Universidad de Chile
Facultad de Cs. Físicas y Matemáticas
Departamento de Ciencias de la Computación
CC4303 Redes

Tarea 3 - Simulando RIP

Profesor: Catalina Álvarez

Auxiliar: Ivana Bachmann

Objetivos

En esta tarea deberán crear routers virtuales y comunicarlos; deberán simular una topología arbitraria y lograr que sus routers hagan llegar mensajes correctamente de un punto al otro, sin perder paquetes en el camino.

Eso sí, no deben hacer todo solos. Se les entrega una base para partir, un grupo de clases que simulan un router y sus puertos, y un grupo de funciones para levantar topologías dado un archivo de configuración y mandar paquetes.

Lamentablemente, la implementación de los routers está incompleta. Cuando reciben un paquete no saben como rutearlo correctamente, solo eligiendo un puerto aleatorio para forwardearlo, a veces mandándolos a ninguna parte. Es trabajo de ustedes arreglarlo, implementando RIP para llenar tablas de ruteo en sus routers.

Implementación entregada

Se les entregan cuatro archivos base sobre los cuales deben trabajar:

1. **send_packet**: Contiene una sola función, del mismo nombre, que sirve para enviar un paquete a un cierto puerto. Se les deja para poder realizar pruebas más fácilmente. Es importante notar que send_packet no realiza ningún formateo de los paquetes (eso queda a responsabilidad suya).
2. **topology**: Contiene dos funciones, start (que dado un archivo de topología crea los routers necesarios, levantando threads como sea necesario) y stop (para terminar los threads correctamente).
3. **router**: Clase que modela un router. Un router tiene puertos que utiliza para comunicarse con otros routers (estos puertos se modelan usando la clase RouterPort). Las decisiones internas de qué hacer con los paquetes las toma el router con sus funciones:

- **_success**: el paquete es para el router que lo recibe, se imprime la data que lleva en consola.
- **_new_packet_received**: cuando se recibe un nuevo paquete de un puerto, se revisa el destino del paquete: si es para el router, se llama a **_success**, sino, se elige a quien forwardear el paquete.

4. **router_port**: Clase que modela el puerto de un router; se modela como un thread que es levantado por el Router. Un router por un determinado puerto puede recibir y mandar tráfico simultáneamente, lo que no es posible (o al menos recomendado) con sockets.
   - Para modelar este comportamiento, un puerto usa dos puertos del sistema operativo, uno para enviar los datos (output) y otro para recibir los datos (input).
   - El RouterPort guarda datos que debe mandar en una cola, y revisa constantemente si esta cola tiene algo para mandar. Cuando tiene un paquete que mandar, crea un socket con el puerto de output y manda el paquete, cerrando el socket cuando termina.
   - Para recibir los datos constantemente, cada RouterPort levanta un thread hace binding del socket de entrada y está escuchando por nueva data. Cuando recibe un paquete, lo envía al router usando el método de callback definido (por defecto en la implementación, **_new_packet_received**), para que decida que hacer con él.
   - Cuando se tiene un paquete que enviar, el Router lo pone en una cola que controla el RouterPort.

Los mensajes que se envían los routers son diccionarios stringificados usando el clásico método json.dumps de Python. Los mensajes tienen dos campos: "data" (la información que se desea mandar) y "destination" (a quien va dirigido el mensaje). Tienen libertad de agregar más parámetros a los mensajes (**hint**: puede que lo necesiten).

Los archivos de topología siguen una estructura similar a la del archivo de ejemplo entregado:

1. Los routers se entregan en una lista, cuya llave es "routers".
2. Cada router tiene dos parámetros: nombre (identificador del router) y ports (puertos o "salidas" del router, lo que en clases conocemos como interface), en forma de una lista.
3. Los puertos de los routers se modelan como dos sockets: uno de entrada (input) y uno de salida (output).

Finalmente, es posible configurar el nivel de logging de los routers usando el parámetro logging al instanciarlos. Por defecto, logging está activado y muestra en consola todas las operaciones de los routers.

Implementación pedida

La implementación pedida es simple. Deben modificar la implementación de Router para incluir ruteo utilizando RIP; deben modificar algunas funciones y agregar parámetros al router. El objetivo final es que los mensajes lleguen correctamente a cualquier router sin importar la topología (por supuesto, mientras sea posible, es decir, que haya una conexión).

Ejemplo de ejecución

Para correr el ejemplo entregado, se les recomienda usar la consola de Python (y no hacerlo programáticamente), de manera de poder enviar varios paquetes a la red sin problemas y ver los mensajes de logging más fácilmente. A continuación un ejemplo que pueden correr para probar el sistema y entender como funciona:

Listing 1: Ejemplo creación de topología + envío de paquete

```
1 from topology import start, stop
2 from send_packet import send_packet
3 import json
4
5 routers = start('topology.json')
6 send_packet(4321, json.dumps({'destination': "Router#1", 'data': ...
      "Saludos!"}))
7 stop(routers)
```

Revisando la topología definida en el archivo, vemos que el puerto 4321 corresponde al input de un RouterPort del Router2 que no tiene router al otro lado. Este tipo de RouterPort se consideran "cables sueltos" a los cuales ustedes se pueden enchufar para enviar paquetes a la red; son el equivalente a la entrada de un router a la que ustedes se conectan usando un cable ethernet. De esta forma, el código anterior mete un paquete a la red con destino Router1.

Informe

Igual que para las dos tareas anteriores **deben entregar un README explicando**

como correr su tarea, y aclarando sus supuestos. Por favor consideren que mientras mejor documentada su tarea, más fácil su evaluación y menos problemas tendrán.

Implementación

Dado que la base entregada está en Python, están relativamente acotados a este lenguaje. En caso de desearlo, están autorizados de reimplementar la base en otro lenguaje y trabajar desde allí, pero no es lo recomendado; en este caso, se debe consultar primero con la auxiliar la factibilidad de revisión.

Cualquier duda o pregunta o reporte de bugs, dirigirse al foro de U-cursos.

Evaluación

Esta tarea será evaluada tomando en cuenta la implementación de RIP que incluyan: tanto llenando las tablas de ruta como ruteando correctamente. Se les evaluará en base a pruebas con una topología distinta a la que se les entrega de ejemplo por lo que no pueden hardcodear la topología entregada.

Tienen estrictamente prohibido entregarle la topología entera a todos los routers para routear, deben seguir un acercamiento distribuido. Usar un acercamiento como este será graduado con nota mínima.