Experiment 9: Simple DNS Server

Implement a simple local DNS server called LocalDNSServer. LocalDNSServer accepts queries from clients and tries to resolve them, first by checking its local cache and in case of a miss, by asking other name servers. LocalDNSServer must reply to A, AAAA, NS, and CNAME requests, and it may drop queries of all other types. Before you start to implement your server, you should have a clear understanding of the DNS protocol and, specifically, of the message format.

LocalDNSServer must be able to resolve queries iteratively starting from a given root name server. In other words, LocalDNSServermust work even with a root server that does not support recursive queries. So, to respond to a particular query, the server might have to query multiple servers, starting from the given root server for the root domain, and then iterating through the relevant authoritative name servers for each specific subdomain. For example, the root, then “in”, then “ac” and then “nitc” for “www.nitc.ac.in”. LocalDNSServermust also correctly handle canonical names. This means that a server (the root server or others) might reply with a CNAME record in response to a request for the A record for a particular name. In this case, LocalDNSServer should first resolve the CNAME record and then reply to the original request with a DNS message that includes both the canonical-name record (CNAME) for the original name and the address record (A) for the canonical name. LocalDNSServer must implement and use an internal cache. For each client request and also for each intermediate request (e.g., a request for the “in.” domain), LocalDNSServer must first consult its cache and use a valid cached record if one exists. LocalDNSServer would then issue a query to an external server only if the needed record is not in the cache, or if the one in the cache is no longer valid. The validity of a record is determined by the TTL value for that record. LocalDNSServer must then insert every fresh record received from an external server in its cache. LocalDNSServer does that by either replacing the same (invalid) record in the cache, or by creating a new record in the cache if the cache is not full, or by evicting and replacing another existing record. In this latter case, LocalDNSServer must replace any invalid record in the cache before replacing a valid one. In order to process DNS queries and replies, LocalDNSServer must be able to parse DNS messages.

Students are allowed to use the inbuilt library by calling nslookup recursively, with DNS Server as the public DNS Server hosted by google, from their c program for recursively resolving the Domain.

LocalDNSServer must accept two command-line arguments: the UDP port to listen on, and the IP address of a root DNS server. For example: *LocalDNSServer 1234 8.8.8.8* You can easily test your implementation using the *nslookup* command with the proper arguments, by giving port number as 1234 and the DNS Servers IP address as 127.0.0.1.

Deadline: 05 April 2021 (1.00 pm)
Evaluation mode: Program will be given based on the experiment