Algoritmos y Estructuras de Datos

Guía de laboratorio
Taller de Sorting

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

En este taller obligatorio se propone implementar un conjunto de algoritmos de ordenamiento para resolver algunos problemas -parecidos a los reales- en el contexto de la infraestructura de Internet. Para aprobar el taller, todos los tests que les proveemos deben pasar. Recuerden subir únicamente el archivo `InternetToolkit.java` y las clases creadas por ustedes (ej.: `HeapSort.java`) al repositorio de entrega dentro de la carpeta `taller5`.

Última fecha de entrega: domingo 03 de diciembre.

1. Introducción

Internet está determinada por millones de equipos comunicándose entre sí. Miles de fabricantes de hardware y sistemas operativos distintos consiguen intercambiar información de forma interactiva. Es verdaderamente un milagro técnico, dado que el protocolo de comunicación dominante -IP (Internet Protocol)- es "best effort", es decir que no garantiza que los mensajes lleguen a destino, o que lleguen ordenados.

En Internet los equipos se dividen en dos grandes grupos: los hosts o equipos terminales y los routers o equipos intermedios. Los hosts son todos los clientes y servidores distribuidos por el mundo que consumen y ofrecen servicios. Mientras que los routers tienen la tarea de rutear los mensajes entre el host origen y el host destino, son equipos que tienen varias interfaces (placas) de red y tablas de ruteo mediante las cuales son capaces de enviar los mensajes por la mejor ruta posible (de acuerdo a distintos criterios).

Para lograr establecer una verdadera conexión entre los hosts origen y destino se creó TCP (Transmission Control Protocol), un protocolo que está "por encima" de IP, esto quiere decir que en la parte de los datos (payload) de IP se envía la información TCP que a su vez contiene la información de algún protocolo de aplicación (por ejemplo HTTP). Esta arquitectura en capas -que parecen mamushkas- se denomina modelo OSI.

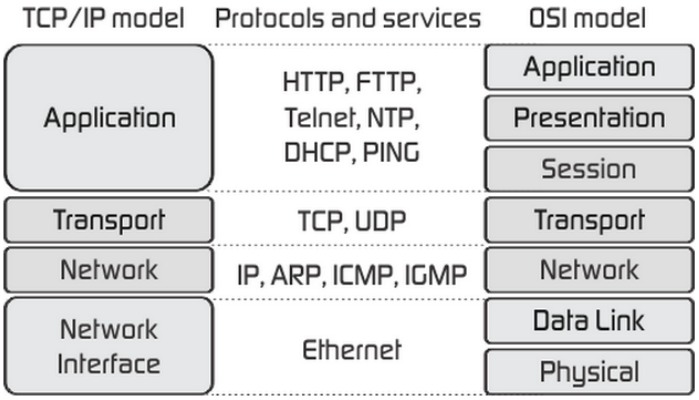

Figura 1: Modelo real TCP/IP y el modelo teórico OSI de 7 capas

TCP tiene, entre otras, las siguientes características: 1) es confiable: el receptor le notifica al emisor la correcta recepción de la información, 2) es ordenado: reordena la información, 3) tiene control de flujo: el receptor puede notificar al emisor que modifique la tasa de envío de datos, etc.

2. Ejercicios

Se pide elaborar la clase `InternetToolkit.java` que provea un conjunto de funciones para resolver algunos problemas ligados al tráfico de Internet.

Ejercicio 1

A pesar de que IP no es confiable los mensajes en general llegan en orden. Es por esto que la probabilidad de que en una conexión TCP haya un mensaje desordenado es 0,01. Se pide escribir el siguiente método:

`Fragment[] tcpReorder(Fragment[] fragmentos)`: dado un arreglo que contiene los fragmentos de los mensajes en el orden en que fueron recibidos, devolver el mismo arreglo con los fragmentos ordenados. Dado que la probabilidad de que

haya un elemento fuera de orden es muy baja y además en ese caso el elemento en general está ubicado en una posición "relativamente cercana" a la que le corresponde, considerar la posibilidad de implementar un algoritmo que en el caso promedio tenga complejidad O(n).

Ejercicio 2

Es muy importante que los routers no se saturen porque en ese caso empiezan a perder paquetes de datos. Para eso periódicamente se analiza el volumen de tráfico a nivel global. Esa información es relevante para crear dinámicamente nuevas rutas que descongestionen a los routers más saturados. Se pide escribir el siguiente método:

`Router[] kTopRouters(Router[] routers, int k, int umbral)`: dado un arreglo de Routers, devolver otro arreglo de Routers ordenado por tráfico de forma decreciente, que contenga a lo sumo los k routers con mayor tráfico que superen el umbral. Se pide que la complejidad del algoritmo sea O(n + k log n), donde n es la cantidad de routers de entrada.

Ejercicio 3

La forma de indentificar a los equipos en Internet es a través de las direcciones IP. Existen 2 tipos de direcciones: IPv4 e IPv6 que corresponden a dos versiones distintas del protocolo IP. Los dos tipos de direcciones conviven desde hace varios años. Usualmente los fabricantes de routers necesitan generar reportes estadísticos del tráfico para luego poder mejorar sus capacidades.

Una dirección IPv4 es una secuencia de 32 bits que usualmente se describe mediante 4 números enteros en el rango [0,255] -denominados *octetos*- separados por puntos, por ejemplo: "192.168.1.1".

Se pide escribir el siguiente método:

IPv4[] sortIPv4(String[] ipv4): dado un arreglo de direcciones IPv4, se pide ordenarlas de forma creciente de acuerdo a los 4 octetos que las componen. El orden considera que el octeto más significativo es el de la izquierda y los demás se organizan de forma decreciente de significancia. Considerar los siguientes ejemplos:

- 192.168.1.1 > 60.41.2.45
- 192.168.1.1 > 192.168.0.1
- 192.168.1.1 < 192.168.1.2

Nota: en este último ejercicio no hay una restricción de complejidad, se admiten soluciones O(n^2) donde n es la cantidad de direcciones ip de entrada.