EXPERIMENT: 27

IMPLEMENTATION OF A DNS SERVER AND CLIENT IN C USING UDP

SOCKET

Aim: To implement a DNS server and client in java using UDP socket

Algorithm:

Server

1. Create an array of hosts and its IP address in another array
2. Create a datagram socket and bind it to a port
3. Create a datagram packet to receive client request
4. Read the domain name from client to be resolved
5. Lookup the host array for the domain name
6. If found then retrieve corresponding address
7. Create a datagram packet and send ip address to client
8. Repeat steps 3-7 to resolve further requests from clients
9. Close the server socket
10. Stop

Client

1. Create a datagram socket
2. Get domain name from user
3. Create a datagram packet and send domain name to the server
4. Create a datagram packet to receive server message
5. Read server's response
6. If ip address then display it else display "Domain does not exist"
7. Close the client socket

Procedure:

DNS Server-side implementation:

1. Create a UDP socket using the `socket()` function with the `AF_INET` address family and `SOCK_DGRAM` socket type.
2. Set socket options using the `setsockopt()` function to allow reuse of the address and port.
3. Bind the socket to a specific IP address and port using the `bind()` function.
4. Receive a DNS query from a client using the `recvfrom()` function.

5. Parse the DNS query to extract the requested domain name and record type.

6. Lookup the requested domain name and retrieve the corresponding IP address or other records.

7. Create a DNS response packet with the appropriate format.

8. Send the DNS response to the client using the `sendto()` function with the client address and port obtained from `recvfrom()`.

9. Close the socket using the `close()` function.

DNS Client-side implementation:

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

2. Set the server address and port in a `struct sockaddr_in` structure.

3. Prepare a DNS query packet with the desired domain name and record type.

4. Send the DNS query to the server using the `sendto()` function with the server address and port.

5. Receive the DNS response from the server using the `recvfrom()` function.

6. Parse the DNS response packet to extract the requested information.

7. Process and display the received information as needed.

8. Close the socket using the `close()` function.

DNS Server-side implementation:

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

2. Set socket options using the `setsockopt()` function to allow reuse of the address and port.

3. Bind the socket to a specific IP address and port using the `bind()` function.

4. Receive a DNS query from a client using the `recvfrom()` function.

5. Parse the DNS query to extract the requested domain name and record type.

6. Lookup the requested domain name and retrieve the corresponding IP address or other records.

7. Create a DNS response packet with the appropriate format.

8. Send the DNS response to the client using the `sendto()` function with the client address and port obtained from `recvfrom()`.

9. Close the socket using the `close()` function.

DNS Client-side implementation:

1. Create a UDP socket using the `socket()` function with the `AF_INET` address family and

`SOCK_DGRAM` socket type.

2. Set the server address and port in a `struct sockaddr_in` structure.

3. Prepare a DNS query packet with the desired domain name and record type.

4. Send the DNS query to the server using the `sendto()` function with the server address and port.

5. Receive the DNS response from the server using the `recvfrom()` function.

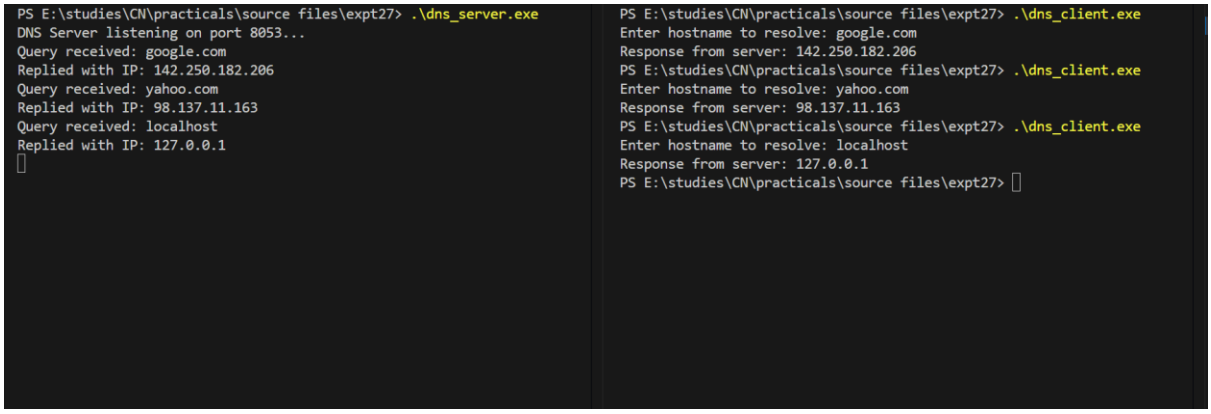
6. Parse the DNS response packet to extract the requested information.

7. Process and display the received information as needed.

8. Close the socket using the `close()` function.

Note: 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, DNS lookup, and response creation.

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, DNS lookup, and response creation.



Result: Thus a DNS server and client in java using UDP socket is implemented successfully