Práctica de Conmutación de Paquetes: Route Lookup

1. Objetivos

- Introducir al alumno en el problema de la búsqueda del siguiente salto (*route lookup*) en un router CIDR, y su complejidad.
- Familiarizar al alumno con mecanismos básicos de *route lookup*.
- Lectura de algoritmos en la literatura.
- Implementar un algoritmo de *route lookup* libre que mejore el algoritmo de búsqueda lineal de referencia.
- Evaluar las prestaciones ofrecidas por el algoritmo de búsqueda implementado, en comparación con el método de búsqueda lineal.
- Plantear el reto de mejorar los algoritmos existentes.

2. Documentación y selección de un algoritmo

Lea los artículos [1] y [2]. El algoritmo de referencia está descrito en [1] (página 8 *Linear Search of Hash Tables*), mientras que el algoritmo que se pide implementar está descrito en [2]. El concreto se trata de un ***Multi-bit trie* en *hardware* de dos niveles.**

Tenga en cuenta que cualquier algoritmo más eficiente es aceptable como solución. Por tanto cabe la posibilidad de que el alumno proponga e implemente algún algoritmo distinto al descrito en [2]

3. Especificaciones

El programa implementado por el alumno deberá cumplir con los siguientes requisitos:

- Lenguaje de programación: C.
- Sistema Operativo: Linux.
- Límite de memoria: 50 MB de memoria RAM, para 25000 rutas con prefijos de todo tipo de longitud (0 a 32). Puede suponer que el número de posibles *next-hops* o interfaces es inferior a 32000.
- Se desarrollará y entregará un programa en C que genere un único ejecutable: **my_route_lookup**, con dos parámetros que se pasarán de esta manera:

./**my_route_lookup** *FIB InputPacketFile*

donde:

- *FIB*: nombre del fichero ASCII que contiene la tabla de reenvío.
- *InputPacketFile*: nombre del fichero ASCII con un listado de direcciones IP destino a procesar, separados por salto de línea (**\n** en lenguaje C)

- El archivo *FIB* será un fichero de texto representando una tabla de reenvío simplificada[1]

[1] Recuerde que una FIB completa real incluiría además de la interfaz de salida, la dirección IP del siguiente salto, e información de reenvío de nivel 2 (normalmente, la dirección de nivel 2 asociada a ese vecino (típicamente dirección MAC ethernet), información de encapsulación del paquete, y el *handler* o función *hardware/software* a invocar con estos parámetros y el puntero al paquete IP en cuestión).

con una entrada por línea con pares <*Prefijo_de_Red_CIDR, Interfaz de salida*>. Ambos campos estarán separados por un único carácter de tabulación (**\t** en lenguaje C). Ejemplo:

Prefijo de Red	*Interfaz de salida*
192.163.10.0/24	1
192.163.0.0/16	2
193.110.27.192/26	3

- Para cada dirección IP destino contenida en el fichero de entrada *InputPacketFile*, el programa computará el interfaz de salida por el que debería encaminar un paquete hacia dicho destino de acuerdo a la *FIB*.
- Asuma que los ficheros de entrada no tendrán direcciones o prefijos incorrectos en ningún caso.
- Las interfaces del *router* se numeran a partir de 1. **El 0 representa la no existencia de una interfaz**.
- Si no existe una ruta por defecto, el paquete procesado se descarta.
- Debe decidir cómo modelar la ruta por defecto en su implementación.
- Como salida, el programa generará un único fichero, *OutPutFile*, en el que se almacenará una tupla <*IPaddress, OutIfc, AccessedTables, ComputationTime*> en cada línea, para cada dirección IP del fichero de entrada (*InputPacketFile*). Los campos de la tupla se definen a continuación:

  - *IPaddress*: la dirección IP leída del fichero *InputPacketFile*.
  - *OutIfc*: dirección IP del siguiente salto correspondiente a la dirección destino (se imprimirá la cadena **MISS**, si no se encuentra interfaz de salida).
  - *AccessedTables*: número de accesos a memoria necesarios para obtener la dirección de la interfaz de salida.
  - *ComputationTime*: tiempo necesario para obtener la interfaz de salida correspondiente a la dirección IP destino. Este tiempo incluye ÚNICAMENTE el tiempo de cálculo de la interfaz de salida en la FIB, y NO INCLUYE el tiempo consumido en operaciones de lectura / escritura en ficheros. El tiempo de cómputo será un valor entero expresado en nanosegundos.

- El fichero *OutPutFile*, cumplirá las siguientes condiciones:
  - El nombre del fichero se obtendrá a partir del nombre del fichero de entrada, concatenándole a éste la cadena "**.out**".
  - Contendrá una única tupla por línea.
  - Los campos de cada tupla estarán delimitados por el carácter "**;**". Tras el último campo (Tiempo de computación) se imprimirá únicamente un salto de línea.
  - Entre dos tuplas cualesquiera no se imprimirán líneas en blanco.
  - Se imprimirá una única línea en blanco tras la última tupla.
  - A continuación se indicará el número total de direcciones IP destino procesadas, el número medio de tablas accedidas y el tiempo medio de cómputo correspondiente a las direcciones procesadas, estos dos últimos con dos decimales de precisión. Asimismo, se indicará la memoria y el tiempo de CPU consumidos. Para ello se utilizará el siguiente formato:

```
Packets processed= X
Average table accesses= Y.YY
Average packet processing time (nsecs)= Z.ZZ
Memory (Kbytes) = KB
CPU Time (secs)= T.TTTTT
```

Estos parámetros relativos al rendimiento junto con el número de aciertos se tendrán en cuenta a la hora de calificar la práctica.

4. Ejemplo

A continuación se muestra un ejemplo completo de ejecución del programa, junto con la salida obtenida:

File: r.txt	
192.163.10.0/24	1
192.163.0.0/16	2
193.110.27.192/26	3

File: i.txt
192.163.10.123
195.50.230.11
193.110.27.224

Command line	File: i.txt.out
pract> ./my_route_lookup r.txt i.txt pract>_	192.163.10.123;1;1;2030 195.50.230.11;MISS;2;584 193.110.27.224;3;2;426 Packets processed= 3 Average table accesses= 1.66 Average packet processing time (nsecs)= 1013.33 Memory (Kbytes) = 21258 CPU Time (secs)= 0.044427

5. Ficheros proporcionados

Para facilitar al alumno que pueda reproducir el formato del fichero de salida, se le proporcionan ficheros con funciones que le serán de utilidad para desarrollar la práctica. Los ficheros suministrados son:

- **io.c/h:** Estos ficheros tienen el código necesario para parsear una tabla de rutas y leer un fichero con direcciones IP a encaminar como los que se le darán para procesar en su práctica. Hay funciones que le permiten imprimir también un fichero en el formato que se le pide en apartados anteriores.
- **utils.c/h:** Estos ficheros tienen una función que calcula una máscara IP en función de la longitud de prefijo. Estos ficheros también incluyen una función de *hash* de ejemplo que puede serle de utilidad en caso que decida implementar un algoritmo diferente de [2] que utilice tablas de *hash*. Puede utilizar otras funciones de hash que le convengan más si lo desea.
- **routing_table.txt/routing_table_simple.txt:** Tablas de rutas de ejemplo que usted deberá procesar en su práctica para asegurarse de que funciona correctamente.
- **prueba0/1/2/3.txt:** Ficheros con direcciones IP de ejemplo que usted deberá encaminar como se especifica en el enunciado de esta práctica.
- **Makefile:** *Makefile* de ejemplo que le muestra cómo generar el ejecutable que se le pide. Usted deberá modificar este fichero cambiando la lista de ficheros que revisa el *Makefile* para que se ajuste a los de su práctica.
- **linearSearch:** Ejecutable que hace una búsqueda lineal y que se utilizará como referencia para comparar con su solución.

6. Entrega de la práctica

La fecha límite de entrega de la práctica es el viernes 23 de Marzo de 2018 a las 23:59 CET.

El alumno debe entregar un fichero *.zip* a través del sistema de entregas de Aula Global, cumpliendo con los siguientes requisitos:

- El *zip* debe llamarse GRUPO0x. Si el grupo es el 5: GRUPO05.zip.
- El *zip* debe incluir todo el código que sea necesario (tanto vuestro como proporcionado como referencia) para poder compilar y ejecutar vuestra práctica.

- El *zip* debe incluir un *Makefile* que permita compilar la práctica automáticamente. El *Makefile* no debe ejecutarse siempre. En su lugar, debe tener la forma:

all: my_route_lookup

my_route_lookup: <lista_de_ficheros>

En este makefile, el objetivo "*all*" depende del objetivo "*my_route_lookup*". El objetivo "*my_route_lookup*" se generará siempre que:

a) No exista el fichero objetivo "my_route_lookup".

b) Cambie alguno de los ficheros de <lista_de_ficheros>.

La lista <lista_de_ficheros> es la lista con todos los ficheros fuente de vuestro programa (tanto vuestros como proporcionados como parte de la práctica).

Aparte, debe incluirse siempre la directiva:

.PHONY: clean

Para indicar que *clean* no es un fichero.

- El *zip* también debe contener un fichero llamado "nias.txt", que debe contener dos líneas con los NIAS de la pareja que realiza la práctica. Por ejemplo:

100012345
100023456

- Los ficheros deben comprimirse directamente en la raíz del *zip*, no dentro de una carpeta.

- No se corregirá ninguna práctica que no cumpla con estos requisitos.

El número asociado a cada grupo será el escogido en el formulario de Aula Global. El registro de los grupos estará abierto hasta la finalización de la segunda sesión de laboratorio de la práctica.

En caso de que el número de estudiantes inscritos sea impar y un único alumno figure inscrito en solitario, éste podrá ser reagrupado en un grupo de 3 integrantes.

NOTA IMPORTANTE

Es responsabilidad del alumno tanto el no copiar código como el proteger su propio código en todo momento. Con la entrega de la práctica, el alumno declara implícitamente la autoría del código entregado. El único código ajeno a emplear debe ser el proporcionado por los profesores. La utilización de librerías Open Source para las tablas hash debe consultarse con los profesores. Se utilizará un detector de copias. Los casos de copia de código fuente se resolverán a través de la autoridad académica correspondiente. Tanto copiador como copiado serán sancionados.

7. Evaluación y condiciones

No se corregirá ninguna entrega que:

- No compile
- Tenga fugas de memoria
- Produzca errores de ejecución en los tests de evaluación.
- Utilice más de 50 MB de memoria RAM, para 25000 rutas con prefijos de todo tipo de longitud (0 a 32). Nótese que esto es más exigente que en los casos reales.
- Nunca calcule el siguiente salto correctamente.

Puede suponer que el número de posibles *next-hops* o interfaces es inferior a 32000.

La calificación constará de dos partes:

1. **Entrega de código.** Se evaluará la mejora de las prestaciones del esquema de búsqueda lineal en tablas, así como la claridad y organización del código.
2. **Evaluación en laboratorio.** Esta evaluación consta de dos partes. En la segunda sesión de laboratorio los alumnos deberán presentar un pseudocódigo donde se esboce el algoritmo que van a implementar y la estructura de datos que utilizarán como base en dicho algoritmo. Tras la entrega se realizará un examen presencial en el que se preguntará a ambos miembros del grupo de forma individual en detalle sobre el diseño e implementación de las partes del programa entregado. También se pueden realizar preguntas sobre las características técnicas de la solución escogida en términos de tiempo de ejecución y memoria necesaria, así como sus limitaciones.

8. Bibliografía

[1] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner. Scalable High Speed Prefix Matching. ACM Transactions on Computer Systems, 2001.

[2] P. Gupta, S. Lin, and N. McKeown. Routing Lookups in Hardware at Memory Access Speeds. IEEE INFOCOM, April 1998, Vol 3, pp. 1240-1247, San Francisco.