Redes de Ordenadores
Implementación de un protocolo de encaminamiento

1 Introducción

Internet es una vasta colección de redes heterogéneas identificadas globalmente por un identificador único conocido como dirección de red. La dirección de una red no es más que un modo compacto de referirse a la colección de posibles direcciones IP que son alcanzables a través de una interfaz de un router. Al carecer Internet de una jerarquía de direccionamiento global, es necesario disponer de mecanismos que permitan descubrir las rutas (óptimas) entre cualquier par de redes conectadas. Esta tarea recae de manera habitual en los *protocolos de encaminamiento.*

Probablemente, los protocolos de encaminamiento más sencillos de implementar son los basados en el mecanismo de vector de distancias, siendo RIP el más conocido, a la vez que sencillo y limitado.

2 Implementación de un protocolo de encaminamiento de vector de distancias

En esta práctica los alumnos deberán implementar una sencilla variante de RIP que funcione correctamente en los equipos del laboratorio. Como todos los equipos del laboratorio se encuentran en la misma subred IP, como parte de la práctica simularemos otra topología mediante la creación de adyacencias lógicas entre los distintos procesos. Las adyacencias creadas se detallarán en el fichero de configuración que se describirá más tarde.

La tarea del programa es enviar periódicamente un vector de distancias con las redes conocidas a cada uno de los vecinos definido en el fichero de configuración y procesar los mensajes que le lleguen de sus vecinos.[1] Además, deberá enviar a la salida estándar el estado de su tabla de encaminamiento periódicamente.

El formato de los mensajes a intercambiar debe corresponderse, **obligatoriamente**, con el de una de las versiones de RIP [1].[2]

3 Requisitos

El programa entregado deberá calcular correctamente la distancia y la dirección del siguiente salto para cada una de las redes definidas en los ficheros de configuración considerando como métrica el número de saltos.

El programa podrá recibir como argumento de entrada una dirección IP (`A.B.C.D`) o una dirección IP y un puerto (`A.B.C.D:puerto`). De no haber recibido ningún parámetro, usará obligatoriamente como valor `A.B.C.D` la primera dirección IPv4 configurada en la interfaz de nombre *"eth0"*, que obtendrá de manera automática. Si no existe dicha interfaz (en el laboratorio puede asumir que sí existe) ni ha recibido como parámetro dicha dirección, el programa finalizará con un mensaje de error.

La dirección recibida sirve para dos funciones. En primer lugar, es la **única** dirección IP que debe usar para recibir los paquetes que envíen los vecinos, y el puerto el puerto de escucha UDP. De no indicarse ningún puerto, se empleará el puerto 5512.[3]

En segundo lugar, también se utiliza para determinar el nombre del fichero de configuración que se habrá creado previamente. El fichero de configuración se llamará `ripconf-A.B.C.D.topo`. El formato consistirá en una serie de líneas, cada una de ellas con uno de los siguientes formatos:

1. `E.F.G.H[:puerto]`: nombra a cada uno de los routers adyacentes. El campo `[:puerto]`, es opcional. Si aparece se corresponde con el puerto UDP donde espera recibir los paquetes el vecino. Si no aparece se asume que se emplea el puerto UDP 5512.

[1] Recomendamos usar una frecuencia media de 6 envíos por minuto.
[2] Recomendamos comprobarlo mediante el análisis del tráfico con el programa wireshark.
[3] Se considerará un error que el programa admita paquetes en cualquier otra dirección configurada en el equipo.

2. `I.J.K.L/len`: define una red alcanzable a distancia 1 desde el equipo (una ruta conectada) y, por tanto, que debe ser anunciada en el vector de distancias.

Adicionalmente, el programa debe anunciar la red `A.B.C.D/32` a los vecinos, siendo nuevamente `A.B.C.D` la dirección IP de la interfaz Ethernet del ordenador del laboratorio donde se está ejecutando.

Por tanto, las líneas de formato `E.F.G.H[:puerto]` (sin longitud de prefijo) serán las empleadas para definir la topología sobre la que se calculan las rutas.

Opcionalmente se podrán añadir las siguientes funcionalidades, que serán valoradas en la calificación final de la práctica:

1. Split-horizon en alguna de sus variantes. Apartado 3.4.3 del estándar.
2. Triggered updates. Apartado 3.4.4 del estándar.
3. Autenticación básica de las tramas. Apartado 5.2 del estándar.
4. Autenticación criptográfica. Ver [2].

Cada una de las funcionalidades extras permiten obtener un punto adicional, excepto la última que se valora con hasta dos puntos extra. En todo caso, la nota final nunca excederá de diez puntos.

En el caso de implementar autenticación, el programa debe **obligatoriamente** solicitar una contraseña al usuario nada más arrancar.

4 Entrega y evaluación

El código podrá ser desarrollado en C, C++, Python o Java a elección del alumno, aunque obligatoriamente con las versiones instaladas en el laboratorio. En los dos primeros casos es necesario incluir un Makefile para facilitar la compilación, mientras que en el caso de Java la clase principal debe estar en el *paquete por defecto* y llamarse *Rip*.

Todo el código desarrollado se incluirá en un único fichero de nombre "ro1516-*grupo*.tar.gz". A la hora de evaluarlo, se tendrán en cuenta los siguientes puntos:

- Funcionalidades incluidas en el programa.
- Estructuración y legibilidad del código fuente.

En particular, la legibilidad del código fuente se considerará requisito imprescindible para poder superar la práctica.

Referencias

[1] G. Malkin, RIP version 2, RFC 2453 (Nov. 1998).
URL https://tools.ietf.org/html/rfc2453

[2] R. Atkinson, RIPv2 cryptographic authentication, RFC 4822 (Feb. 2007).
URL https://tools.ietf.org/html/rfc4822