Examen de Introducción a las Redes de Computadoras y Comunicación de Datos (ref: eirc0407.doc) 23 de julio de 2004

***Atención**: para todos los ejercicios, suponga 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).*

Ejercicio 1

Se desea implementar sobre un nodo que interconecta una red con su proveedor de servicio, un procedimiento que permita controlar que las unidades de transferencia (celdas) enviadas desde la red, estén separadas por lo menos *i seg.*, (donde entonces está a autorizado a enviar *1/i* celdas por seg.). Para casos de necesidad, el usuario podrá enviar celdas hasta *l* seg. antes de que se active la autorización para la próxima celda. El algoritmo del procedimiento GCRA - Generic Cell Rate Algorithm, es el siguiente (se ejecuta para cada celda y tiene una variable global que controla TAT Theoretical Arrival Time)

```
GCRA specification for increment i and limit l :
When a cell arrives at time t :
if first cell then TAT := t                          // initialize TAT
if t >= TAT then                                     // late cell
       TAT := t
else                                                 // early cell
       if t < TAT - l then                           // too early
              return NON CONFORMING
TAT := TAT + i
return CONFORMING
```

Para las celdas que se retorna CONFORMING son enviadas, el resto podrán descartarse.

Se pide:

a) Se cuenta con un sistema que tiene dos estructuras de encolamiento de celdas, para posterior transmisión con política FIFO de los mismos. Se desea implementar que las celdas que llegan en tiempo correcto tengan prioridad ante posibles congestiones, y en consecuencia las que llegaron previamente al TAT, posean más posibilidad de descarte. Modificar el procedimiento (GCRA) para poder implementarlo.

b) El enlace por donde se transmite la información es de 128 Kbps, y se desea controlar el retardo las celdas cuyo tamaño es de 53 bytes, tal que sea como máximo 50 mseg. En caso de no poder cumplirse deberá descartarse la celda, y en caso de tener canal ocioso se enviarán las celdas correspondientes que no cumplieron con el TAT. Especificar en un lenguaje de alto nivel el scheduler de las colas.

Ejercicio 2

Se considera una red basada en TCP/IP del tipo conocido como "*peer to peer*". En este escenario se tiene una cierta cantidad de hosts, los "*peers*", todos funcionalmente equivalentes entre si. Estas redes pueden contener grandes cantidades de *peers*, del orden de cientos de miles, por lo cual es importante optimizar los protocolos.

Cada *peer* tiene en su disco duro una serie de archivos para compartir con los otros *peers*, la cual esta disponible para el programador como una lista "*ListaArchivos*", con campos para el nombre y la longitud del archivo. También esta disponible una "*ListaPeers*" en la cual están listados todos los *peers* activos, con campos para la dirección IP y puerto. Ambas listas se suponen dato del problema.

La red necesita dos protocolos para funcionar. Un protocolo de búsquedas (basado en UDP) y un protocolo de transferencia de archivos (basado en HTTP/TCP.)

El protocolo de búsquedas simplemente envía un paquete UDP conteniendo el nombre del archivo que se busca, y el peer consultado debe responder con otro paquete UDP conteniendo las strings "YES" o "NO". Al ser basado en UDP es posible consultar a muchos peers simultáneamente y esto debe ser aprovechado para aumentar la velocidad de las búsquedas.

El protocolo de transferencia de archivos es basado en HTTP y TCP. El peer envía un comando "**GET nombre_archivo <comienzo> <largo>**" a otro peer. En caso de ser posible satisfacer el pedido, el peer consultado envia el mensaje "**200 OK**", seguido por los bytes del archivo que comienzan en <comienzo> y terminan en <comienzo + largo> y luego cierra la conexión. En caso de no poder satisfacer el pedido, el cliente responde con "**400 ERROR**" y cierra la conexión.

Se pide:

a) Implementar la función **`buscar_archivo()`** que recibe como parámetro el nombre de un archivo a buscar y devuelve una lista "*ListaPeers2*" conteniendo los *peers* que tienen el archivo buscado. Debe implementar el protocolo de búsqueda descrito anteriormente.

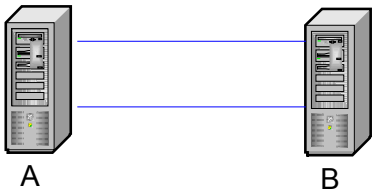
b) Implementar la función **`transferir_archivo()`** que recibe como parámetro el nombre de un archivo a buscar y lo transfiere al disco duro local.

Ambas funciones deben implementarse en pseudocódigo. Se pueden utilizar todas las primitivas clasicas para manejo de listas y para el acceso a la red deben utilizarse las primitivas de sockets vistas en el curso y en el laboratorio.

Ejercicio 3

Se considera la siguiente configuración, en la que los dos servidores (A y B) están comunicados mediante dos enlaces punto a punto, full dúplex, cuya velocidad varía en el tiempo:

Las comunicaciones entre estos dos servidores (que NO se realizan mediante TCP/IP) han sido implementadas respetando un esquema de capas como el de OSI. Cada vez que se debe enviar un paquete desde uno de los equipos hacia el otro, la capa 3 del emisor enfrenta el problema de elegir cuál de los dos enlaces debe utilizar.



En ese sentido, se desea que el tráfico se reparta, entre ambos enlaces, en forma inversamente proporcional al tiempo que demore en obtenerse la respuesta a una "solicitud de eco", la que se responderá por el mismo enlace por el que fue solicitada. Este tiempo deberá medirse periódicamente, desde cada equipo y para cada enlace.

Se pide:

a) Especificar el formato de los paquetes utilizados por el protocolo.

b) Especificar en un lenguaje de alto nivel, el protocolo que realiza el cáclulo de retardo de los enlaces.

c) Especificar en un lenguaje de alto nivel, un programa que implemente el forwarding de paquetes, según las restricciones especificadas en el problema.