Back to Course

Project 3

PYTHON GROUP

Details Exploit Fix

Assignment Details

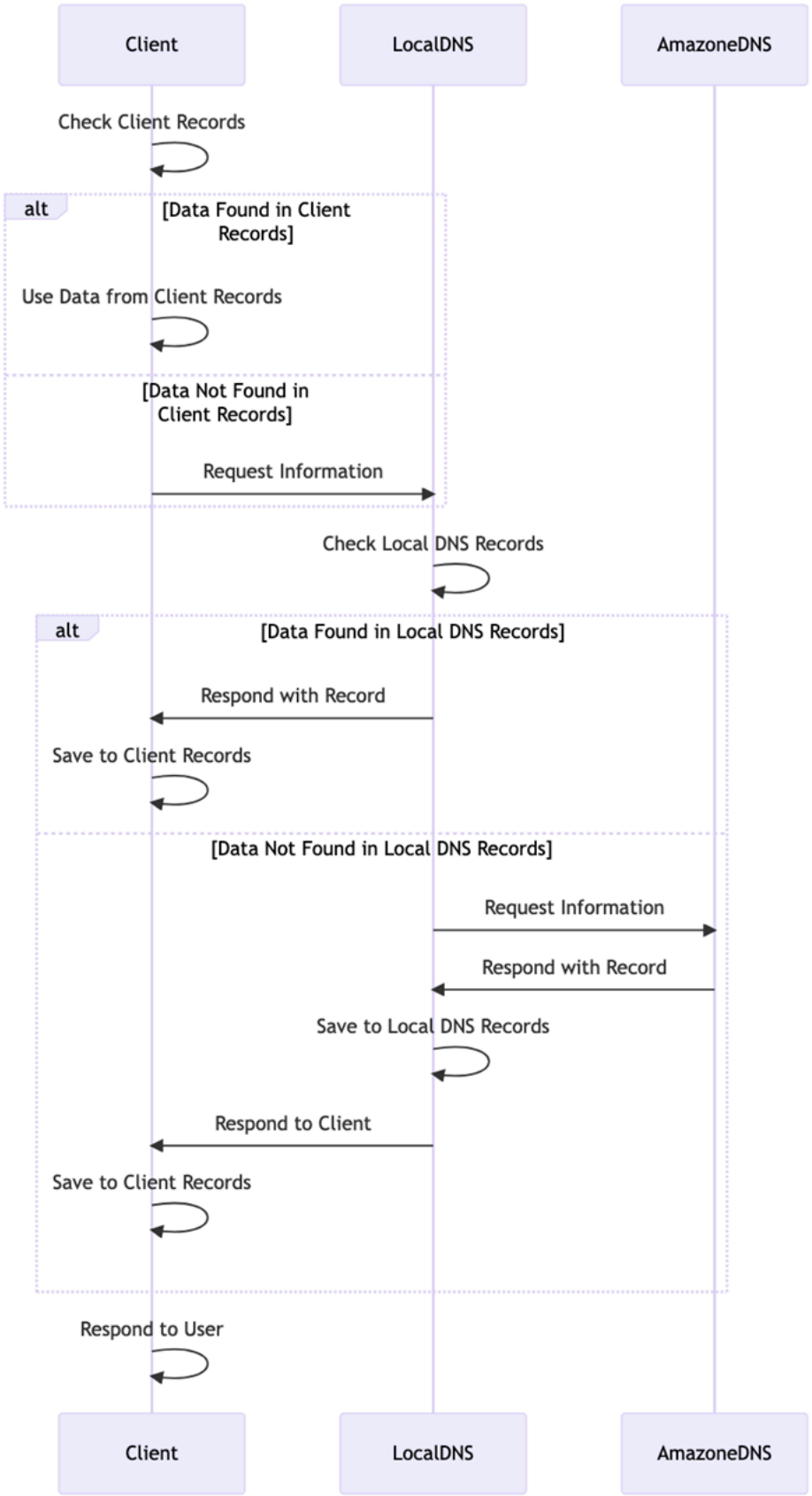
In this project, you will develop a console-based DNS client and two DNS server programs to mimic a simplified DNS system. Each of your program will maintain its own resource record (RR) table, which is used to store DNS records. To send and receive records between programs, you could use UDP sockets. Here are the details for each program:

Role	Program name	Port number
DNS Client	client.py	
Local DNS Server	localserver.py	21000
Amazone DNS Server	amazoneserver.py	22000

DNS Client: The DNS client program allows users to look up info about hostnames/domains. After receiving a hostname from users, it first checks its own RR table. If it's not found, it asks the local DNS server. When it gets a response, it saves the result/record in its RR table and then prints out its RR table.

Local DNS Server: The local DNS server handles requests from the client. It first checks its own RR table for the request. If it's not found, it asks the authoritative DNS server for the requested hostname/domain. When it gets a response, the local DNS server saves the result/record in its RR table and sends it to the client. Then, it prints out its RR table.

Amazone DNS Server: The Amazone DNS server handles requests from the local DNS server for Amazone-related requests. This means it's an authoritative DNS server for Amazone. It checks its own RR table for the request and sends the result/record to the local DNS server. Then, it prints out its RR table.



In this project, we assume the user is at CSUSM campus. This means the local DNS server is also the authoritative DNS server for CSUSM.

The Resource Record (RR) table contains 6 fields:

1. **Record number**: An integer starting from 0 and incrementing by 1 for each new record.
2. **Name**: the hostname or domain name.

3. **Type**: one of the 4 types (A, AAAA, CNAME, NS).
4. **Result**: the corresponding data for the given name and type.
5. **TTL (Time To Live)**: An integer indicating for how many more seconds this record will remain valid and stay in the RR table.
6. **Static**: This field indicates whether this record is Static (1: manually entered and remains valid) or Dynamic (0: automatically entered by the DNS program and will remain valid until the TTL reaches 0). In this project, this field is 0 for all entries in the client's RR table.

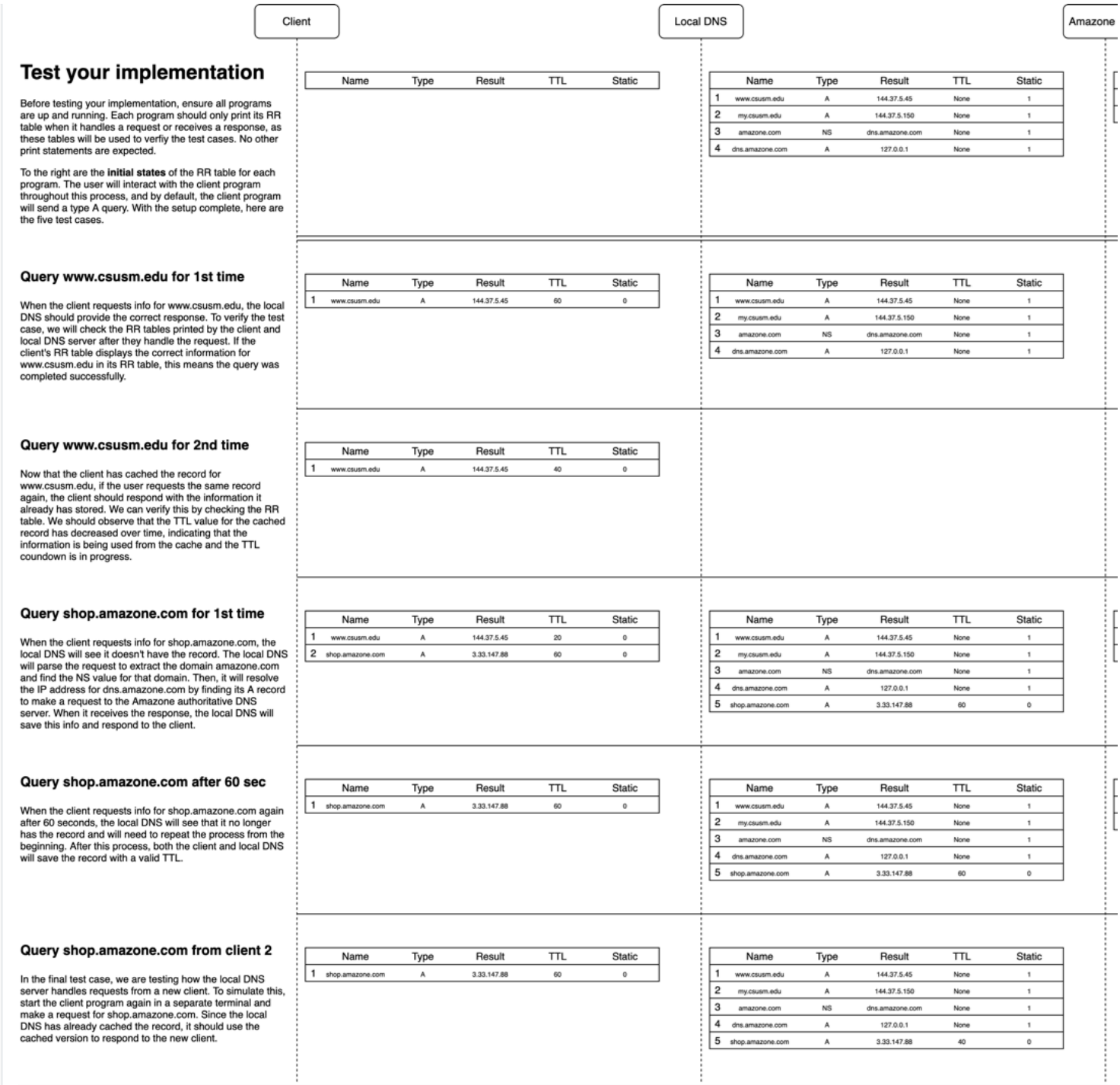
The initial value for TTL should be 60 seconds. Every second, both the client and local DNS server decrement the TTL fields of all Dynamic records on their RR tables. Once the TTL reaches 0, the record will be removed from the RR table. Note that the TTL field of Static fields is always None. Each time a program responds to a request or receives a response, it will print its current RR table on the screen. This should be a comma separated table like the following example:

- record_no,name,type,result,ttl,static
- 0,www.csusm.edu,A,144.37.5.45,None,1

The structure of DNS query and response contain the following fields.

1. **Transaction ID (32 bits)**: The host assigns a unique Transaction ID to each DNS query, and temporarily stores the query until it receives a response for it. The response should contain the same ID. When a host receives a DNS response, it first checks whether its ID matches with the sent query and also the flags match the sent request. If yes, it removes the temporarily stored request and stores the response in the RR table. We assume this field is a 4-byte (32-bit) integer stored in binary format. Start from 0 and increment by 1 for each new query.
2. **Flag (4 bits)**: Query (0000) or Response (0001).
3. Question:
    - **Name**: A string containing the requested name.
    - **Type (4 bits)**: A (1000), AAAA (0100), CNAME (0010), NS (0001).
4. Answer:
    - **Name**: A string containing the requested name.
    - **Type (4 bits)**: A (1000), AAAA (0100), CNAME (0010), NS (0001).
    - **TTL**: An integer indicating for how many more seconds this record will remain valid and stay in the RR table.
    - **Result**: A string containing the requested data.

If a record is not found, add "Record not found" in the result field. The program receiving the response should handle this case by not including the record in its RR table. The following shows the details of **five test cases** and the expected result for each test case.



Extra credit (10 points): There are multiple types of DNS queries, e.g., A, AAAA, CNAME and NS. In this project only type A (IPv4) DNS query is required. For extra credit, you can add support for AAAA, CNAME, and NS query types. Users can specify the query type by adding it after the hostname/domain, separated by a space. If a user only provides the hostname, it should default to sending an A record query type. The input prompt should be as follows:

Enter the hostname (or type 'quit' to exit) <hostname> <query type>

Note that the text <hostname> and <query type> are placeholders to show the order in which users can enter data and should not be typed out literally in the prompt.

Submit Code

Group 1