CS352 - Project 1 Report

Arun Felix and Bhavya Patel

March 16, 2025

1 Team Details

- Arun Felix (netid: ajf277)
- Bhavya Patel (netid: bsp75)

2 Collaboration

We used the Python documentation, and Bhavya focused on the first half. Arun did the last half and developed testcases.

3 Implementation of recursive and iterative query functionality

In this project, recursive resolution queries are implemented by having the client send a recursive query to the root server (RS), which then processes the query to determine if the domain is in its domain map. If not, the RS identifies the appropriate top-level domain (TLD) server (TS1 or TS2) based on the domain's TLD and forwards the query to that server using a recursive connection. The TLD server processes the query and sends back the resolved IP address or an indication that the domain is not found. The RS then forwards this response back to the client, completing the recursive resolution process.

3.1 Iterative Query Implementation

The iterative query functionality in this project works by having the client first send a request to the Root Server (RS) with the "it" flag. When the RS receives an iterative query, it checks if the domain exists in its own database. If found, it directly returns the IP with an "aa" (authoritative answer) flag. If not found, instead of recursively forwarding the query to the appropriate Top-Level Domain server (TS1 or TS2) as in the recursive mode, it responds with the IP address of the appropriate TS server and a "ns" (name server) flag. The client then handles this response by establishing a new connection directly to the indicated

TS server and sending a second query. The client's *handleit*() function manages this two-step process: first querying the RS server, then examining the response to determine if a follow-up query to a TS server is needed.

4 Functionality

No, everything works as expected and as is detailed within the instructions.

5 Difficulties

We encountered significant challenges with socket connections while implementing this DNS simulation system. The most persistent issue was managing connection states, particularly when dealing with parallel connections and connection lifetimes. We struggled with connections timing out or being refused when trying to establish multiple connections to different servers simultaneously. When implementing the iterative query mode, we had to be careful about properly opening and closing sockets between the client and various servers to prevent resource leaks. Additionally, in the recursive implementation, the Root Server needed to maintain its connection with the client while simultaneously establishing new connections to the TS servers, which created complexity in connection management. We had to implement careful error handling and connection cleanup to ensure that sockets were properly closed even when errors occurred, preventing resource exhaustion. These networking challenges highlighted the complexities of distributed systems communication where timing and connection state management are critical factors.

6 What did we learn?

- **Two DNS Resolution Types:**
  - Implemented both **recursive** and **iterative** resolution types in our DNS simulation
  - Recursive: Client sends one query, Root Server handles all communication with other servers
  - Iterative: Client must make multiple queries to different servers to resolve a domain
- **Three-Server Architecture:**
  - Created a system with a Root Server (RS) and two Top-Level Domain servers (TS1, TS2)
  - RS served as the first point of contact in the DNS resolution process
  - TS1 and TS2 served as specialized domain servers for different domains

- **Socket Connection Management:**
  - Learned to establish and maintain reliable TCP connections between multiple servers
  - Handled challenges with connection timing and preventing connection refusals
  - Implemented proper error handling for socket connection issues