Experiment No.: 08

Title: Develop a DNS Client–Server to resolve the given host name or IP address

1. Objective

- To understand the working of **DNS (Domain Name System)** and its client-server architecture.
- To **develop a DNS client and server program** that resolves:
    - a **hostname** to its **IP address**, and
    - an **IP address** to its **hostname** (reverse lookup).
    -
- To study the **communication process at the application layer** using socket programming.

2. Theory

- **DNS (Domain Name System)** is a distributed hierarchical database that translates **domain names** (like www.example.com) into **IP addresses** (like 93.184.216.34).
- It operates primarily on **UDP port 53** (and TCP port 53 for large responses).
- **DNS Server:** Maintains a mapping between domain names and IP addresses.
- **DNS Client (Resolver):** Queries the DNS server for the IP or hostname.

The DNS resolution process involves:

1. A client sends a query to the DNS server.
2. The server checks its database (or cache) for the requested mapping.
3. The server sends back the corresponding IP address (for hostname) or hostname (for IP).

3. Software & Tools Required

- Programming Language: **Python 3** (or C if preferred)
- Tools: Any IDE or terminal (e.g., VS Code, Linux shell)
- Libraries: socket, json (for Python)

4. Algorithm

Server:

1. Start a UDP socket on a specific port (e.g., 12345).
2. Create a dictionary containing hostname–IP mappings.
3. Wait for the client to send a query (hostname or IP).
4. Check the query in the dictionary (forward and reverse).
5. Send the resolved IP or hostname back to the client.
6. If not found, send an error message.

Client:

1. Create a UDP socket.
2. Take input from user — hostname or IP.
3. Send the query to the server.
4. Wait for the response.
5. Display the resolved result.

5. Program

(A) DNS Server – dns_server.py

```
import socket

# DNS mapping (static)
dns_table = {
    "www.google.com": "142.250.190.68",
    "www.yahoo.com": "98.137.11.163",
    "www.dypitpune.edu.in": "192.168.1.100",
    "192.168.1.100": "www.dypitpune.edu.in",
    "142.250.190.68": "www.google.com"
}

# Create UDP socket
server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server.bind(('localhost', 12345))
print("DNS Server started... waiting for queries")

while True:
    data, addr = server.recvfrom(1024)
    query = data.decode()
    print(f"Received query: {query}")

    # Resolve query
    response = dns_table.get(query, "No record found")
    server.sendto(response.encode(), addr)
    print(f"Sent response: {response}\n")
```

(B) DNS Client – dns_client.py

```
import socket



# Create UDP socket

client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```
server_address = ('localhost', 12345)

while True:
    query = input("\nEnter Hostname or IP (or 'exit' to quit): ")
    if query.lower() == 'exit':
        break

    # Send query to server
    client.sendto(query.encode(), server_address)

    # Receive response
    data, _ = client.recvfrom(1024)
    print("Resolved Address:", data.decode())

client.close()
```

Sample Output

Server Side:

```
DNS Server started... waiting for queries
Received query: www.google.com
Sent response: 142.250.190.68
Received query: 192.168.1.100
Sent response: www.dypitpune.edu.in
```

Client Side :

```
Enter Hostname or IP (or 'exit' to quit): www.google.com
Resolved Address: 142.250.190.68
Enter Hostname or IP (or 'exit' to quit): 192.168.1.100
Resolved Address: www.dypitpune.edu.in
```

```
Enter Hostname or IP (or 'exit' to quit): www.unknown.com
Resolved Address:
```

Result

A DNS client-server system was successfully developed using Python sockets.

The system was able to resolve hostnames to IP addresses and vice versa.

The experiment demonstrated application-layer communication and name resolution in a simplified DNS setup. No record found