Middle East Technical University | Department of Computer Engineering

CENG 213

Data Structures

Fall '2015-2016

Programming Assignment 3

Due date: 4 January 2016, Sunday, 23:55

1 Objectives

This assignment aims to make you familiar with hashing techniques and design an efficient domain name server (DNS) manager program to resolve the given Internet Protocol (IP) addresses of different domain names (URLs) by using a distributed hash tables mechanism.

Keywords: *Hashing, distributed hash table, domain name server*

2 Problem Definition

An IP address is arranged as a string of four sets of digits, such as 31.13.93.36, that acts as an identifier for devices across the Internet (WWW). Computers, servers, and other online devices know each other according to their unique IP addresses. However, as memorizing these addresses are difficult for end users and the IP addresses may change over time, websites are generally referred to by their URLs, such as "www.facebook.com".

Each DNS does the job of mapping domain names to IP addresses to reach the targeted website. For example, when you enter "www.facebook.com" in your browser, the DNS resolves the URL into an IP address, such as 31.13.93.36, for reaching Facebook Web server machines. This process must happen as fast as possible to avoid delays for the users.

In this homework, you are asked to implement a DNS manager program to resolve a given URL into its corresponding IP address. To make the problem more challenging and also better simulate the real world, there will be multiple domain name servers. The *master* DNS may contain the actual IP address of a URL or it may contain the address of another DNS which contains the actual IP address. In other words, to resolve an IP address, you may have to *hop* multiple times until you reach the DNS that has the actual IP address.

3 DNS Manager Class

This is the main class that you are expected to implement. It should support the following interface functions:

`void registerDNS(const string& dnsIP)` Register the given IP address as the address of a valid domain name server. The first registered dnsIP is the master DNS.

`void deleteDNS(const string& dnsIP)` Delete the DNS with the given IP address and all of its contents.

`void registerURL(const string& url, const string& ip, const vector<string>& dnsChain)` Register the given URL and its IP address. dnsChain contains the IP addresses of one or more domain name servers. The first one in this chain must match the IP address of the master DNS. The given IP address must be stored in the last element of this chain with each previous element storing the next address in this chain.

`void deleteURL(const string& url)` Delete all information about this URL from the system.

`string access(const string& url, int& accessCount, int& hopCount)` Access the given URL and return its IP address if this URL was previously registered (and not deleted thereafter). Also return, through output variables, the number of accesses to this URL and how many hops was made to find its actual IP address. Otherwise return "`not found`".

`void mergeDNS(const string& dnsIP1, const string& dnsIP2)` Add the entire contents of the server indicated by `dnsIP2` into `dnsIP1` and delete `dnsIP2` from the system.

4 Important Notes

- You must use hash tables for implementing this assignment. We will register millions of URLs and time our accesses. If the access times are found to be larger than expected from a correct hash implementation, there will be a "significant" penalty of grade reduction.
- You can use any hash function that you want. But make sure that the factors used in the hash function and the table size are prime numbers.
- You are strongly recommended to use **open addressing**. You may use linear hashing, quadratic hashing, or double hashing for implementing your hash tables. Faster implementations will receive a "bonus" grade (assuming that they are correct).
- Make sure to keep the load factor less than or equal to 0.5.
- You can only use the `vector` and `string` C++ libraries. You cannot use any other code or libraries from other sources. Using codes from the lecture slides is allowed.

5 Example Usage Scenario

Below is a simplified sample usage scenario:

```
DNSManager mgr;

mgr.registerDNS("100.200.300.400");
mgr.registerDNS("150.250.350.450");

vector<string> dnsChain;
dnsChain.push_back("100.200.300.400");
mgr.registerURL("www.facebook.com", "31.13.93.36", dnsChain);

dnsChain.push_back("150.250.350.450");
mgr.registerURL("www.google.com", "216.58.209.164", dnsChain);
mgr.registerURL("www.intel.com", "195.175.114.195", dnsChain);
```

At this point, Facebook's real IP address is stored in the first DNS. Google's and Intel's real IP addresses are stored in the second one. For both of them, the first DNS stores the IP address of the second DNS.

```
string ip;
int accessCount, hopCount;
ip = mgr.access("www.facebook.com", accessCount, hopCount);
ip = mgr.access("www.google.com", accessCount, hopCount);
ip = mgr.access("www.facebook.com", accessCount, hopCount);
```

For the first access to Facebook, `accessCount` and `hopCount` should both be 1 (the hop count is 1 because we reach the real IP address by going over one DNS). For the Google access, the `accessCount` should be 1 and the `hopCount` should be 2. For the second access to Facebook, `accessCount` becomes 2 and the `hopCount` stays at 1.

```
mgr.deleteURL("www.google.com");
ip = mgr.access("www.google.com", accessCount, hopcount);
mgr.deleteDNS("150.250.350.450");
ip = mgr.access("www.intel.com", accessCount, hopcount);
```

Both accesses should return "`not found`". The Google access fails because its URL has been deleted from the system. The Intel access fails because the DNS which contains Intel's information is deleted.

6 Regulations

1. **Programming Language:** You will use C++.
2. You are not allowed to use any libraries/includes except for the ones specified above.
3. **Late Submission:** Refer to the syllabus for late submission policy.
4. **Cheating:** We have zero tolerance policy for cheating. In case of cheating, all parts involved (source(s) and receiver(s)) get zero. People involved in cheating will be punished according to the university regulations.

5. Remember that students of this course are bound to code of honor and its violation is subject to disciplinary action.

6. **Newsgroup:** You must follow the newsgroup (news.ceng.metu.edu.tr) for discussions and possible updates on a daily basis.

7 Submission

- Submission will be done via COW.
- Create and submit *hw3.tar.gz* that contains at least *DNSManager.h*. Do not modify the public interface.
- You may include other .h and .cpp files if you wish (for example, you may implement DNSManager in a separate .cpp file). But do not submit a source file which contains a main function. It will be provided by us.
- We will compile your codes as follows: `g++ *.cpp -o hw3`.
- If your submission fails to follow these specifications and therefore does not compile, there will be a "significant" penalty of grade reduction.