CS 4461 Programming Assignment 2

Spring 2017

Due: 17 April 2017

Let's start with a high-level view of what must be accomplished in this assignment. Use this as your "North Star" for making big progress. As you get closer to submission, consult the sections below to make sure your program meets all of the requirements of the assignment.

The Big Picture

Your assignment is composed of three small programs. The first program is a simple *port mapper*. It should listen on UDP port 4461. It will receive a request on that port to learn the port number for a given service. We will only have one service running (a DNS lookup), but your mapper should be able to respond to a request for any number of possible services. The mapper should then respond to the request back to the original requestor with the port number of the requested service. In this case, you can choose any port number you like.

The second program is simplified Domain Name Server. It will listen on same UDP port you chose above for DNS queries. The "official" DNS query format is more complicated than we want to deal with, so this will just be a simple ASCII string of the URL to lookup. The program will search through a limited set of entries and respond with an IPv4 address of the requested server or an error response if the requested server isn't in the limited database.

The final program is a client that performs the DNS lookup. It should start by connecting to the default port (4461) and asking what the appropriate port is for DNS lookups. Upon receiving the port number you chose, the client terminates the first connection and creates a new connection to the DNS program. It will make the DNS query and print the result.

A sample run of the program is shown below.

```
> ./dnslookup 127.0.0.1 google.com
DNSLookup: Contacting port mapper...
DNSLookup: Connected to port mapper on port 4461.
DNSLookup: DNS lookups are on port 4723.
```

```
DNSLookup: Disconnected from port mapper.
DNSLookup: Contacting DNS Resolver...
DNSLookup: Connected to DNS on port 4723.
DNSLookup: google.com is 216.58.192.142
DNSLookup: Disconnected from DNS.
>
```

The exact formatting of the messages is not important, but the basic sequence is. The details are listed below.

The Port Mapper

You may use C#, Java, Python, C or C++ for this program. The port mapper shall be a standalone program listening for UDP connections on port 4461. It must use sockets for communication.

A properly formatted request should simply be an ASCII string. In our case, we are simply looking for "DNS". Any other value should return an error code.

The response will be two integers. The first will be a Boolean flag that indicates if the port number returned is valid (1) or not (0). The second number is the port number for the requested service. Both integers should be in one packet. You may send them as a `struct` or `class` or you may create a string and send the values that way. A valid request for the DNS port should be something like `(1, 4723)`. An invalid request should be something like `(0, 0000)`.

This program should be able to be tested on its own. While it is not required to have this program print out status messages, it helps in the grading.

DNS Resolver

You may use the same selection of languages for this program. The DNS resolver shall be a standalone program listening for UDP connections on the port of your choosing (be sure not to use something important). It must also use sockets for communication.

A properly formatted request should simply be an ASCII string of the URL (e.g., `google.com`).

A properly formatted request should again be an ASCII string, `struct` or `class` that includes the valid/invalid code (as in the Port Mapper specification, above) and the four octets of an IPv4 address. A valid request for the google.com IP would be `(1, 216.58.192.142)`. A request for an improperly formatted hostname, unknown hostname or improperly formatted request should be something like `(0, 0.0.0.0)`.

This program should also be able to be tested on its own. While it is not required to have this program print out status messages, it helps in the grading.

The Client

The client may be written in C, C++, C#, Java or Python as well. It shall be a standalone program that makes a properly formatted request to the port mapper, uses the response from the port mapper to make a DNS request to your resolver and prints out the response.

It must also print out status messages of what the program is doing. You *must* print out status messages for the following events:

- Beginning the connection to the port mapper or resolver.
- Terminating the connection.
- Initiating the request.
- The result of the request (in other words, the response).

Other status messages are optional.

Submission and Testing

Submit all source code and *all* files necessary to compile your project. If your project fails to compile, you will receive an initial grade of 0 points, but you may be approved for a second submission to rectify this.

Your program should compile and run on a standard campus Windows or Linux machine. The client will be tested with at least two test cases: one where the host is in the database and one where it is not. There may be additional tests that are run, but you can expect at least two.

Miscellaneous

You will be provided with a small plaintext DNS table. These should be included in your program in whatever method you choose.

You are granted a lot of freedom in the implementation of this project as long as the requirements are met.

Grading

This project will be out of 75 points.