CSCI 4211: Introduction to Computer Networks
Spring 2023
PROGRAMMING PROJECT 1

Due 11:59pm Wednesday March 1st

The Domain Name System (DNS) provides a vital service to the modern Internet, translating human-friendly and easily readable hostnames to the IP addresses routing protocols used to direct connections and packets. Many protocols such as HTTP, SMTP, and FTP use this the DNS system.

Your Assignment:

Create a simple server servicing requests for address to IP translation (DNS)

- The server should listen on port 9889, use TCP, and be able to handle multiple requests at the same time (e.g., you will need to use threading). Servicing a request to completion should not close the server down.
- The server should utilize a basic DNS algorithm you will write:
    1. A Query comes in from a client or application
        a. <hostname> like www.google.com or www.myserver.org

    Note: www.google.com and google.com are separate DNS entries and can point to two different servers. For most sites one is just a CNAME of the other, meaning that they both end up pointing to the same server, but this is not a requirement.

    2. Check the host's local cache (this can be a file system, database, etc.)
        a. If the local cache has the response, send the entry as the response
        b. If not in the local cache, attempt to resolve with a DNS API call (such as *gethostbyname*), if the hostname is not resolvable return an error message as the response
            i. Note: it is possible for DNS to resolve to multiple addresses, take the first one
            ii. (optional) if resolving multiple addresses use ping to return the fastest address
        c. Save the response into the local cache (if not already present) to reduce the need to resolve via the DNS API for future queries
            i. Cache Format: <hostname>,<ipaddress>

        ii. E.g., www.google.com,172.217.0.164
            www.notfound.no,Host not found

3. Send the response back
    a. Response Format:<hostname>:<answer>:<how request was resolved>
    b. Examples:
        i. www.google.com:172.217.0.164:API
        ii. www.notfound.no:Host not found:API
        iii. www.google.com:172.217.0.164:CACHE

- The server should output a log file of the requests in a file (as dns-server-log.csv)
    - <hostname>,<answer>,<how request was resolved>
    - An example has been included (logfileexample.csv)
- The cache stores the answers for the queries you received. Load this file when the server starts up, load the data into a data structure. You continue to use this data structure for the reads and writes. When the server stops, you write out the data structure to the cache file. Note that this is different from the dns-server-log.csv, which logs an additional format about how you resolved each query using API or CACHE.

What to Submit:

- **A zip file** (.tar.xz or .zip formats), named as <StudentID-FirstName_LastName>-project1.zip (or .tar.xz) (example: 1234567-Alice_Bob-project1.tar.xz), to the class canvas site containing:
    1) **Your server source code**
        a) (You can use any language you wish, but it must run, compile, and execute on a CSE-Lab machine (linux))
            - If using a compiled language (like c, c++) you must include a compiled executable of your code and any make files you used
    2) **A log file** from your server showing the queries and their resolutions (dns-server-log.csv)
        a) Formatted as each line <hostname>,<answer>,<jpw request was resolved>
        b) E.g.,
            - www.google.com,172.217.0.164,API
            - www.nowhere.no,Host not found,API
            - www.google.com,172.217.0.164,CACHE
    3) **A readme file** (as README.md using Markdown syntax) with sections:
        a) A header (comprised of two lines)
            - <Your name>,CSCI4211S23,<current date (month/day/year)>

        - E.g., Marky Mark,CSCI4211S23,03/01/2023
        - <coding language>,<server source code file name>,<compile script (if any)>,<executable file name>
        - E.g.,
            - Java,server.java,compile.sh,server.class
            - Python3,server.py,,server.py
                - Note: ‘,,’ is not a typo, here it means “No compile script needed”
            - Python2,server.py,,server.py
            - C,server.c,makefile,main
            - C++,server.cpp,makefile,main
            - Go,server.go,build.sh,main

b) <u>Compilation section</u>
    - Step by step instructions on how to compile and setup your code
    - E.g.,
        1. Run gcc -o test test.c in command line
        2. Copy input.txt to dir/
        3. Etc.…

c) <u>Execution/Running section</u>
    - Step by step instruction on how to run/execute your program
    - E.g.,
        1. Run myprogram.exe
        2. Etc.…

d) <u>Description section</u>
    - Describe overall what your program does and how logical it operates, be detailed
    - E.g., This program does x with y and creates z ...

To help with the assignment:

- Some client code has been written that you can use to test your server and included with the assignment zip
    - How to use the client code:
        - For each DNS request, client opens a socket to server to query its desired host names as needed, providing host name for the query.
        - The format of the query will have the host name “www.yahoo.com” or “google.com” (of course for stdin input, you need to follow that with a carriage return).

        - Holding the socket open, the client will wait for the server to respond with a corresponding IP address (or an error message, or "Host not found").
        - Entering a 'q' or 'Q' from input, terminates the clients program
            - All the sockets will be closed
            - Server will remain online
- Skeleton server code has been written to help you structure your server code and included with the assignment zip

Additional project implementation instructions:

- Our evaluation script is going to run your code. For that, we need you to keep the DNS file up to date when the program is closed by the script. One way to do that is to start a new thread that saves the cache to the file every 15 seconds.
- Make sure that your program can create your cache file and "dns-server-log.csv" file if they don't exist in the current working directory.

Example python code is given here:

```
# Spring 2023 CSci4211: Introduction to Computer Networks
# This program serves as the server of DNS query.
# Written in Python v3.

import sys, threading, os, random
from socket import *
import time
DNS_FILE = "DNS_mapping.txt"

def main():

        # Check if DSN_FILE exists (creates it if it does not)
        # Read DNS FILE
        # Save cache information to a data structure

        host = "localhost" # Hostname. It can be changed to anything you desire.
        port = 5001 # Port number.

        #create a socket object, SOCK_STREAM for TCP

        #bind socket to the current address on port

        #Listen on the given socket maximum number of connections queued is 20

        monitor = threading.Thread(target=monitorQuit, args=[])

        monitor.start()

        save = threading.Thread(target=saveFile, args=[])

        save.start()
        print("Server is listening...")
        while 1:

                #blocked until a remote machine connects to the local port

                connectionSock, addr = serverSocket.accept()

                server = threading.Thread(target=dnsQuery, args=[connectionSock,
addr[0]])

                server.start()

def dnsQuery(connectionSock, srcAddress):

        #match with the query sent from the client
        #check the data structure to see if the host name exists

         #If match, use the entry in cache.

              #However, we may get multiple IP addresses in cache, so call dnsSelection
to select one.

        #If no match, query the local machine DNS lookup to get the IP resolution

        #print response to the terminal

        #send the response back to the client

        #Close the server socket.
        #update "dns-server-log.csv" file

def dnsSelection(ipList):

        #checking the number of IP addresses in the cache

        #if there is only one IP address, return the IP address

        #if there are multiple IP addresses, select one and return.

        ##optional: return the IP address according to the Ping value for better
performance (lower latency)

def saveFile():
    while(1):
        #check updates from data structure and save your "DNS_mapping.txt" here
        #enter your code below
        time.sleep(15)


def monitorQuit():

    while 1:
        sentence = input()
        if sentence == "exit":
            os.kill(os.getpid(),9)



main()
```