# Introducción a las Redes de Computadores

# Obligatorio 5 – 2010

# Routing y Forwarding

### Introducción a las Redes de Computadoras –

**Facultad de Ingeniería**
**Instituto de Computación**
**Departamento de Arquitectura de Sistemas**

*El objetivo de este obligatorio es implementar, sobre un simulador de topologías de red, un protocolo tipo Distance Vector que cumpla con las funcionalidades de la capa de red.*

**Nota previa - IMPORTANTE**

Se debe cumplir íntegramente el "Reglamento del Instituto de Computación ante Instancias de No Individualidad en los Laboratorios", disponible en `http://www.fing.edu.uy/inco/pm/uploads/Ense%f1anza/NoIndividualidad.pdf`

En particular está prohibido utilizar documentación de otros grupos, de otros años, de cualquier índole, o hacer público código a través de cualquier medio (news, correo, papeles sobre la mesa, etc.).

**Forma de entrega**

La entrega debe realizarse mediante el formulario correspondiente en el sitio web del curso `http://www.fing.edu.uy/inco/cursos/redescomp/`. El sistema de entregas soporta múltiples entregas por grupo, llevando un histórico de las mismas. Se recomienda realizar una entrega vacía con tiempo, a los efectos de verificar que su sistema le permite entregar correctamente. Se considerará como válida la última entrega dentro del plazo asignado.

Se debe entregar un solo archivo 'ob5.tar.gz' que contenga los ítems descritos en el apartado **Entregable**. Dicho archivo deberá ser generado utilizando la herramienta GNU TAR y compresión gzip. Otros formatos (bz2, rar, zip, cab, jar, etc.) no son válidos y serán rechazados.

**Fecha de entrega**

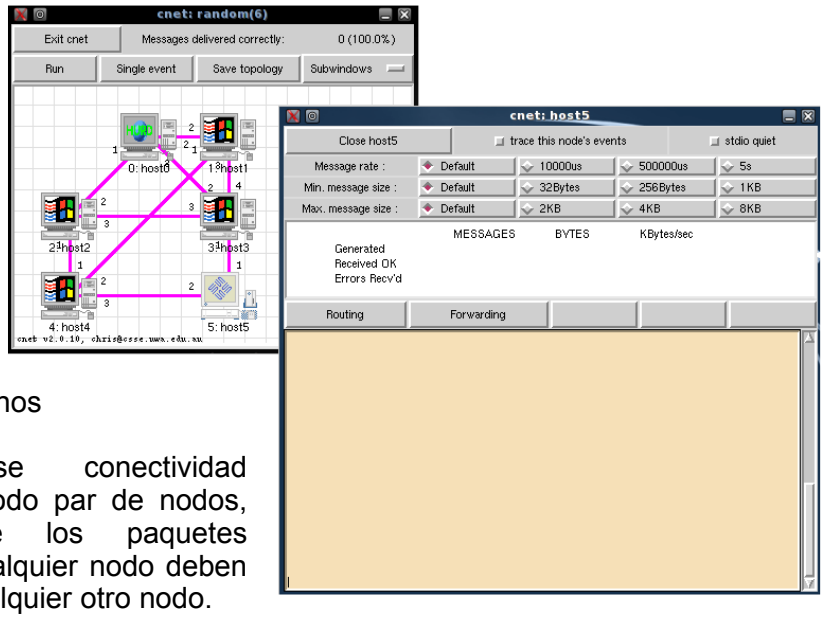
Los trabajos deberán ser entregados siguiendo el procedimiento descrito anteriormente antes del domingo 20 de junio de 2010 a las 23:30 horas, sin excepciones. No se aceptará ningún trabajo pasada la citada fecha. En particular, no se aceptarán trabajos enviados por mail a los docentes del curso, ni entregados en medios magnéticos en el instituto.

**Observaciones**

La corrección se hará en la máquina virtual del curso tal como se distribuyó, agregando solamente los componentes descritos en este obligatorio, por lo que debe probarse su funcionamiento en ese ambiente previo a la entrega.

## Descripción del problema

Las principales funcionalidades de la capa de red son el *routing* y el *forwarding* de paquetes. El routing refiere al descubrimiento por parte de los nodos de los caminos existentes hacia los demás nodos. El forwarding tiene como objetivo, para cada paquete, reenviarlo por la interfaz de salida apropiada para que llegue al nodo destino.




Se desea implementar un protocolo de capa de red en nuestro entorno de simulación que permita lo siguiente:

- Cada nodo debe poder descubrir dinámicamente los caminos óptimos hacia los demás destinos de la red.
- Debe permitirse conectividad completa entre todo par de nodos, es decir, que los paquetes generados en cualquier nodo deben poder llegar a cualquier otro nodo.
- En caso de falla de alguno de los nodos, el protocolo deberá eventualmente reacomodar las rutas para reflejar el cambio de la topología.

## Entorno de simulación CNET

CNET es un entorno de simulación de topologías de red orientado a la enseñanza que nos provee una forma rápida de crear redes de nodos que ejecuten nuestro protocolo y observar cómo interactúan. CNET implementa la capa de aplicación y la capa física de cada nodo, y queda como responsabilidad del usuario implementar las capas intermedias.

En este obligatorio implementaremos una capa de red para el simulador, que haga llegar los paquetes generados por la aplicación a cualquiera de los otros nodos. No realizaremos simulación de la capa de transporte y utilizaremos los mecanismos confiables de capa de enlace (función `CNET_write_physical_reliable`) para evitar tener que enfrentarnos a los problemas de esta capa.

Cada nodo de la simulación representará un equipo que estará ejecutando nuestro protocolo. Los nodos solamente conocen su propia dirección (tipo CnetAddr, similar a una dirección IP en una red real), y los enlaces que tienen conectados, pero no saben nada acerca de los destinos a los cuales se puede llegar siguiendo dichos enlaces. A su vez CNET en cada nodo estará generando mensajes para enviar a otros nodos, y guardará una estadística de los paquetes que logramos hacer llegar a destino.

Los protocolos a utilizar por CNET deben ser implementados en C (no permite ninguna de las extensiones de C++) y utilizar el modelo de eventos de la herramienta. El modelo de eventos permite que implementemos los handlers correspondientes a ciertas funcionalidades, y CNET invocará estos handlers en los momentos en que la simulación los necesite. Los handlers a implementar en nuestra tarea serán:

- reboot_node, un handler especial que debe estar presente en cualquier protocolo implementado en CNET.
- EV_APPLICATIONREADY, handler que se invoca cuando la capa de aplicación de un nodo tiene un paquete para enviar a otro nodo.
- EV_PHYSICALREADY, handler que se invoca cuando la capa física ha recibido datos de otro nodo.
- Los timers de CNET que sean necesarios, que son los handlers EV_TIMER*.
- Los eventos de debug que sean necesarios, que son los handlers EV_DEBUG*.

Dentro de la implementación de cada uno de los handlers, se hará uso de las funcionalidades del API de CNET para interactuar con las capas superiores e inferiores del nodo (enviar y recibir paquetes de la aplicación y de la capa física). Por más información sobre el funcionamiento de los handlers, referir a la documentación incluida en los fuentes que se instalan con CNET o a la documentación on line.

CNET nos permite la generación de topologías aleatorias para probar nuestro protocolo. Para eso alcanza con llamarle al archivo donde implementamos nuestro protocolo protocol.c y ejecutar el siguiente comando:

```
cnet -r N
```

Donde *N* es la cantidad de nodos (mayor a 1) de la topología deseada. Además de estas topologías aleatorias, se proveerá archivos con topologías para probar casos especiales.

**Protocolo a implementar**

*Routing*

Se debe implementar un protocolo de *routing* de tipo Distance Vector. En este algoritmo se busca generar una tabla de ruteo formada por las tuplas

```
<address, interface, nextHop, distance>
```

Donde:

- `address` es la CnetAddr que corresponde al destino indicado en esa fila de la tabla.
- `interface` es el número de interfaz de salida desde el nodo actual para llegar al destino.
- `nextHop` es la CnetAddr del primer nodo que se encontrará un paquete en el camino luego de salir del nodo actual.
- `distance` es la cantidad de saltos (hops) que debe tomar el paquete hasta llegar a destino.

Cada nodo estará ejecutando un proceso según los siguientes puntos:

- Inicialmente un nodo conoce solamente cómo llegar hasta él mismo, lo cual simbolizaríamos con una tupla `<my_address, 0, my_address, 0>` en la tabla de ruteo.
- Cada cierto tiempo TIMEOUT_DV el nodo enviará un paquete de control a todos sus vecinos conteniendo su dirección y su vector de distancia. El vector de distancia es una versión resumida de la tabla y contiene solo los destinos conocidos y la distancia a la que se encuentran del nodo.
- Una vez que un nodo recibe un vector distancia de uno de sus vecinos, realizará el siguiente procesamiento:
  - Para cada destino nuevo encontrado en el vector distancia, agrego una tupla para este destino a la tabla (con la distancia indicada más 1).
  - Por cada destino del vector distancia que ya tenía, si la distancia más 1 es menor a la distancia de mi camino actual, actualizo mi ruta (sobreescribo con la nueva ruta encontrada).
- Debe proveerse un mecanismo de envejecimiento de las entradas en la tabla. Si no han recibido datos de cierto nodo vecino luego de un tiempo TIMEOUT_NODE, se considerará que este nodo está caído y se eliminarán todas las rutas conocidas que pasan por este nodo.
  **NOTA:** Para esta tarea se utilizará la funcionalidad de CNET de apagar o reiniciar

nodos durante la simulación, pero no se utilizará la funcionalidad de desconectar o reconectar enlaces.
- Cada nodo recordará solamente el mejor camino aprendido hasta el momento, y descartará los demás. No se pide que se implemente ninguna mejora respecto a recordar caminos subóptimos que podrían utilizarse en caso de que se detecte como perdido el camino óptimo.

*Forwarding*

La información contenida en la tabla de routing que se va construyendo se utilizará para armar la tabla de forwarding de paquetes. Esta tabla está compuesta por las tuplas **<address, interface>** que nos indican los diferentes destinos y la interfaz de salida por la que debemos enviar el paquete para que llegue al destino.

- Cuando la capa de aplicación genera un mensaje para enviar, indicará el contenido del mensaje y la dirección del destino. Se deberá encapsular el mensaje en un paquete de datos y enviarlo por la interfaz que más lo acerque al destino.
- Cuando la capa física entrega un paquete de datos, si el paquete tiene como destino el nodo actual, se deberá entregar a la capa de aplicación. Si tiene como destino otro nodo, se lo reenviará por la interfaz que más lo acerque al destino.
- En cualquier caso, si no se conoce una interfaz por la cual enviar el paquete, se lo redirigirá por el DEFAULT_GATEWAY (en nuestro caso será la interfaz 1).
- Para evitar que un paquete pueda llegar a ser redireccionado indefinidamente por los nodos sin llegar a destino (por ejemplo en caso de encontrarse un loop en las tablas de forwarding), se limitará a MAX_TTL la cantidad de saltos que el paquete puede dar en la topología.

*Salida estándar*

Cada nodo en CNET cuenta con su propia salida estándar y eventos de debug para poder interactuar con él. Durante el transcurso de la simulación, se deberá imprimir en la salida estándar del nodo información referida a los siguientes sucesos:

- Cuando el nodo aprende un destino nuevo imprimirá: **`Destino nuevo encontrado: addr=XX interface=XX next_hop=XX distance=XX`**
- Cuando el nodo aprende un camino más corto a un destino conocido imprimirá: **`Camino nuevo encontrado: addr=XX interface=XX next_hop=XX distance=XX`**
- Cuando el nodo descubre que uno de sus vecinos está caido imprimirá: **`Se perdio contacto con un nodo: addr=XX interface=XX`**
- Cuando el nodo descubre que el camino que tenía hacia un nodo empeoró su métrica imprimirá: **`Camino aumento su metrica: addr=XX interface=XX next_hop=XX distance=XX`**
- Cuando el nodo pierde el camino conocido para llegar a algún nodo imprimirá: **`Camino eliminado: addr=XX interface=XX next_hop=XX distance=XX`**
- Cuando el nodo descarta un paquete por TTL vencido imprimirá: **`Mensaje descartado src_addr=XX dest_addr=XX size=XX`**

Además, se deben programar los siguientes eventos de debug para visualizar información sobre el funcionamiento del protocolo:

- DEBUG1: Se imprimirá la tabla de routing.
- DEBUG2: Se imprimirá la tabla de forwarding.

**Se pide**

Diseñar el formato de paquete que permita cumplir con ambas funcionalidades descritas.

Implementar un protocolo ejecutable en el entorno de simulación CNET que cumpla con las funcionalidades de capa de red especificadas.

## Entregable

Se deberá entregar:

1. Documentación, con énfasis en el protocolo, definición del formato de paquete y descripción exhaustiva del protocolo en idioma español. Definición de las estructuras utilizadas (por ejemplo, la tabla de routing). Pseudocódigo del funcionamiento de cada uno de los eventos.

2. Implementación:
a)Código fuente (implementando el pseudocódigo). El archivo principal del protocolo deberá llamarse protocol.c
b)En caso de utilizarse más de un archivo para la solución, debe entregarse un *Makefile* para la compilación del mismo, e indicar en el informe la línea de compilación para ejecutar la simulación.

## Insumos

De la página del curso se puede descargar la versión de CNET a utilizar (cnet 2.0.10). Esta versión no es la última, pero es la que presenta mejor desempeño en la máquina virtual utilizada, y nos basaremos en esta versión para este obligatorio.

Para instalar CNET en la máquina virtual, se deben descargar los cuatro archivos publicados en la web del curso: cnet-2.0.10.tgz, Makefile, libelf-0.8.6.tar.gz, y configure. Se copian estos archivos en la misma carpeta y se ejecuta el script ./configure

Se proveerá un conjunto de topologías particulares que podrán utilizarse como ejemplos para probar el protocolo a implementar.

## Nota

La documentación del obligatorio debe incluirse dentro del archivo de la entrega. La misma debe entregarse como un único archivo tipo PDF de nombre **informegrupoXX.pdf**. El mismo deberá respetar la numeración de secciones acorde a los requerimientos anteriores.

Es necesario que en el informe figuren el nombre y la cédula de identidad de cada integrante del grupo. En caso que esto no se cumpla el obligatorio no será corregido, con la consecuente pérdida del curso de sus autores. NO se aceptarán otros formatos de informe. (ver http://www.universidad.edu.uy/odfpdf/).

## Referencias

Página oficial de CNET:
http://www.csse.uwa.edu.au/cnet/
Documentación de CNET 2 (versión que utilizaremos en el obligatorio):
http://www.csse.uwa.edu.au/cnet/old-cnet2/