

UDP Programming Project

*Assigned: November 20**Due: December 4, midnight*

Overview

For this assignment, you will write a client that issues queries to a Domain Name System (DNS) server using the UDP socket interface. You will not write a server, instead you will make use of an existing DNS server for interaction. This makes your job both harder and easier — you have less code to write, however when debugging you will not be able to “see” what is happening at the server. Your client will take a host name as input, form a correct DNS query packet, connect to a DNS server, receive a DNS reply packet, and extract and print the IP address from the reply.

Resources

In addition to the resources used in the TCP programming assignment, you will need to fully understand the format of DNS queries and responses. DNS is described in the Leon-Garcia book.

The text may not contain enough detail for you to be able to complete the assignment. I will make available the description of DNS from the W. Richard Steven’s textbook “TCP/IP Illustrated, Volume 1”. I’ll have copies available in class on Tuesday. Pay particular attention to the encoding of Internet host names (p. 192, 193) and the use of compression in the DNS responses (p. 197). You will obviously need to encode the host name correctly in the query, and you will need to determine whether compression is in use in order to parse the reply packet.

For those who prefer a truly authoritative and detailed source of information, RFC 1034 (Domain Names—Concepts and Facilities) and RFC 1035 (Domain Names—Implementation and Specification) are the place to look.

Details

Your client will communicate with the DNS server using the datagram service provided by UDP. Thus the socket type is `SOCK_DGRAM` and the family is `AF_INET`. The DNS server listens for requests on the well-known port number 53.

The DNS server your program will interact with is:

```
lennon.cc.gatech.edu
IP address: 130.207.114.10
```

As is described in the Steven’s text, a DNS query consists of a 12-byte DNS header followed by four variable length sections (questions, answers, authority, and additional information). Your client will ask one question: the IP address corresponding to a single host name. You should set the `identification` field to any value desired (this allows DNS responses to be matched up with queries), the `flags` field to hexadecimal 256 (to indicate a query with recursion desired), the `number of questions` field to hexadecimal 0001, and the other header fields to 0.

You should fill in the question section with your single query, encoding the host name as described in the text. Your `query type` should indicate that you want an A record, corresponding to the IP address. The `query class` is 1, for Internet addresses. **Do not forget that multi-byte fields must be converted to network byte order before transmission.**

The response you receive will have the same packet header format as your query (though obviously the content will differ). It may repeat your question in the questions section. If your query is properly formatted, it should contain the answer in the answers section, in the form of a resource record. It may also contain some authority resource records.

Do not forget that multi-byte fields must be converted to host byte order before parsing. You should parse the content of the answers section to determine the IP address for your query. Be careful to check if compression is used, which affects the encoding of the answer. You should extract and print the IP address (making sure it is for the host name you provided). Also print the TTL associated with the answer.

Your code should be well-written. For example, you should avoid hardcoding constants, and instead define constants in a header file that you include in your client and server programs. You should also check the return values of all system calls and do something appropriate if the return value indicates an error has occurred.

You are encouraged to help one another via the newsgroup, with questions that deal with the language or infrastructure. For example, “how do I compile with library X” or “why does the compiler complain about a type mismatch in this fragment of code...?” or “how do I access the third byte of an int?”. However, you are to otherwise work in your group of one or two on the code.