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

Examen – Julio de 2012

(ref: eirc1207.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.**
- **Apague su celular.**
- **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.**
- **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).**
- **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)

(a) ¿Cuál es el objetivo principal de la capa de enlace?
Respuesta: La capa de enclace de datos tiene la responsabilidad de transferir datagramas desde un nodo a otro adyacente, a través de un link.

(b) Determinados protocolos hacen uso de un único canal de *broadast* compartido, lo que lleva a que dos o más transmisiones simultáneas generen colisiones. Detalle el protocolo de acceso múltiple CSMA/CD.
Respuesta: Capitulo 5, slide 12 del material del curso.

Pregunta 2 (10 puntos)

(a) El camino por el que es enrutado un paquete IPv4 a través de Internet, ¿puede ser fijado de antemano? Justifique.
Respuesta: El protocolo IP enruta usando la direccion IP de destino que viene en el paquete (también IP permite hacerlo por la IP de origen), y ese enrutamiento es realizado en cada router que atrviesa el paquete hasta su destino. De esta forma, el protocolo IP toma la decisión de ruteo en cada hop de la red, de manera que no hay forma de decirle al paquete cuál es el camino que debe seguir para llegar al end-system destino (ni tampoco como retornar). Es por esta propiedad que las redes IP son típicamente llamadas redes de conmutacion de paquetes, en contraste a las redes de conmutación de circuitos, en las cuales sí es posible prefijar y reservar un camino y recursos por los cuales será siempre enrutado un flujo para los mismos end-system que son origen y destino de la comunicación.

(b) ¿En qué campo/s del encabezado IP se basa el enrutamiento en IPv4?
Respuesta: El campo utilizado por el protocolo para conocer cuál será el next-hop al cual entregar el paquete es el campo Destination IP (también IP permite hacerlo por la IP de origen) que tiene 32 bits.

(c) Describa la relación entre la MTU del enlace y la fragmentación IP, mencionando los campos del encabezado IP definidos para este fin.
Respuesta: El protocolo IPv4 permite la fragmentación de un paquete, de manera que el paquete pueda ser transmitido por la capa de enlace. Esta fragmentación además, depende de los enlaces por los que atraviesa el paquete hasta el destino, ya que podrían existir distintos tipos de capas de enlaces con diferentes tamaños de MTU. En consecuencia, el protocolo IP divide el paquete en unidades que sean menores a la MTU del segmento a atravesar, indicando en cada uno de ellos que es un paquete fragmentado, cuál es el offset de esos datos dentro del paquete original y por último, indicando si existen o no mas fragmentos a re-ensamblar para volver a armar el paquete original. IPv4 prevee también en el encabezado que el nodo origen pueda decir que el paquete no sea fragmentado en absoluto aún cuando no pueda ser transmitido por el enlace sin ser fragmentado. Cada uno de los campos y su uso está descrito en detalle en el material del curso del capítulo 4 de la capa 3 clase 1 en la slide 12.

Pregunta 3 (8 puntos)

Explique el uso de los siguientes registros de DNS y brinde un ejemplo de cada uno:

(a) NS
Registro utilizado para definir el nombre del servidor de nombres autoritativo de una zona.
Ejemplo: irc.fing.edu.uy IN NS dns.irc.fing.edu.uy

(b) A
Registro utilizado para asociar una dirección IPv4 a un nombre
Ejemplo: dns.irc.fing.edu.uy IN A 164.73.128.6

(c) MX
Registro utilizado para indicar los servidores que están dispuestos a recibir los correos electrónicos destinados a determinado dominio de correo
Ejemplo: irc.fing.edu.uy IN MX 10 mail.irc.fing.edu.uy

(d) AAAA
Registro utilizado para asociar una dirección IPv6 a un nombre
Ejemplo: dns.irc.fing.edu.uy IN AAAA fdda:5cc1:23:4::1f
En los ejemplos brindados no se incluye el TTL

Pregunta 4 (6 puntos)

Describa el principio de funcionamiento de las Redes de Distribución de Contenido (CDN por su sigla en inglés).

Respuesta: Una arquitectura muy genérica de cómo se implementa el servicio asociado a las CDN comprende un proveedor de contenido y nodos y servidores de la propia CDN (en general las CDNs brindan servicio a varios proveedores de contenido). El proveedor de contenido define qué objetos de todo su contenido estarán disponibles mediante la CDN y los carga en ella; primero en algún nodo de distribución, lo que luego se replica a los servidores distribuidos en la red.
El principio de funcionamiento es el siguiente:
Si al proveedor de contenido lo identificamos como `www.pdc.com` y a la CDN como `www.cdn.com`, cuando el cliente, desde su navegador, solicita el contenido, por ejemplo, de la página www.pdc.com`/sports/london2012.html`, en el HTML de la misma encontrará referencias del estilo `http://www.cdn.com/www.pdc.com/sports/london2012/survival-muse.mpg`. Ello hará que el navegador requiera de una consulta DNS para determinar la dirección IP asociada a `www.cdn.com`; el servidor autoritativo correspondiente (el de la CDN) le responderá con la dirección IP del servidor de contenido más "cercano" en la red. En un tercer paso, el navegador solicitará a dicha IP el recurso `http://www.cdn.com/www.pdc.com/sports/london2012/survival-muse.mpg`.

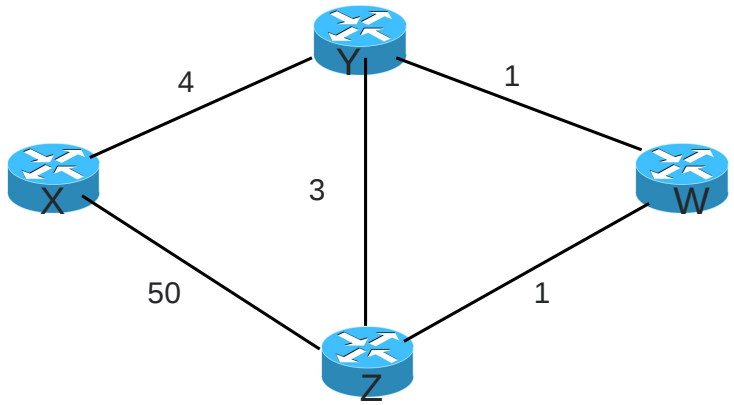
Pregunta 5 (6 puntos)

Suponga que en un futuro no muy lejano, su computadora se encuentra conectada a Internet y está descargando archivos .ogg utilizando algún sistema para compartir archivos entre pares (p.e. Kazaa, BitTorrent, etc.). El cuello de botella en Internet es su enlace de acceso residencial: 512kbps, full-duplex, simétrico. Mientras está descargando los archivos que le interesan, otros diez usuarios empiezan, de repente, a cargar archivos .mp3 desde su computador. Asumiendo que su computador es muy potente y que el trabajo adicional es imperceptible (CPU, disco, etc.) ¿harán las cargas simultáneas que se enlentezcan sus descargas? Fundamente.

Respuesta: Cuando 10 clientes arrancan las transferencias, saturan el ancho de banda de subida de la conexión. Los paquetes de las subidas (cargas) compiten con los ACK de las descargas, y dado que saturan el enlace, algunos se pierden. Esos ACKs que no llegan, fuerzan re-transmisiones por parte del emisor. Esas re-transmisiones son consideradas congestión por el stack en el emisor, y por ende baja la velocidad de transmisión de nuestro flujo, enlenteciendo nuestra descarga.

Problema 1 (30 puntos)

Considere la red de la Figura 1 en la que el costo de enrutar por un determinado enlace es el número que aparece junto a él. Suponga que se utiliza un algoritmo de ruteo de vector-distancia que implementa la técnica de "reversa envenenada" (*poisoned reverse*).



a) Una vez que el algoritmo de ruteo se estabilizó, los *routers* W, Y y Z se informan entre si sus distancias a X. ¿cuáles son esos valores?

Respuesta:

Router Z	informa a W: $D_Z(X)=inf$
Router Z	informa a Y: $D_Z(X)=6$
Router W	informa a Y: $D_w(X)=inf$
Router W	informa a Z: $D_w(X)=5$
Router Y	informa a W: $D_y(X)=4$
Router Y	informa a Z: $D_Y(X)=4$

b) Ahora suponga que el costo del enlace entre X e Y sube a 60. ¿Por qué se producirá el problema del conteo infinito?
c) ¿Cuántas iteraciones del algoritmo de ruteo son necesarias para volver a un estado estable? Justifique.

Respuestas b) y c)

Si,ocurrirá el probelma del conteo infinito.

	t0	t1	t2	t3	t4
Z	informa a W: $D_Z(X)=inf$			informa a W: $D_Z(X)=inf$	
Z	informa a Y: $D_Z(X)=6$			informa a Y: $D_Z(X)=11$	
W	informa a Y: $D_w(X)=inf$		informa a Y: $D_w(X)=inf$		
W	informa a Z: $D_w(X)=5$		informa a Z: $D_w(X)=10$		
Y	informa a W:	informa a W:			informa a W:

	$D_y(X)=4$	$D_y(X)=9$			$D_y(X)=14$
Y	informa a Z: $D_Y(X)=4$	informa a Z: $D_Y(X)=inf$			informa a Z: $D_Y(X)=inf$

$D_Z(X)$ sube de 5 en 5 cada 3 iteraciones empezando en 6 hasta que llega a 51 en la iteracion 27 (3*(51-6)/5), ahi decide que el camino mas corto es el enlace directo a X.

	t27	t28	t29	t30	
Z	informa a W: $D_Z(X)=50$				
Z	informa a Y: $D_Z(X)=50$				
W			informa a Y: $D_w(X)=51$		
W			informa a Z: $D_w(X)=inf$		
Y		informa a W: $D_y(X)=53$		informa a W: $D_y(X)=inf$	
Y		informa a Z: $D_Y(X)=inf$		informa a Z: $D_Y(X)=52$	

Luego de t30 no hay mas actualizaciones.

d) ¿Cómo modificaría el costo del enlace entre Y y Z para que no ocurra el problema del conteo infinito si el costo del enlace entre Y y X sube de 4 a 60?
Respuesta: poner el costo del enlace entre Y y Z en infinito.

Problema 2 (30 puntos)

Para agregar robustez en la asignación de direcciones en una red local se cuenta con varios servidores DHCP funcionando simultáneamente. Todos los servidores asignan direcciones del mismo rango de direcciones DHCP las cuales están almacenadas en una lista local a cada servidor. Para evitar asignar direcciones no disponibles, cada servidor debe registrar las direcciones asignadas por otros servidores y borrarlas de su lista de direcciones.

Se pide:

a) Explique el funcionamiento del protocolo DHCP.

Respuesta: Capítulo 4 capa de red página 4 del material del curso.

b) Diseñe un servidor DHCP que cumpla con las funcionalidades descritas. Explíquelo mediante un diagrama.

Respuesta:

```
creo socket udp para recibir pedidos
creo socket para escuchar ofertas de otros servidores
creo socket para enviar respuestas

escucho en los dos sockets
  si el mensaje es un ack o un offer (cualquiera de las dos esta bien)
    registro la direccion y la borro de mi tabla
  si el mensaje es un discover
    envio un offer de manera broadcast
  si el mensaje es un request
    verifico si es para mi
      verifico si no asigne ya la direccion
        borro la ip de la tabla
```

```
envio un ack de manera broadcast
```

c) Implemente la solución diseñada en un lenguaje de alto nivel.

Para la implementación se cuenta con una estructura (DhcpMessage) ya definida para los mensajes DHCP. Esta estructura cuenta con los campos: dhcpType, yiaddr (Your Ip Address), transactionId, serverId y lifetime. Además se encuentra ya implementada la función auxiliar createDhcp(), la cual crea un mensaje de tipo DHCP y recibe como parámetros los campos de la estructura.

Respuesta:

```
void servidorDHCP(){
  //creo socket udp recibir pedidos
  int socketServidor = socket(AF_INET, SOCK_DGRAM, 0);

  //socket para escuchar ofertas de otros servidores
  int socketEscucha = socket(AF_INET, SOCK_DGRAM, 0);

  //socket para enviar respuestas
  int socketRespuesta = socket(AF_INET, SOCK_DGRAM,0)

  //se debe habiliar broadcast en el socket
  int broadcast = 1;
  setsockopt(socketRespuesta, SOL_SOCKET, SO_BROADCAST, &broadcast, sizeof broadcast)

  //armo la direccion de broadcast
  struct sockaddr_in addrBroadcast;
  socklen_t addrBroadcast_size = sizeof addrBroadcast;
  addrBroadcast.sin_family = AF_INET;
  addrBroadcast.sin_port = htons(68);
  addrBroadcast.sin_addr.s_addr = inet_addr(INADDR_BROADCAST);

  //bind de los sockets que escuchan
  struct sockaddr_in addrServidor;
  socklen_t addrServidor_size = sizeof addrServidor;
  addrServidor.sin_family = AF_INET;
  addrServidor.sin_port = htons(67); //puerto para recibir pedidos
  addrServidor.sin_addr.s_addr = inet_addr(INADDR_ANY);
  bind(socketServidor, (struct sockaddr*)&addrServidor, addrServidor_size);

  struct sockaddr_in addrEscucha;
  socklen_t addrEscucha_size = sizeof addr;
  addrEscucha.sin_family = AF_INET;
  addrEscucha.sin_port = htons(68)
  addrEscucha.sin_addr.s_addr = inet_addr(INADDR_ANY);
  bind(socketEscucha, (struct sockaddr*)&addrEscucha, addrEscucha_size);

  while (true){

    char* mensaje = malloc(MAX_MSG_SIZE);

    //select entre los dos sockets
    fd_set fds;
    FD_ZERO (&fds);
    FD_SET (socketServidor, &fds);
    FD_SET (socketEscucha, &fds);

    select (socketServidor > socketEscucha ? socketServidor + 1 : socketEscucha + 1, &fds,
NULL, NULL, NULL);

    if (FD_ISSET (socketEscucha, &fds)){
      recv(socketEscucha, mensaje, data_size,0);
      //si el mensaje es un ack o un offer (cualquiera de las dos esta bien)
      DhcpMessage mensajeDhcp = createDhcp(mensaje):
      if (mensajeDhcp.dhcpType == DHCPACK){
        //registro la direccion y la borro de mi tabla
        tablaIp.remove(mensajeDhcp.yiaddr);
      }
```

```
    }

    if (FD_ISSET (socketServidor, &fds)){
      recv(socketServidor, mensaje, data_size,0);
      DhcpMessage mensajeDhcp = createDhcp(mensaje):
      //si el mensaje es un discover
      if (mensajeDhcp.dhcpType == DHCPDISCOVER){
        // envio un offer
        char* dhcpOffer = createDhcp(DHCPOFFER, tablaIp.getNext(), mensajeDhcp.transactionId,
IP_SERVIDOR, TIMEOUT)
        int msg_size = sizeof(dhcpOffer);
        //envio la respuesta de manera broadcast
        sendto(socketRespuesta, dhcpOffer, msg_size, 0, (struct sockaddr*)&addrBroadcast,
addrBroadcast_size);
      }
      //si el mensaje es un request
      if (mensajeDhcp.dhcpType == DHCPREQUEST){
        //verifico si es para mi
        if (mensajeDhcp.serverId == IP_SERVIDOR){
          //verifico que no se haya asignado la direccion
          if (tablaIP.exists(mensajeDhcp.yiaddr)){
            //borro la ip de la tabla
            tablaIp.remove(mensajeDhcp.yiaddr);
            //envio un ack
            char* dhcpAck = createDhcp(DHCPACK, tablaIp.getNext(), mensajeDhcp.transactionId,
IP_SERVIDOR, TIMEOUT)
            int msg_size = sizeof(dhcpAck);
            //envio la respuesta de manera broadcast
            sendto(socketRespuesta, dhcpAck, msg_size, 0, (struct sockaddr*)&addrBroadcast,
addrBroadcast_size);
          }
        }
      }
    }
  }
}
```