN1. LAN1 además alberga un servidor HTTP. El usuario está conectado en la LAN2. La configuración de enrutamiento es estática.

Se pide:

1) Conectividad desde el usuario:

   a) Asigne direcciones IP reales para IP_R2 e IP_R3.
   b) Especifique la rutas que deberán ser agregadas en los distintos nodos de la red para permitir al Usuario acceder al Servidor y a Internet.
   c) Suponga que el Usuario accede al Servidor fing. Asumiendo que todos los enlaces de implementan con tecnología ethernet, muestre los valores de los campos de dirección de los cabezales TCP/IP y ethernet, en cada segmento entre el puesto de Usuario y el router del ISP.

2) Conectividad desde Internet:

   a) Especifique los cambios de configuración necesarios para permitir acceder al Servidor HTTP en LAN1 desde internet. No es viable renumerar las LANs.
   b) Especifique los cambios de configuración necesarios para permitir acceder por SSH a la máquina del Usuario, desde internet. No es viable renumerar las LANs.

Solución

1)

a)
Utilizo para enlace entre routers 3 y 2 el prefijo 192.168.1.0/30.
IP_R3: 192.168.1.2
IP_R2: 192.168.1.1

b)
Agrego las siguientes entradas en las tablas de forwarding:

Router 1)

Destination	Gateway	Interface
0.0.0.0/0	IP_GW	La de MAC MAC_PUB
10.0.0.0/8	-	La de MAC MAC_1a
192.168.0.0/24	10.0.0.200	La de MAC MAC_1a

Router 2)

Destination	Gateway	Interface
0.0.0.0/0	10.0.0.1	La de MAC MAC_2a
10.0.0.0/8	-	La de MAC MAC_2a
192.168.1.0/30	-	La de MAC MAC_2b
192.168.0.0/24	192.168.1.2	La de MAC MAC_2b

Router 3)

Destination	Gateway	Interface
0.0.0.0/0	192.168.1.1	La de MAC MAC_3b
192.168.1.0/30	-	La de MAC MAC_3b
192.168.0.0/24	-	La de MAC MAC_3a

c)
Se representan únicamente paquetes de datos salientes. Los entrantes son los mismos en cada segmento, con las direcciones origen y destino intercambiadas.

IMPORTANTE: Se asume que Servidor fing es un servidor web. Elijo arbitrariamente puerto 5000 para origen en Usuario.

En LAN2:

	Orig	Dest
TCP	5000	80
IP	192.168.0.101	164.73.32.5
MAC	MAC_U	MAC_3a

En enlace entre Router3 y Router2:

	Orig	Dest
TCP	5000	80
IP	192.168.0.101	164.73.32.5
MAC	MAC_3b	MAC_2b

En LAN1:

	Orig	Dest
TCP	5000	80
IP	192.168.0.101	164.73.32.5
MAC	MAC_2a	MAC_1a

En enlace entre Router1 y RouterISP:

	Orig	Dest
TCP	8888*	80
IP	IP_PUB	164.73.32.5
MAC	MAC_PUB	MAC_GW

* El número de puerto es arbitrario, sustituido por el router NAT durante la traducción de direcciones.

2)

a) Se debe configurar el router NAT para redireccionar paquetes con destino IP_PUB:80 a 10.0.0.5:80.

b) Se debe realizar una configuración análoga a la anterior, pero con el puerto 22 en lugar de 80, y la IP 192.168.0.101 en lugar de 10.0.0.5.