SIT 202 – COMPUTER NETWORKS AND COMMUNICATION

DNS SERVER

Summary of the Pseudocode and Implementation

Pseudocode Overview:

1. Server Initialization:

`initialize_server()`: Creates a UDP socket, binds it to port 53, and initializes DNS records.

`create_udp_socket()`: Creates a UDP socket.

`bind_socket(socket, port)`: Binds the socket to the specified port.

`load_dns_records()`: Loads or initializes A and CNAME DNS records.

2. Listening and Processing DNS Queries:

`listen_for_queries(socket)': Continuously listens for queries, parses them, and directs them to the appropriate handler based on the query type.

`receive_query(socket)`: Receives data from the client.

`parse_query(query)`: Decodes the query to extract hostname and query type.

3. Handling A and CNAME Records:

`handle_a_record_query(hostname, client_address, socket)`: Looks up and responds with A record.

`handle_cname_record_query(hostname, client_address, socket)`: Looks up and responds with CNAME record.

lookup_a_record(hostname)`: Retrieves the A record from the DNS records.

`lookup_cname_record(hostname)`: Retrieves the CNAME record from the DNS records.

4. Generating DNS Responses:

`generate_response(hostname, record_data, query_type)`: Creates a DNS response message.

- `create_dns_response()`: Initializes a DNS response object.

`send_response(response, client_address, socket)`: Sends the response to the client.

`send_data(socket, response, client_address)`: Sends data over the network.

5. Main Function:

`main()`: Initializes the server and starts processing queries.

Python Implementation:

1. DNS Server (`dns_server.py`):

- Creates a UDP socket, binds it to port 53.
- Listens for incoming queries.
- Parses and processes `A` and `CNAME` queries.
- Responds with predefined DNS records or error messages.

2. DNS Client (`dns_client.py`):

- Sends a DNS query (either `A` or `CNAME`) to the server.
- Prints the server's response.

The DNS server and client implementation helps by:

1. **Understanding DNS**: Provides hands-on experience with DNS operations, including handling A and CNAME queries.
2. **Networking Skills**: Offers practical experience with UDP communication and socket programming.
3. **Server-Client Architecture**: Teaches how to set up and interact with server-client models.
4. **Problem-Solving**: Enhances skills in debugging network issues and handling errors.
5. **Real-World Applications**: Lays the foundation for more advanced networking topics and practical applications.
6. **Educational Value**: Bridges theoretical concepts with practical implementation, reinforcing learning through hands-on experience.

Reflections:

1. What is the most important thing you learned in this module?

Understanding of DNS Protocols: The most important lesson is how DNS queries and responses work, including how DNS servers handle different types of records and how they interact with clients over UDP.

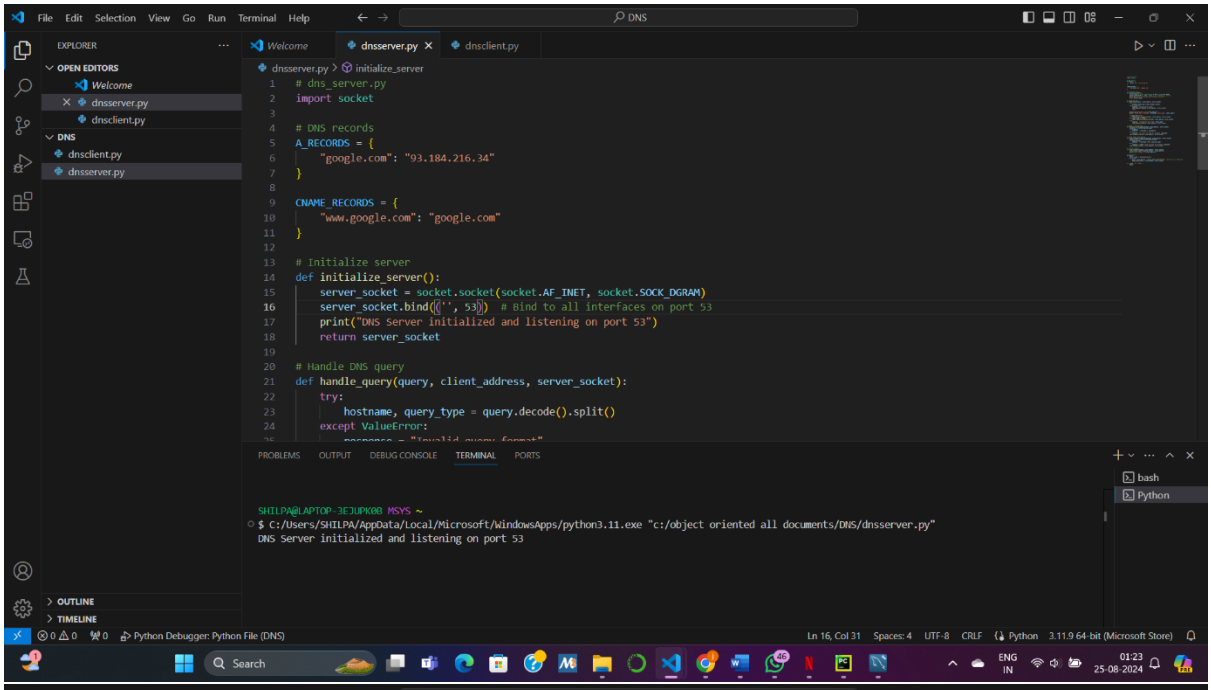
2. How does this relate to what you already know?

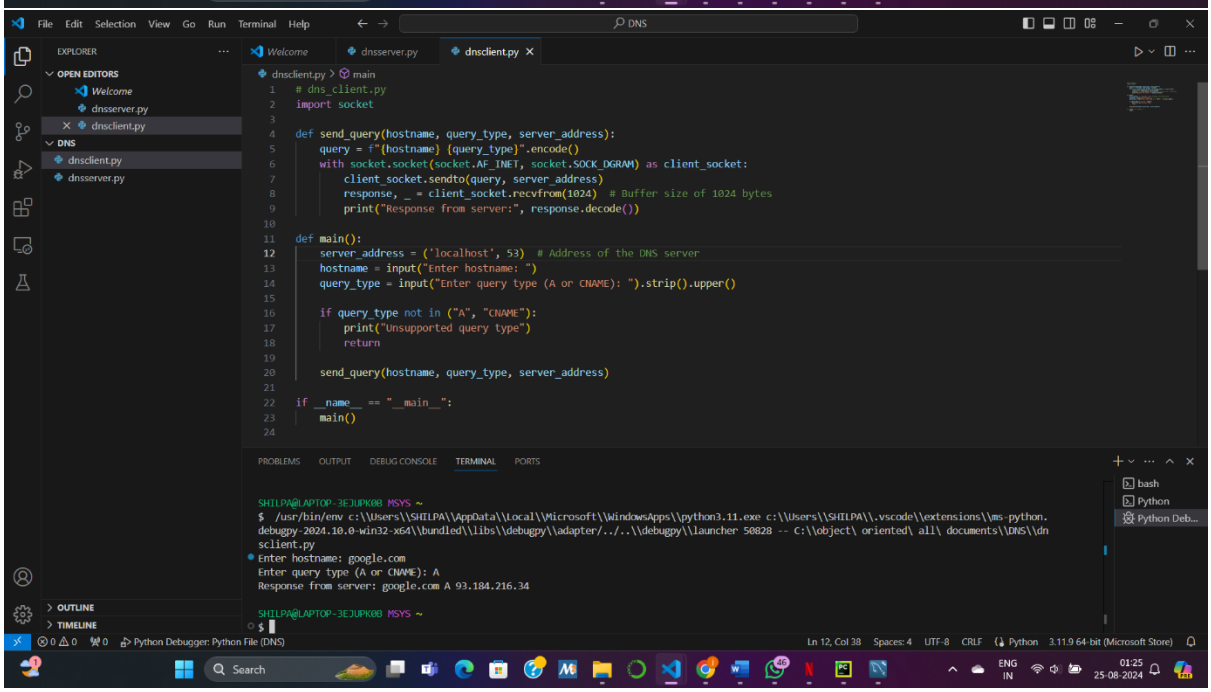
Building on Network Fundamentals: This module builds on basic networking knowledge, particularly UDP communication. It extends understanding to how DNS (a fundamental network service) operates, providing practical insights into real-world network applications.

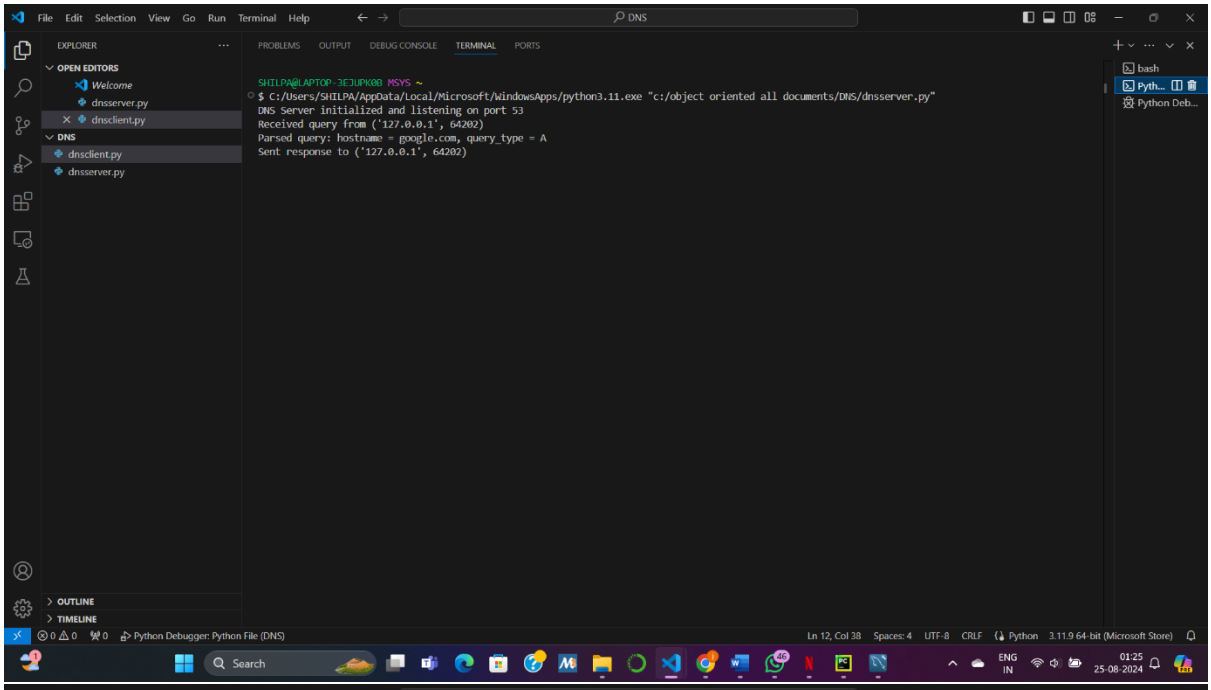
3. Why do you think your course team wants you to learn the content of this module?

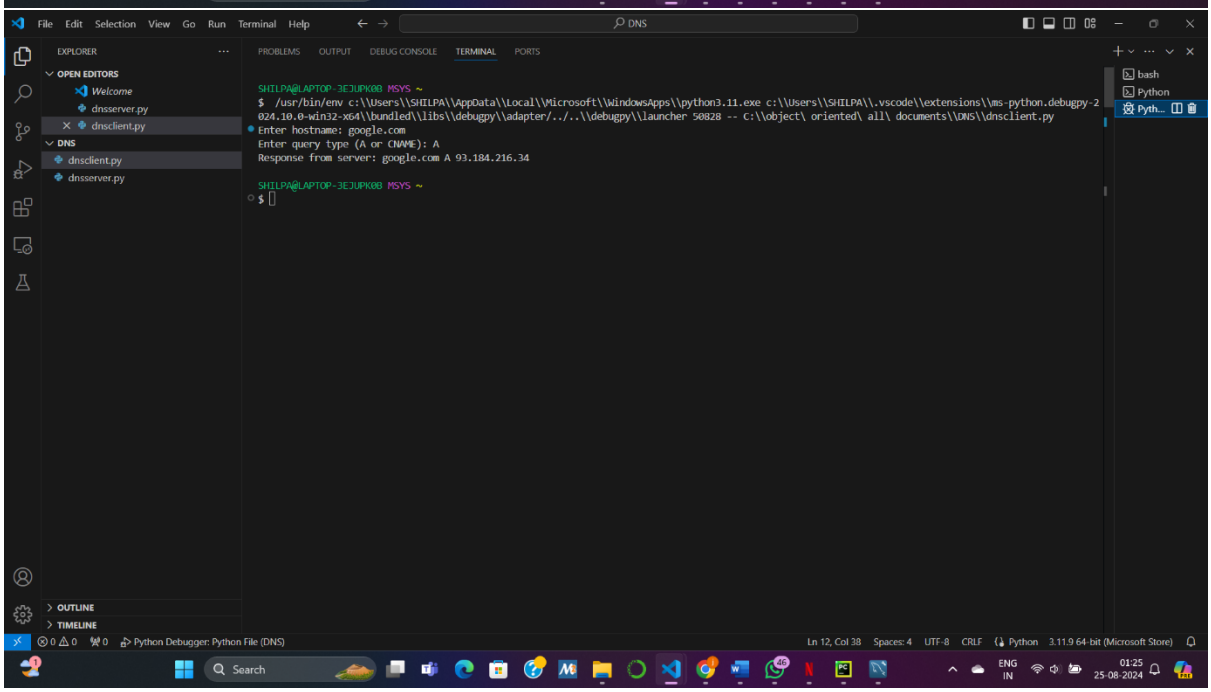
Foundation for Networking: Learning about DNS and how to implement a simple DNS server provides foundational knowledge for more complex network services and applications. Understanding DNS is crucial for troubleshooting network issues, configuring network services, and developing applications that rely on domain name resolution.

EVIDENCES:


File Edit Selection View Go Run Terminal Help
DNS
EXPLORER
OPEN EDITORS
Welcome
dnsserver.py
dnsclient.py
DNS
dnsclient.py
dnsserver.py
dnsserver.py > initialize_server
# dns_server.py
import socket
# DNS records
A_RECORDS = {
"google.com": "93.184.216.34"
}
CNAME_RECORDS = {
"www.google.com": "google.com"
}
# Initialize server
def initialize_server():
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind(('', 53)) # Bind to all interfaces on port 53
print("DNS Server initialized and listening on port 53")
return server_socket
# Handle DNS query
def handle_query(query, client_address, server_socket):
try:
hostname, query_type = query.decode().split()
except ValueError:
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash
Python
SHILPA@LAPTOP-3EJUPK0B MSYS ~
$ C:/Users/SHILPA/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/object oriented all documents/DNS/dnsserver.py"
DNS Server initialized and listening on port 53
OUTLINE
TIMELINE
Python Debugger: Python File (DNS)
Ln 16, Col 31 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Search
ENG IN
01:23 25-08-2024


File Edit Selection View Go Run Terminal Help
DNS
EXPLORER
OPEN EDITORS
Welcome
dnsserver.py
dnsclient.py
DNS
dnsclient.py
dnsserver.py
dnsclient.py > main
# dns_client.py
import socket
def send_query(hostname, query_type, server_address):
query = f"{hostname} {query_type}".encode()
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as client_socket:
client_socket.sendto(query, server_address)
response, _ = client_socket.recvfrom(1024) # Buffer size of 1024 bytes
print("Response from server:", response.decode())
def main():
server_address = ('localhost', 53) # Address of the DNS server
hostname = input("Enter hostname: ")
query_type = input("Enter query type (A or CNAME): ").strip().upper()
if query_type not in ("A", "CNAME"):
print("Unsupported query type")
return
send_query(hostname, query_type, server_address)
if __name__ == "__main__":
main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash
Python
Python Deb...
SHILPA@LAPTOP-3EJUPK0B MSYS ~
$ /usr/bin/env c:\\Users\\SHILPA\\AppData\\Local\\Microsoft\\WindowsApps\\python3.11.exe c:\\Users\\SHILPA\\.vscode\\extensions\\ms-python.debugpy-2024.10.0-win32-x64\\bundled\\libs\\debugpy\\adapter/../..\\debugpy\\launcher 50828 -- C:\\object\ oriented\ all\ documents\\DNS\\dnsclient.py
Enter hostname: google.com
Enter query type (A or CNAME): A
Response from server: google.com A 93.184.216.34
SHILPA@LAPTOP-3EJUPK0B MSYS ~
$
OUTLINE
TIMELINE
Python Debugger: Python File (DNS)
Ln 12, Col 38 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Search
ENG IN
01:25 25-08-2024


File Edit Selection View Go Run Terminal Help
DNS
EXPLORER
OPEN EDITORS
Welcome
dnsserver.py
dnsclient.py
DNS
dnsclient.py
dnsserver.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
SHILPA@LAPTOP-3EJUPK0B MSYS ~
$ C:/Users/SHILPA/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/object oriented all documents/DNS/dnsserver.py"
DNS Server initialized and listening on port 53
Received query from ('127.0.0.1', 64202)
Parsed query: hostname = google.com, query_type = A
Sent response to ('127.0.0.1', 64202)
bash
Pyth...
Python Deb...
OUTLINE
TIMELINE
Python Debugger: Python File (DNS)
Ln 12, Col 38 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Search
ENG IN
01:25 25-08-2024


File Edit Selection View Go Run Terminal Help
DNS
EXPLORER
OPEN EDITORS
Welcome
dnsserver.py
dnsclient.py
DNS
dnsclient.py
dnsserver.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
SHILPA@LAPTOP-3EJUPK0B MSYS ~
$ /usr/bin/env c:\\Users\\SHILPA\\AppData\\Local\\Microsoft\\WindowsApps\\python3.11.exe c:\\Users\\SHILPA\\.vscode\\extensions\\ms-python.debugpy-2024.10.0-win32-x64\\bundled\\libs\\debugpy\\adapter/../..\\debugpy\\launcher 50828 -- C:\\object\ oriented\ all\ documents\\DNS\\dnsclient.py
Enter hostname: google.com
Enter query type (A or CNAME): A
Response from server: google.com A 93.184.216.34
SHILPA@LAPTOP-3EJUPK0B MSYS ~
$
bash
Python
Pyth...
OUTLINE
TIMELINE
Python Debugger: Python File (DNS)
Ln 12, Col 38 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Search
ENG IN
01:25 25-08-2024