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

Examen – Marzo de 2012

(ref: eirc1203.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 (6 puntos)

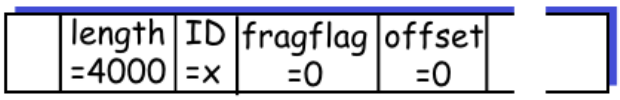
a) Enumere las capas típicas de una arquitectura de red (siguiendo el modelo de 5 capas visto en el curso). Identifique dos protocolos para cada capa a partir de la 2 hacia arriba.
b) Indique qué entidades comunican las capas 2, 3 y 4 respectivamente.

Pregunta 2 (8 puntos)

Considere una computadora conectada a Internet a través de un acceso Ethernet, donde se ejecuta un navegador que realiza un `HTTP GET` a un servidor web con el que ya está conectado. Enumere los pasos necesarios para enviar el método desde origen a destino así como los protocolos involucrados, considerando que expiraron tanto la tabla de ARP como el caché de DNS en el origen; considere únicamente al emisor.

Pregunta 3 (10 puntos)

a) Explique el uso de los campos *Longitud del Datagrama (**length**)*, *Identificador (**ID**)*, *Indicador de fragmentación (**fragflag**)* y *Desplazamiento de Fragmentación (**offset**)* del encabezado IP.
b) Considere un datagrama cuyo encabezado no tiene campos opcionales, permite fragmentación, y tiene los siguientes valores en los campos referidos:



Un *router* debe enviar dicho datagrama por un enlace con **MTU = 1500 bytes**. Indique:
i. Una manera conveniente de fragmentarlo.
ii. Para cada fragmento, los valores de los campos indicados en el datagrama de la figura.

Pregunta 4 (6 puntos)

Suponga que el receptor UDP calcula el *checksum* para un segmento recibido y comprueba que se corresponde con el valor almacenado en el campo *checksum* del cabezal UDP. ¿Puede el receptor estar completamente seguro de que no hay ningún bit erróneo? Justifique.

Pregunta 5 (10 puntos)

a) Explique los conceptos *flooding* y *flooding controlado.*
b) Describa el funcionamiento de Reverse Path Forwarding y que objetivo persigue.
c) Mencione un protocolo de suscripción a grupos de multicast y describa su funcionamiento.

Problema 1 (30 puntos)

Se desea implementar el sistema de *forwarding* de un router. La tabla de *forwarding* de un router es una lista simple llamada `routeList` con elementos tipo `routeStep`. Para realizar el *forwarding* deberán implementarse dos funciones:

- `routeStep findNextHop(ip_addr address);`
- `void forwardPacket(ip_pack packet);`

La primera devuelve el *nexthop* hacia el destino dado según la tabla de *forwarding*. En caso de no encontrarlo, devuelve null.
La segunda, utiliza la primer función para determinar el *nexthop* y realiza el envío del paquete. En caso de no existir un *nexthop* hacia el destino dado, debe devolverle al emisor un paquete de error (ver operaciones auxiliares). Asuma que siempre existe una ruta al emisor de un paquete.

Se cuenta con las estructuras:

```
routeStep{
        ip_addr net;
        ip_addr netmask; // Forma 111...0000
        interface if;
        ip_addr next_hop; // Si es null, esta
        conectado directamente
}
```

```
ip_pack{
        ip_addr dest;
        ip_addr from;
        int size;
        int ttl;
        byte[] data;
}
```

`ip_addr` es un arreglo de 32 bits. `next_hop` pertenece a la subred de la interfaz `if`.

Para el envío de datos, solamente se cuenta con la siguiente operación:

- `void sendTo(ip_pack packet, ip_addr destInNetwork, interface if);`

Esta operación envía el paquete `packet` al nodo `destInNetwork` vía la interfaz `if`. `destInNetwork` debe estar en la subred de la interfaz `if`.

Se cuenta con las siguientes operaciones auxiliares:

- `ip_addr andBitABit(ip_addr, ip_addr);`
- `ip_addr orBitABit(ip_addr, ip_addr);`
- `ip_addr not(ip_addr);`
- `int numberOfBits(ip_addr); // Da la longitud de una máscara, asumiendo que es de la forma 11...110...00. Ej: numberOfBits(11100...000) = 3.`
- `ip_pack createNoRouteError(ip_addr dest);`

Se pide:

1. Implementar la función `findNextHop`.
2. Implmentar la función `forwardPacket`.
3. ¿Porqué es necesario especificar el parámetro `destInNetwork` en `sendTo`?

Problema 2 (30 puntos)

Suponga que tenemos dos entidades de red, A (receptor) y B (emisor) y un canal bidireccional entre ellas. B tiene que enviar hacia A un conjunto de mensajes de datos cumpliendo que:

- Cuando A recibe una solicitud de la capa superior para obtener el siguiente mensaje de datos (D) de B, A tiene que enviar un mensaje de solicitud (R) a B. Sólo cuando B recibe una mensaje R puede devolver un mensaje D a A.
- A tiene que entregar exactamente una copia de cada mensaje D a la capa superior.
- Los mensajes R se pueden perder (pero no corromper) en el canal.
- Los mensajes D, una vez enviados, siempre son correctamente entregados.
- El retardo del canal es desconocido y variable.

Proporcione las máquinas de estados del protocolo descripto anteriormente.

Considere que, ademas de las primitivas habituales de los protocolos RDT, cuenta con las siguientes:

- `rqst_data_to_app(datos)` – utilizada por la capa de transporte para solicitarle datos a la capa de aplicación. (asuma que siempre habrá datos para ser enviados)
- `app_rqst_data()` - utilizada por la capa de aplicación para solicitarle datos a la capa de transporte.

Si necesita crear nuevas primitivas, describa su funcionamiento.