This project is due at 11:59:59pm on November 21, 2011. You must complete it with a partner. You may only complete it in a group of three if you have the instructor’s explicit permission to do so for this project.

Note that there is a milestone deadline for this project, at 11:59:59pm on November 14th, 2011. More details are in the Milestone section below.

1 Description

The Domain Name System (DNS) is a hierarchical system for converting domain names (e.g., www.google.com) to Internet Protocol (IP) addresses (e.g., 209.85.129.99). DNS is often referred to as a “phone book” for the Internet, translating human-friendly domain names into machine-friendly IP addresses. In this project, you will implement a DNS client program, which handles DNS requests by querying other machines. Note that the graduate version of this project has additional requirements, which serve as an opportunity for extra credit for students enrolled in the undergraduate version of this course.

2 Requirements

Your will write a DNS client program which, given a name to query for and a DNS server to query will:

• Construct a DNS query packet for the specified name
• Send the query to the specified DNS server using UDP
• Wait for the response to be returned from the server
• Interpret the response and output the result to STDOUT

Your client must support the following features:

• Queries for A records (IP addresses)
• Responses that contain A records (IP addresses) and CNAMEs (DNS aliases)

You should be strict; if the returned message does not conform to the DNS specification, you should assert an error. You may receive other packets that are not responses to your query; you should ignore these and continue to wait for a response to your query. Remember that network-facing code should be written defensively. We will test your code by sending corrupted packets to your client; you should handle these errors gracefully and not crash.
3 Your client program

For this project, you may use any programming language of your choice. You may not use any DNS libraries in your project (e.g., getaddrinfo or gethostbyname in C). **You must construct the DNS request packet yourself, and interpret the reply yourself.**

The command line syntax for your client is given below. The client program takes command line argument of the domain name to interpret and the IP address of the domain server to query. The syntax for launching your program is therefore:

```
./3600dns @<server:port> <name>
```

**port** (Optional) The UDP port number of the DNS server. Default value: 53.

**server** (Required) The IP address of the DNS server, in a.b.c.d format.

**name** (Required) The name to query for.

After sending the request, your client should wait for a reply for 5 seconds. If no reply is heard within this time window, you should exit indicating that a timeout occurred (see below).

If you use C, your executable should be named 3600dns. If you use another language, you must write a brief Bash shell script, named 3600dns, that conforms to the input syntax above and then launches your program with whichever incantations are necessary. For example, if you write your solution in Java, your Bash script might resemble

```
#!/usr/bin/perl -w
$args = join(’ ’, @ARGV);
print ‘java -jar 3600dns.jar $args’;
```

You should develop your client program on the CCIS Linux machines, as these have the necessary compiler and library support. You are welcome to use your own Linux/OS X machines, but you are responsible for getting your code working, and your code must work when graded on the CCIS Linux machines. If you do not have a CCIS account, you should get one ASAP in order to complete the project.

If written in C, your code must be -Wall clean on gcc. Do not ask the TA for help on (or post to the forum) code that is not -Wall clean unless getting rid of the warning is what the problem is in the first place.

Your client must print results to standard using the following format:

```
IP <tab> <IP address> <tab> <seconds can be cached> <tab> <auth|nonauth>
CNAME <tab> <alias> <tab> <seconds can be cached> <tab> <auth|nonauth>
NOTFOUND
NORESPONSE
ERROR <tab> <description of the error>
```

If an response to a query contains multiple answers (such as multiple IP addresses or aliases), your client must print an IP or CNAME line for each one of these. If the requested name does not exist, your client must print a NOTFOUND line. If no response is ever received from the server (i.e., you’ve waited 5 seconds and not received anything), your client must print a NORESPONSE line. Finally, if any other error occurs, your client should print an ERROR line containing a description of the error.
4 Extra credit (15 points)

For extra credit, you can also support queries for **MX** (mail server) and **NS** (name server) records. Therefore, your program should accept the following input syntax:

```
./3600dns [-ns|-mx] @<server:port> <name>
```

where the optional `-ns` or `-mx` flags request their respective records (if no flag is given, you should query the A record). Your output for these records should look like

```
MX <tab> <alias> <tab> <preference> <tab> <seconds can be cached> <tab> <auth|nonauth>
NS <tab> <alias> <tab> <seconds can be cached> <tab> <auth|nonauth>
```

5 Testing your code

We have set up a test DNS server which you should use while developing your code: cs4700dns.ccs.neu.edu, 129.10.112.152. You should *not* send queries directly to any other DNS server (e.g., CCIS’s DNS servers) until your code is reliably working when sending to our test machine. Otherwise, the server operators may view your packets as a security attack, with undesirable consequences for all involved.

6 Submitting your project

6.1 Registering your team

You and your partner should first register as a team by running the `./course/cs3600f11/bin/register` script. You should pick out a team name (no spaces or non-alphanumeric characters, please) and run

```
./course/cs3600f11/bin/register project3 <teamname>
```

This will either report back success or will give you an error message. If you have trouble registering, please contact the course staff.

*You must register your team by 11:59:59pm on October 12th, 2011.*

6.2 Milestone

In order to ensure that you are making sufficient progress, you will have an interim milestone deadline. For the milestone, you must submit a draft of your README, which describes your overall approach. Your 3600dns must be able to create a DNS request, but does not have to be able to interpret the response. In other words, your code must correctly generate a packet for an input query, and send it to the server, but does not have to do any more processing.

You should submit your milestone by running the `./course/cs3600f11/bin/turnin` script. Specifically, you should create a `project3` directory, and place all of your code in it. Then, run

```
./course/cs3600f11/bin/turnin <teamname> project3-milestone <dir>
```
Where <teamname> is the team name you registered and <dir> is the name of the directory with your submission. The script will print out every file that you are submitting, make sure that it prints out all of the files you wish to submit!

You must submit your milestone by 11:59:59pm on November 14th, 2011. No slip days can be used on milestone 1.

6.3 Final submission

For the final submission, you should submit you (thoroughly documented) code along with a plain-text (no Word or PDF) README file. In this file, you should describe your high-level approach, the challenges you faced, a list of properties/features of your design that you think is good, and an overview of how you tested your code. You should also describe the disk layout that you chose, and why you made that choice.

You should submit your project by running the /course/cs3600f11/bin/turnin script. Specifically, you should create a project3 directory, and place all of your code and README files in it.

Then, run

/course/cs3600f11/bin/turnin <teamname> project3 <dir>

Where <teamname> is the team name you registered and <dir> is the name of the directory with your submission. Again, the script will print out every file that you are submitting, make sure that it prints out all of the files you wish to submit!

You must submit your project by 11:59:59pm on November 21st, 2011.

7 Grading

The grading in this project will consist of

- 60% Program functionality
- 15% Error handling
- 15% Style and documentation
- 10% Milestone functionality

You are, however, going to be graded on how gracefully you handle errors. In other words, what will you do if you receive a corrupted response packet? Remember, network-facing code should be graded defensively; you should always assume that everyone is trying to break your program. To paraphrase John F. Woods, “Always code as if the [the remote machine you’re communicating with] will be a violent psychopath who knows where you live.”

8 Advice

A few pointers that you may find useful while working on this project:
• Remember to convert your integers, shorts, and longs to network ordering (using `hton()` and associated functions).

• You should check the output of your program versus other DNS utilities. On Linux/OS X/UNIX, you can use the `dig` program, which outputs a fair amount of debug information about the requests and responses.

• You should also check out the Wireshark (formerly Ethereal) program, which has a very nice protocol debugger built in. It can be configured to listen to the packet you send out/receive back, and help you figure out why your packet is incorrectly formatted.

• Check the Blackboard forum for question and clarifications. You should post project-specific questions there first, before emailing the course staff.

• Finally, get started early and come to the TA lab hours - these are held from 2:00pm - 4:00pm on Tuesdays in the lab at 102 West Village H. You are welcome to come to the lab and work, and ask the TA any questions you may have.