EXPERIMENT: 28

DEVELOPING A CLIENT THAT CONTACTS A GIVEN DNS SERVER TO RESOLVE A GIVEN HOSTNAME in JAVA/C

Aim: To develop a client that contacts a given DNS server to resolve a given hostname.

Description: · Get the host name to be resolve using gethostname() · Check the host name using nslookup · Print the IP address, host name, Address length and Address type. · List the addresses stored in lookup

Algorithm :

Step 1. Find the host name by using get host by name()

Step 2. The host name is followed by the list of alias names

Step 3. Pointer points to the array of pointers to the individual address

Step 4. For each address call the inet_ntop() and print the returned string

Procedure:

1. Initialize necessary variables and structures:

- Create a socket using the `socket()` function with the `AF_INET` address family and `SOCK_DGRAM` socket type.

- Set the DNS server address and port in a `struct sock addr_in` structure.

- Prepare a DNS query packet with the desired hostname and record type.

2. Send the DNS query to the DNS server:

- Use the `send to()` function to send the DNS query to the DNS server using the socket descriptor and the server address structure.

3. Receive the DNS response from the DNS server:

- Use the `recvfrom()` function to receive the DNS response from the DNS server using the socket descriptor.

- Store the response in a buffer.

4. Parse and process the DNS response:

- Extract the necessary information from the DNS response packet based on the DNS protocol.

- Handle any error conditions, such as a non-existent hostname or failed resolution.

5. Display or use the resolved information:

- Extract the resolved IP address or other relevant data from the DNS response.

- Process and utilize the resolved information as needed in your application.

6. Close the socket:

55

- Use the `close()` function to close the socket descriptor.

Remember to include the necessary header files (`<stdio.h>`, `<stdlib.h>`, `<string.h>`,

`<sys/socket.h>`, `<netinet/in.h>`, etc.) and handle errors appropriately in your code. Additionally,

you may need to implement DNS-specific functions for packet parsing and response handling based

on the DNS protocol.

It's important to note that DNS resolution involves dealing with DNS protocol intricacies, including

packet format, header fields, and query/response structure. Familiarize with the DNS protocol

specifications to ensure correct handling of DNS messages.

```
PS E:\studies\CN\practicals\source files\expt28> .\dns_client_real.exe
Enter hostname to resolve: saveetha.com
Resolved IP: 198.185.159.145
PS E:\studies\CN\practicals\source files\expt28> .\dns_client_real.exe
Enter hostname to resolve: google.com
Resolved IP: 142.251.223.238
PS E:\studies\CN\practicals\source files\expt28> .\dns_client_real.exe
Enter hostname to resolve: youtube.com
Resolved IP: 142.250.193.174
PS E:\studies\CN\practicals\source files\expt28>
```

Result: Thus a client that contacts a given DNS server to resolve a given hostname is developed

successfully