Programming Assignment 3: Simplified DNS Client and Server

Using UDP sockets, you will write a simplified version of a DNS client and server. The server will be responsible for the domain "student.test". The client will send a request to the server to look up the IP address of a specified host in that domain, and the server responds with the type A resource record associated with the host. The client and server communicate using the message format specified in this document.

The client will perform the following functions:

1. Read in 3 arguments from the command line:
    a. IP address of server (127.0.0.1)
    b. Port of server (e.g. 9999)
    c. Hostname (e.g. host1.student.net)
2. Send a request with the specified hostname to the server using the message format specified
3. Wait for a response using a 1 second timeout period.
    a. If a response arrives within the timeout period, print out the server response as shown in this document
    b. If not, re-send the message (same sequence number) for a maximum of 3 attempts before printing an applicable message and exiting

The server will perform the following functions:

1. Read in 2 arguments from the command line:
    a. IP address of server ((127.0.0.1)
    b. Port of server (e.g. 9999)
2. Read in the master file named "dns-master.txt" (See "dns-master.txt" for format and example. OK to have this filename hard-coded.)
3. Store the resource records (type A) in data structures in main memory suitable for searching
4. Respond to requests from the DNS client for hostnames in the domain using the message format specified
5. Return an error if the name queried does not exist in the domain.
6. The program should still work if the master file is modified to include different hostnames, or IP addresses

<u>Simplifying Assumptions:</u>

- Only one question in question section
- Only one answer in answer section (one IP address per host)
- Class is always of type Internet (IN)
- Only type A resource records

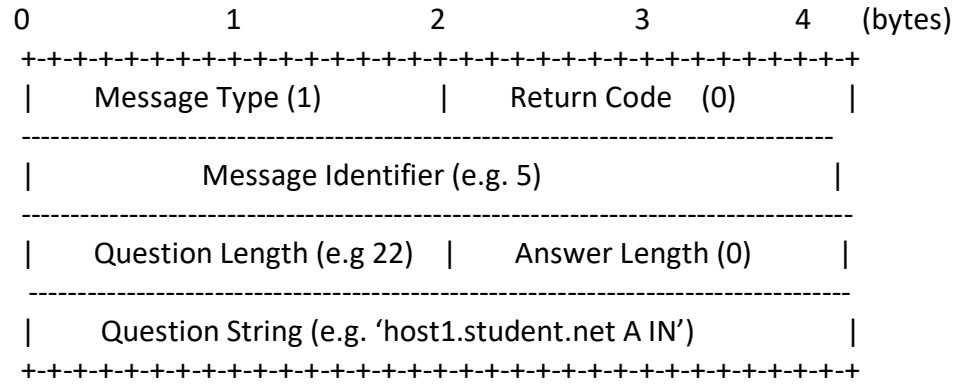
<u>Test Cases:</u>

All DNS messages must adhere to the DNS Message Format specified in this document:

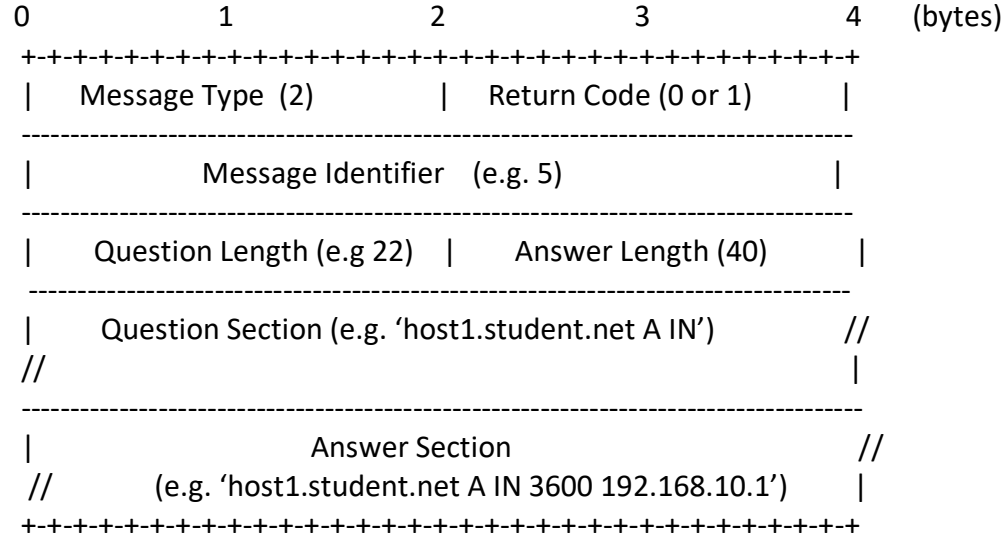
1. Look up existing name
2. Look up non-existent name
3. Look up name when server not running (On Windows, run the server but comment out responses)

Message Format

In a DNS request, the application data has the following format:



In a DNS response, the application data has the following format:



Message Type (16 bits): 1 on request; 2 on response
Return Code (16 bits) : 0 on request ; in response, 0 if name found, 1 if name does not exist
Message Identifier (32 bits): Uniquely identifies a message in a request, server echoes same number back in response. Should be generated randomly in range between 1 and 100
Question Length (16 bits): In request and response, length of resource record in Question Section in bytes.
Answer Length (16 bits): 0 in request (because no answer section). In response, length of resource record in Answer Section in bytes.
Question Section (variable length): In request, String carrying Question in the form of a DNS resource record containing hostname, 'A' and 'IN' separated by a single space e.g. 'host1.student.net A IN'); Echoed back in server response

Answer Section (variable length): In request, there is no answer section. In response, the server includes the DNS record containing the IP address of the hostname in the request. e.g. 'host1.student.net A IN 3600 192.168.10.1'). Empty, if not known.
Resource Record: Resource records in the Answer Section must contain 5 pieces of information in the order specified: full hostname (e.g. host1.student.net) , type (A), class(IN), TTL in seconds (e.g. 3600) and IP address in dotted-decimal notation (e.g. 192.168.10.1). The information is represented in a single string with items separated by a single space character e.g. "host1.student.net A IN 3600 192.168.10.1". Resource records in the Question Section are a string containing only the first 3 items in the order specified e.g. "host1.student.net A IN".

Test Output

1. Test Case 1: Client Output Example (Name found)

```
Sending Request to 127.0.0.1, 9999:
Message ID:   57
Question Length: 22 bytes
Answer Length: 0 bytes
Question: host1.student.net A IN

Received Response from 127.0.0.1, 9999:
Return Code: 0 (No errors )
Message ID:   57
Question Length: 22 bytes
Answer Length: 40 bytes
Question: host1.student.net A IN
Answer: host1.student.net A IN 3600 192.168.10.1
```

2. Test Case 2: Client Output Example (Name not found)

```
Sending Request to 127.0.0.1, 9999:
Message ID:   93
Question Length: 31 bytes
Answer Length: 0 bytes
Question: host-not-exist.student.net A IN

Received Response from 127.0.0.1, 9999:
Return Code: 1 (Name does not exist)
Message ID:   93
Question Length: 31 bytes
Answer Length: 0 bytes
Question: host-not-exist.student.net A IN
```

3. Test Case 3: Client Output Example (Server does not respond)

```
Sending Request to 127.0.0.1, 9999:
Message ID:   32
Question Length: 22 bytes
Answer Length: 0 bytes
Question: host3.student.net A IN

Request timed out …
Sending Request to 127.0.0.1, 9999:
```

```
Request timed out …
Sending Request to 127.0.0.1, 9999:
Request timed out … Exiting Program.
```

Submission Instructions:

Please submit the following individual files to Canvas by due date. **Please, NO zip files.**

- ✓ Submit the client and server source program files (please name the file dns*client.py* and dns*server.py* respectively and include name, UCID, section in comments at top of source files)
- ✓ Submit a single wireshark .pcap file captured while running all three test cases of the program in sequence (please name *dns.pcap*)
- ✓ Submit screenshots in .pdf format showing the trace output of the client and server
- ✓ Submit the README file (C programs only; No need for README for Python programs)

NOTE: Please do not hard-code pathnames to files in your programs. Your program MUST be portable. (You may use the pathlib library to open files in the current working directory.)

Grading: Accounts for 10% of the overall grade**.**

Late Submission Policy: 10% penalty for each day late until the cut-off date.

Academic Integrity Policy:

If academic integrity standards are not upheld, no credit is given. This includes copying of program or wireshark lab or .pcap file from any source, or hard-coding of results in your program.

References:

- Python
  - https://docs.python.org/3/library/socket.html
  - https://docs.python.org/3/library/struct.html
  - https://docs.python.org/3/library/random.html
  - https://docs.python.org/3/library/pathlib.html