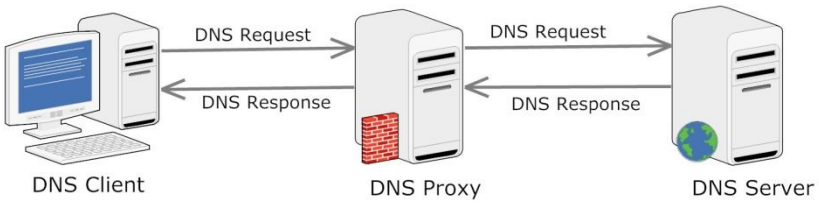
Multi-stage DNS Resolving System using Client-Server socket programming

In this assignment, you require implementing three C programs, namely Client, Proxy Server (which will act both as client and server) and DNS Server and they communicate with each other based on TCP sockets. The aim is to implement a simple 2 stage DNS Resolver System.



Initially, the client will connect to the proxy server using the server's TCP port already known to the client. After successful connection, the client sends a Request Message (Type 1/Type 2) to the proxy server. The proxy server has a limited cache (assume a cache with three IP to Domain_Name mapping entries only). After receiving the Request Message, proxy server based on the Request Type (Type 1/Type 2) searches its cache for corresponding match. If match is successful, it will send the response to the client using a Response Message. Otherwise, the proxy server will connect to the DNS Server using a TCP port already known to the Proxy server and send a Request Message (same as the client). The DNS server has a database (say .txt file) with it containing set of Domain_name to IP_Address mappings. Once the DNS Server receives the Request Message from proxy server, it searches in its file for possible match and sends a Response Message (Type 3/ Type 4) to the proxy server. On receiving the Response Message from DNS Server, the proxy server forwards the response back to the client. If the Response Message type is 3, then the proxy server must update its cache with the fresh information using FIFO scheme. After each negotiation phase, the TCP connection on both sides should be closed gracefully releasing the socket resource.

Request Message Format:

Request_Type	Message

- Type 1: Message field contains Domain Name and requests for corresponding IP address.
- Type 2: Message field contains IP address and request for the corresponding Domain Name.

Response Message Format:

Response_Type	Message

- Type 3: Message field contains Domain Name/IP address.
- Type 4: Message field contains error message "entry not found in the database".

You should accept the IP Address and Port number from the command line (Don't use a hard-coded port number). Prototype for command line is as follows:

Prototypes for Client and Server
Client: <executable code><Server IP Address><Server Port number>
Server: <executable code><Server Port number>

*Please make necessary and valid assumptions whenever required.