ABOVE AND BEYOND PASS

COMPUTER NETWORKS AND COMMUNICATION

JASVEENA -224001588

APPLICATION LAYER SUMMARY

APPLICATION LAYER delves into the application layer of the TCP/IP model, which is responsible for enabling communication between networked devices through various protocols. Network applications, such as web browsers, email clients, and file-sharing apps, operate at this layer to facilitate data exchange. The module explores the client-server architecture, where servers are always on with a fixed IP address and clients connect intermittently with dynamic IPs. It also covers peer-to-peer architecture, where devices communicate directly without a central server.

Key concepts include application layer protocols like HTTP (Hypertext Transfer Protocol) and HTTPS (Hypertext Transfer Protocol Secure). HTTP governs how web pages are requested and served, while HTTPS adds a layer of security through encryption using SSL/TLS. The module explains HTTP connections and methods, such as GET, POST, and PUT, which define how data is requested, sent, or updated.

Cookies are discussed as small data pieces stored by browsers to remember user information, and web caching is explained as a technique to speed up page loading by storing copies of frequently accessed web pages. Additionally, the module covers DNS (Domain Name System), which translates human-readable domain names into IP addresses.

The role of sockets as endpoints for network communication and the transport layer protocols (TCP and UDP) that manage data transfer through these sockets are also detailed. TCP ensures reliable delivery of data, while UDP provides faster but less reliable transmission.

Reflection

What I Learned: I learned about the crucial role of the application layer in network communication, including how protocols like HTTP and HTTPS manage web interactions and secure data transfer. The module highlighted the importance of cookies, web caches, and DNS in optimizing and securing web traffic.

Usefulness: This understanding is vital for developing, troubleshooting, and securing networked applications. Knowing how these protocols and components work helps in designing efficient applications and implementing effective security measures.

Most Important Thing: The most important takeaway is the understanding of how HTTP and HTTPS function in managing web communication and ensuring security. This foundational knowledge is critical for grasping the broader concepts of network communication and application security.

ACTIVITY 1

ANALYSING TLS

Message Sequence and Protocols Prior to the First HTTP GET Request

Transport Layer Protocol: The transport layer protocol employed is Transmission Control Protocol (TCP).

Application Layer Protocol: The application layer protocol in use is Hypertext Transfer Protocol Secure (HTTPS), which operates over Transport Layer Security (TLS).

Message Sequence:

1. **TCP Handshake:**
   - The client and server initiate a TCP connection through a three-way handshake process:
     - **SYN:** The client sends a SYN (synchronize) packet to the server to initiate the connection.
     - **SYN-ACK:** The server responds with a SYN-ACK (synchronize-acknowledge) packet to acknowledge the client's SYN and synchronize its own sequence numbers.
     - **ACK:** The client sends an ACK (acknowledge) packet to confirm the server’s SYN-ACK, thus establishing the TCP connection.

2. **TLS Handshake:**
   - Once the TCP connection is established, the client and server proceed with the TLS handshake to establish a secure communication channel:
     - **ClientHello:** The client sends a `ClientHello` message, proposing encryption algorithms and other security parameters.
     - **ServerHello:** The server responds with a `ServerHello` message, agreeing on encryption methods and sending its digital certificate for authentication.

- **Key Exchange and Session Keys:** The client and server exchange key information and generate session keys to encrypt the communication.

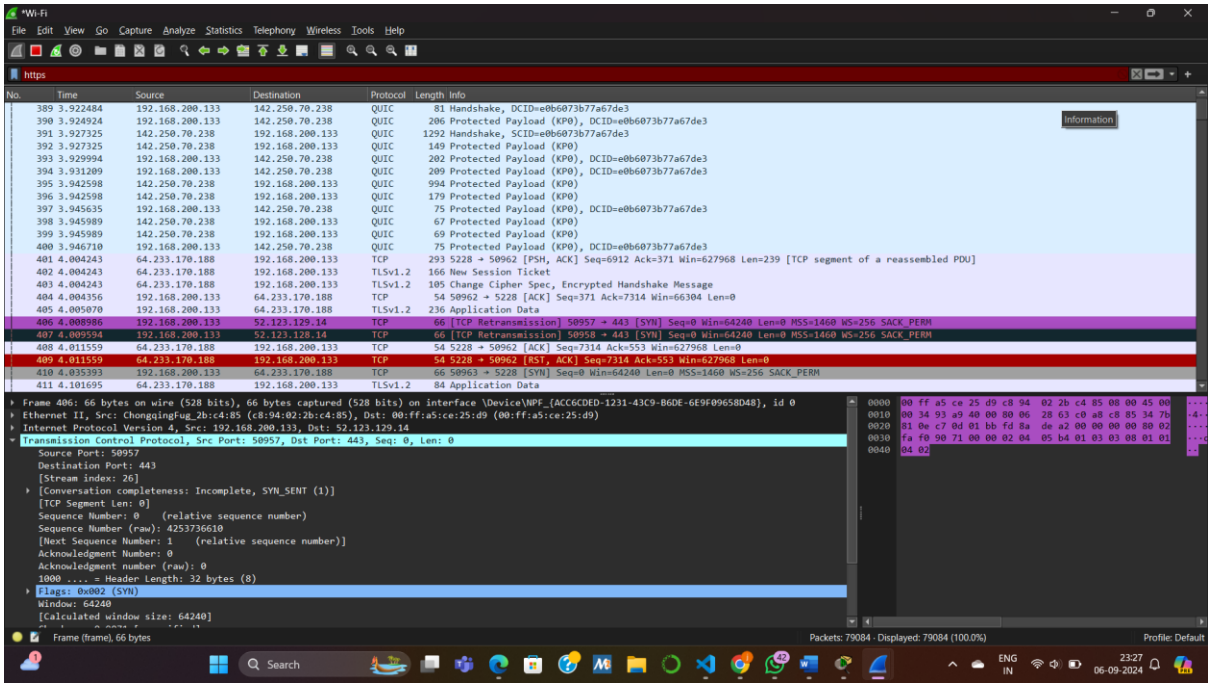
- **Finished Messages:** Both parties send `Finished` messages to confirm that the handshake is complete and the secure connection is established.

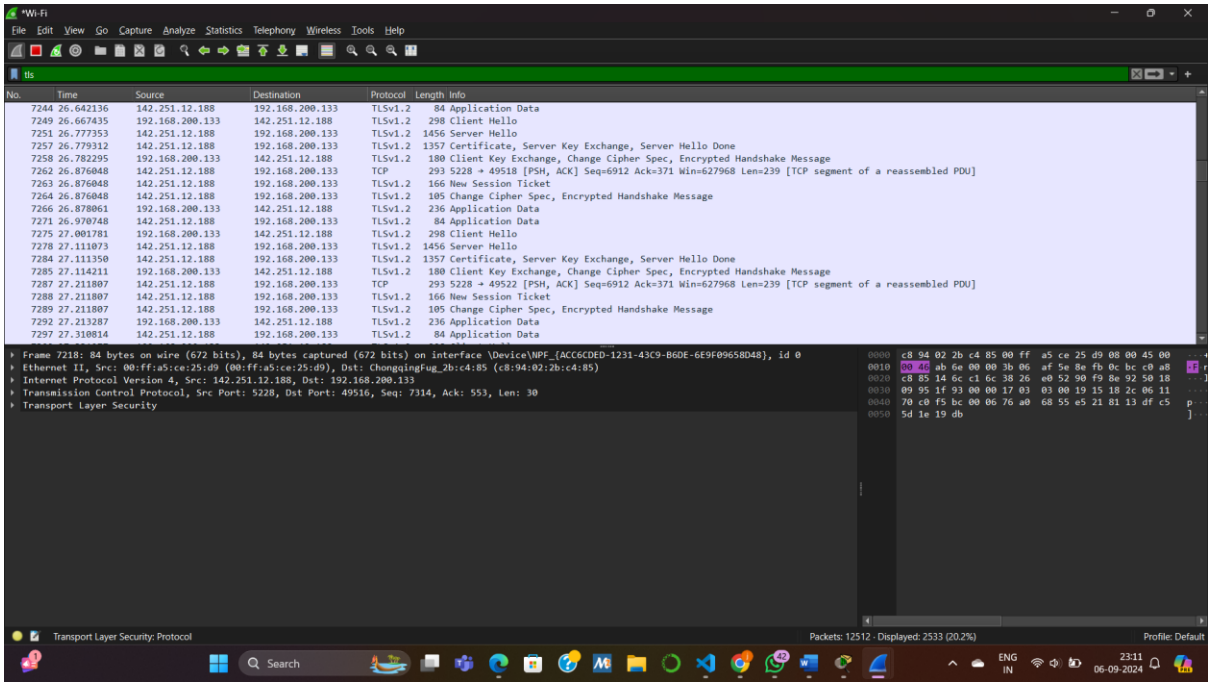
3. **HTTP GET Request:**

- With the secure TLS connection established, the client proceeds to send the first HTTP GET request:

- The client formulates and transmits an HTTP GET request to the server, requesting specific resources over the secured HTTPS connection.

This sequence ensures that the data exchanged between the client and server is securely transmitted and protected from potential eavesdropping or tampering.





TLS Operation and Handshake Process

ClientHello: The TLS handshake process begins with the client sending a `ClientHello` message to the server. This message specifies the TLS versions, cipher suites, and other cryptographic options supported by the client.

ServerHello: In response, the server sends a `ServerHello` message, selecting the TLS version and cipher suite from the client’s proposed list. This message also confirms the server’s choice of protocol and cryptographic settings.

Certificate: The server provides its digital certificate to the client. This certificate contains the server’s public key and is used to authenticate the server's identity.

ServerKeyExchange: If the selected cipher suite requires additional parameters, the server sends a `ServerKeyExchange` message containing these parameters. This message is crucial for establishing the secure key exchange mechanism.

ClientKeyExchange: The client responds with a `ClientKeyExchange` message. This message includes the pre-master secret, encrypted with the server's public key, which is essential for generating the session keys used for encrypting the communication.

Finished: Both the client and server exchange `Finished` messages to signal that the handshake is complete. These messages confirm that all the handshake parameters have been successfully negotiated and that future messages will be encrypted using the derived session keys.

HTTPS Analysis

Due to the encryption provided by HTTPS, the content of HTTPS messages cannot be directly examined using tools like Wireshark. However, analysis of the TLS handshake messages is possible and can provide valuable insights into the encryption process and key exchange mechanisms. This analysis helps in understanding the establishment of secure communication channels, even though the encrypted payloads themselves remain opaque.

ACTIVITY 2 :

Q1: What is the principal application layer protocol used in e-mails?

Principal Application Layer Protocols for Email

1. Simple Mail Transfer Protocol (SMTP)

SMTP is the standard protocol for sending emails over the Internet. It facilitates the submission of outgoing messages from clients to servers and between servers. Key functions include:

- **Message Submission:** SMTP transfers emails from a client to a mail server.
- **Message Relay:** It routes emails between servers until they reach the recipient's server.
- **Communication Flow:** Uses commands and responses to ensure reliable message delivery.

SMTP typically operates on port 25, though ports 587 and 465 are used for secure submissions.

2. Internet Message Access Protocol (IMAP)

IMAP is used for retrieving and managing emails stored on a mail server. It supports:

- **Message Retrieval:** Allows clients to fetch messages from the server, enabling access from multiple devices.
- **Folder Management:** Provides features to organize and manage emails into folders on the server.
- **Message Synchronization:** Ensures real-time updates and synchronization of email status across devices.

- **Partial Download: Downloads only message headers initially, saving bandwidth.**

IMAP usually operates on port 143, with secure connections on port 993. It is ideal for users who need access to their emails from various devices.

SMTP handles sending, while IMAP manages email retrieval and organization.

Q2: What is the underlaying architecture and transport layer protocol used in e- mail application layer protocol?

Underlaying architecture and transport layer protocol used in e- mail application layer protocols are:

SMTP : Architecture: Client-server architecture Transport layer protocol:

TCP on port 25 (unencrypted)

IMAP: Architecture: Client-server architecture Transport layer protocol:

TCP on port 143(unencrypted)

Q3: Can you list down the basic steps involve in sending an e-mail from user A to B?

Email Transmission Process

1. **Email Composition:** User A drafts an email using an email client, such as Gmail.

2. **Email Submission:** The email client sends the composed message to the SMTP server utilizing the SMTP protocol.

3. **Email Processing:** The SMTP server processes the email and determines the recipient's domain based on the email address.

4. **Email Relay:** The SMTP server relays the email through the appropriate mail transfer path to the recipient’s SMTP server.

5. **Email Receipt:** The recipient’s SMTP server receives the email and stores it in the recipient’s mailbox.

6. **Email Retrieval:** User B accesses and reads the email from their mailbox using an email client or application.

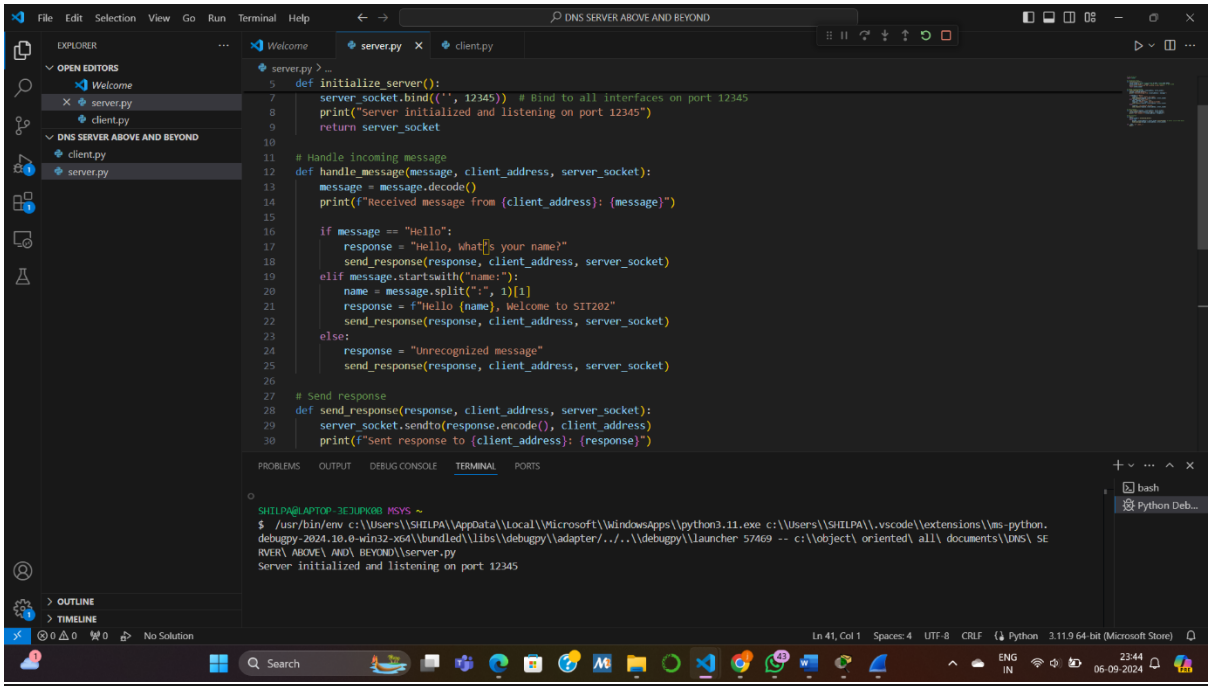
SUMMARY OF TRANSPORT LAYER:

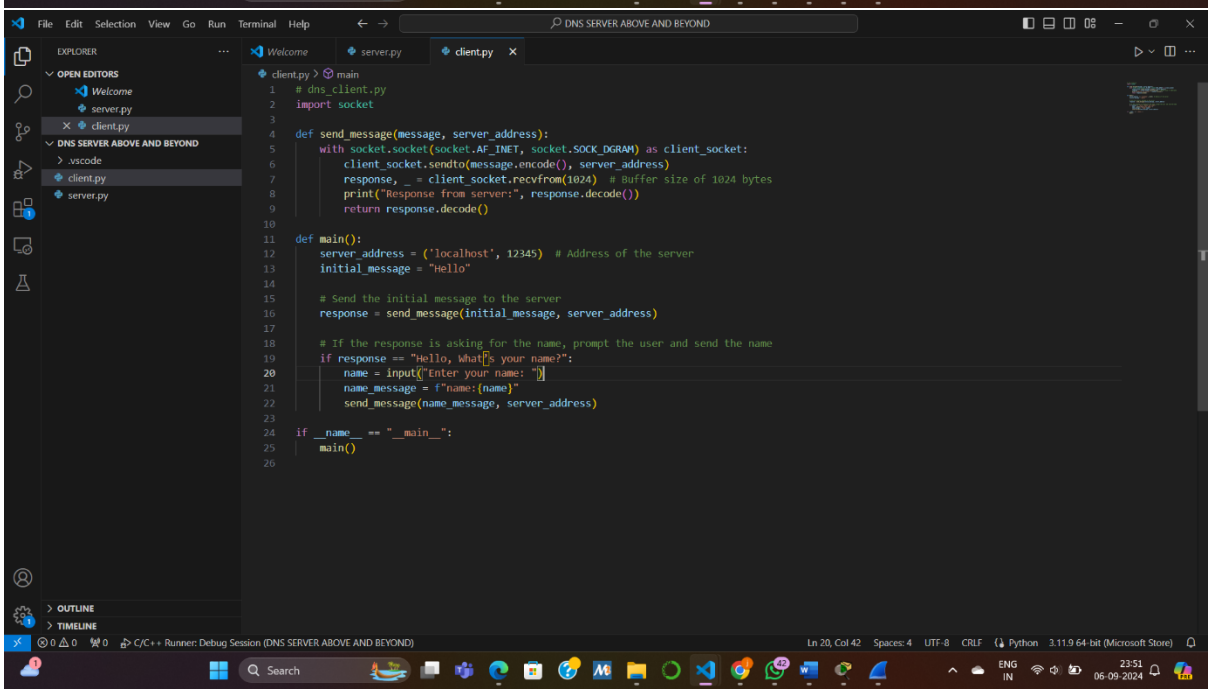
Transport Layer delves into how the transport layer ensures effective communication between application processes running on different hosts. It contrasts the transport layer's role with the network layer, which manages connectivity between hosts rather than processes. Two primary transport layer protocols, TCP (Transmission Control Protocol) and UDP (User Datagram Protocol), are examined in detail.

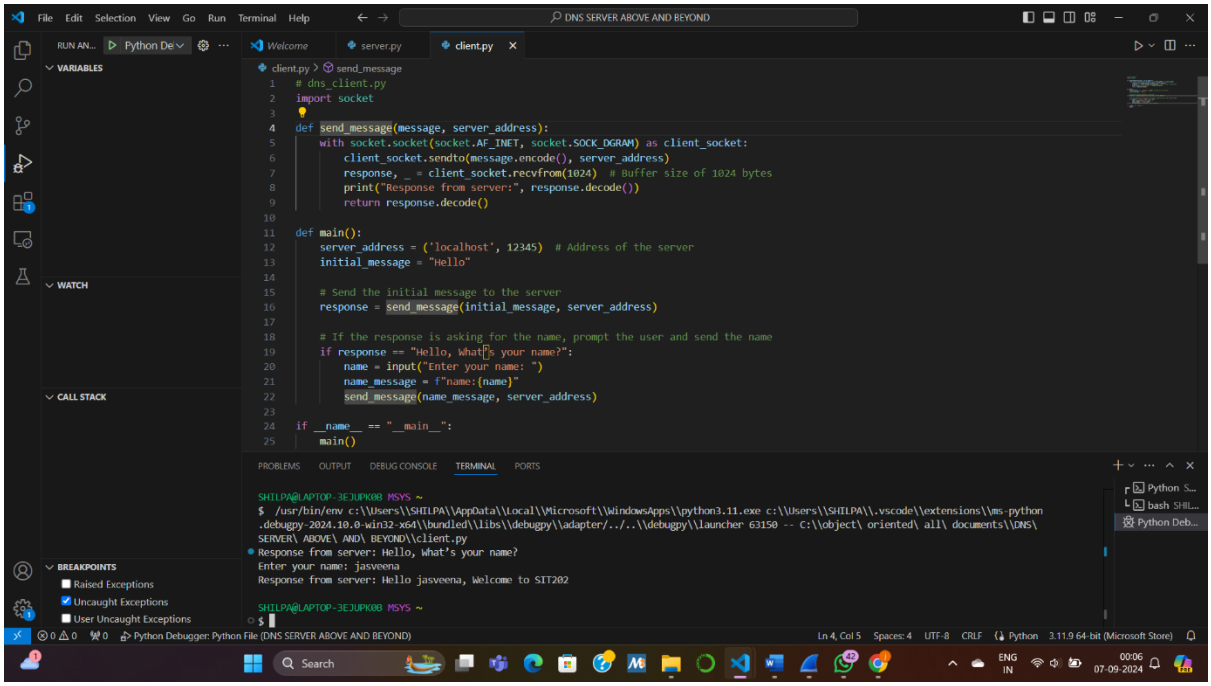
TCP is a connection-oriented protocol providing reliable, in-order data delivery with built-in mechanisms for congestion control, flow control, and error recovery. It uses a connection setup process called the "three-way handshake" to establish a session between sender and receiver. TCP also handles retransmissions to ensure data integrity, uses a sliding window for flow control, and implements Automatic Repeat reQuest (ARQ) protocols like Stop-and-Wait, Go-Back-N, and Selective Repeat to manage packet delivery.

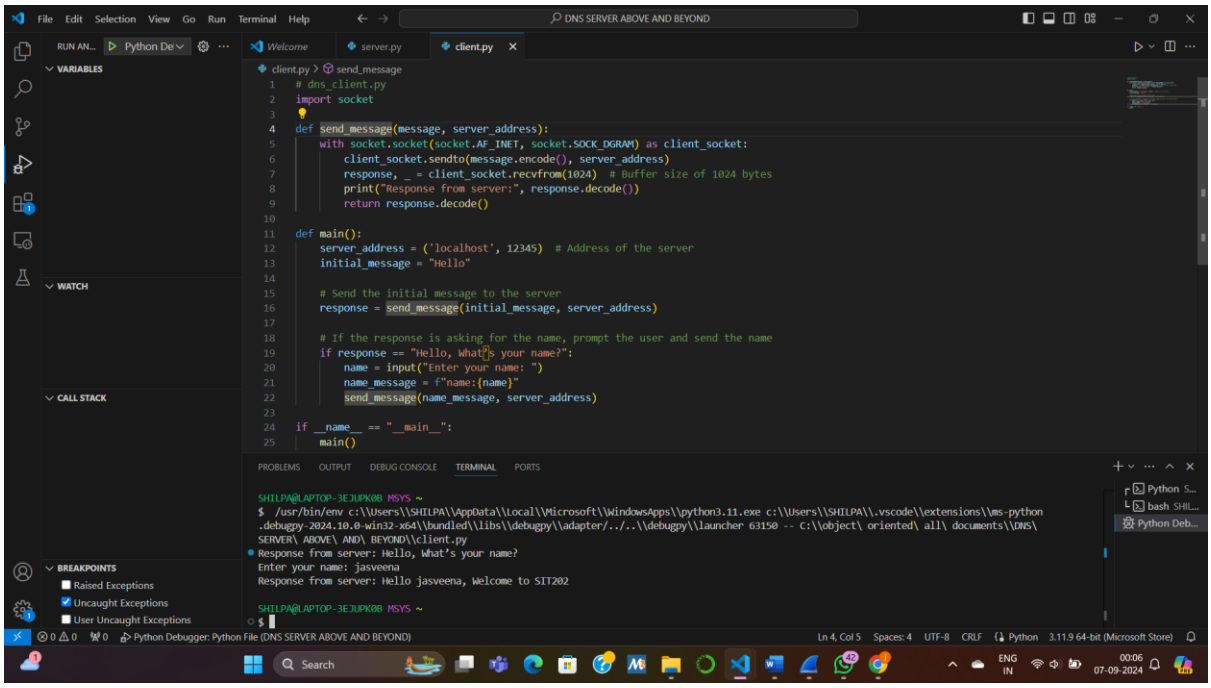
UDP, on the other hand, is connectionless and provides a simpler, faster alternative with minimal overhead. It does not guarantee delivery, order, or error recovery, making it suitable for applications where speed is crucial and occasional data loss is acceptable, such as live video streaming.

The module also introduces the concepts of multiplexing and demultiplexing, which manage how data from multiple processes is combined and separated during transmission. Sockets serve as endpoints for communication, linking application processes with transport layer protocols. Additionally, the module covers TCP's congestion control mechanisms, including Slow Start and Congestion Avoidance, which adapt the transmission rate based on network conditions to prevent congestion and ensure efficient communication.


File Edit Selection View Go Run Terminal Help
DNS SERVER ABOVE AND BEYOND
EXPLORER
OPEN EDITORS
Welcome
server.py
client.py
DNS SERVER ABOVE AND BEYOND
client.py
server.py
Welcome
server.py
client.py
server.py > ...
def initialize_server():
server_socket.bind(('', 12345)) # Bind to all interfaces on port 12345
print("Server initialized and listening on port 12345")
return server_socket
# Handle incoming message
def handle_message(message, client_address, server_socket):
message = message.decode()
print(f"Received message from {client_address}: {message}")
if message == "Hello":
response = "Hello, What's your name?"
send_response(response, client_address, server_socket)
elif message.startswith("name:"):
name = message.split(":", 1)[1]
response = f"Hello {name}, Welcome to SIT202"
send_response(response, client_address, server_socket)
else:
response = "Unrecognized message"
send_response(response, client_address, server_socket)
# Send response
def send_response(response, client_address, server_socket):
server_socket.sendto(response.encode(), client_address)
print(f"Sent response to {client_address}: {response}")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash
Python Deb...
SHILPA@LAPTOP-3E1UPK0B 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 57469 -- c:\\object\ oriented\ all\ documents\\DNS\ SERVER\ ABOVE\ AND\ BEYOND\\server.py
Server initialized and listening on port 12345
OUTLINE
TIMELINE
No Solution
Ln 41, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Search
ENG IN
23:44 06-09-2024


File Edit Selection View Go Run Terminal Help
DNS SERVER ABOVE AND BEYOND
EXPLORER
OPEN EDITORS
Welcome
server.py
client.py
DNS SERVER ABOVE AND BEYOND
.vscode
client.py
server.py
Welcome
server.py
client.py
client.py > main
# dns_client.py
import socket
def send_message(message, server_address):
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as client_socket:
client_socket.sendto(message.encode(), server_address)
response, _ = client_socket.recvfrom(1024) # Buffer size of 1024 bytes
print("Response from server:", response.decode())
return response.decode()
def main():
server_address = ('localhost', 12345) # Address of the server
initial_message = "Hello"
# Send the initial message to the server
response = send_message(initial_message, server_address)
# If the response is asking for the name, prompt the user and send the name
if response == "Hello, What's your name?":
name = input("Enter your name: ")
name_message = f"name:{name}"
send_message(name_message, server_address)
if __name__ == "__main__":
main()
OUTLINE
TIMELINE
C/C++ Runner: Debug Session (DNS SERVER ABOVE AND BEYOND)
Ln 20, Col 42 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Search
ENG IN
23:51 06-09-2024





ACTIVITY 2

Differences Between Protocols and Reasons for Their Use

1. Stop-and-Wait Protocol

- **Description:** In the Stop-and-Wait protocol, the sender transmits a single frame and waits for an acknowledgment (ACK) from the receiver before sending the next frame.

- **Advantages:**

  - **Simplicity:** This protocol is straightforward to implement and understand.

  - **Reliability:** It ensures reliable data transfer since the sender waits for an ACK before proceeding.

- **Disadvantages:**

  - **Inefficiency in High Latency Networks:** The sender's wait time for ACKs can lead to inefficiencies, particularly in high latency environments where waiting for each ACK can significantly impact throughput.

  - **Suitability:** This protocol is best suited for low-latency, low-bandwidth networks where simplicity and reliability are prioritized over efficiency.

2. Go-Back-N Protocol

- **Description:** The Go-Back-N protocol allows the sender to transmit multiple frames before requiring an ACK for the first frame. If an error is detected in any frame, all subsequent frames are retransmitted.

- **Advantages:**

  - **Increased Efficiency:** This protocol improves throughput by enabling multiple frames to be in transit simultaneously, which is more efficient than Stop-and-Wait.

- **Disadvantages:**

  - **Unnecessary Retransmissions:** In case of an error, all subsequent frames are retransmitted, which can lead to inefficiencies if only a single frame is lost or corrupted.

- **Suitability:** It is appropriate for networks with moderate latency and error rates where a balance between efficiency and error handling is required.

3. Selective Repeat Protocol

- **Description:** In the Selective Repeat protocol, the sender transmits multiple frames and retransmits only those frames that are erroneous. The receiver acknowledges each frame individually.

- **Advantages:**

  - **Optimized Efficiency:** This protocol minimizes retransmissions by only resending erroneous frames, which enhances overall efficiency.

- **Disadvantages:**

  - **Complex Implementation:** It requires maintaining multiple buffers and sequence numbers, making it more complex to implement than Stop-and-Wait or Go-Back-N.

  - **Suitability:** It is well-suited for high-latency, high-bandwidth networks where efficiency and minimizing retransmissions are critical.

Importance of Congestion Control in TCP

1. Prevention of Network Congestion:

- Congestion control mechanisms are crucial to prevent network congestion. By regulating the rate of data transmission, these mechanisms help avoid overloading the network, which can lead to packet loss and increased delays.

2. Performance Improvement:

- Effective congestion control enhances overall network performance by reducing packet loss, minimizing delays, and ensuring a smooth flow of data. This contributes to a more efficient use of network resources.

3. Network Stability:

- Congestion control maintains network stability by managing data flow and preventing sudden spikes in traffic. This helps avoid network collapse and ensures consistent network operation.

In summary, congestion control in TCP is essential for maintaining network efficiency, preventing overload, and ensuring stable and reliable data transmission.

Reflection

What I Learned: I gained a deeper understanding of how the transport layer protocols, TCP and UDP, manage data transmission and ensure effective communication between processes on different hosts. Learning about TCP's connection setup, flow control, error recovery, and congestion control mechanisms provided insight into how reliable communication is maintained. On the other hand, UDP's simplicity and speed for less critical data transmissions were also highlighted.

Usefulness: This knowledge is essential for designing and troubleshooting network applications. Understanding TCP's reliability features helps in

developing applications where data integrity is crucial, while UDP’s low overhead makes it ideal for applications where speed is more important than reliability. These concepts are fundamental for optimizing network performance and ensuring robust communication in diverse scenarios.

Most Important Thing: The most important takeaway is the distinction between TCP and UDP and their respective roles in network communication. TCP's comprehensive approach to ensuring reliable data transfer, including its mechanisms for handling errors and congestion, contrasts sharply with UDP's focus on speed and efficiency. This understanding is crucial for selecting the appropriate protocol based on application needs and network conditions.

SUMMARY OF NETWORK LAYER

The Network Layer module provides an in-depth overview of how data is routed and forwarded across networks. This layer is crucial for ensuring that data packets travel efficiently from their source to their destination through various networks.

Key Functions of the Network Layer:

Routing: Determines the optimal path for data packets to travel from the source to the destination.

Forwarding: Transfers packets from input to output ports within routers.

Logical Addressing: Utilizes IP addresses to uniquely identify devices and facilitate communication.

Packet Switching: Breaks down data into smaller packets for routing.

Router Components:

Input Ports: Receive packets from different networks.

Switching Fabric: Directs packets internally within the router.

Output Ports: Send packets out to the next network or device.

The module includes an analogy likening network operations to a postal service, where input ports are mailboxes, switching fabrics are sorting centers, and output ports are mail carriers.

IPv4 and IPv6 Addressing:

IPv4 Datagram: Consists of a header (with fields like Source IP Address, Destination IP Address, TTL, etc.) and a payload. Fragmentation is used when datagrams exceed the Maximum Transmission Unit (MTU) of a network.

IPv6: Offers a much larger address space (128-bit addresses) compared to IPv4, and uses a simplified header structure with no checksum for error-checking, handled by higher layers instead.

Transition Mechanisms to IPv6:

Tunneling: Encapsulates IPv6 packets within IPv4 packets to traverse IPv4-only networks.

Dual Stack: Allows running both IPv4 and IPv6 simultaneously.

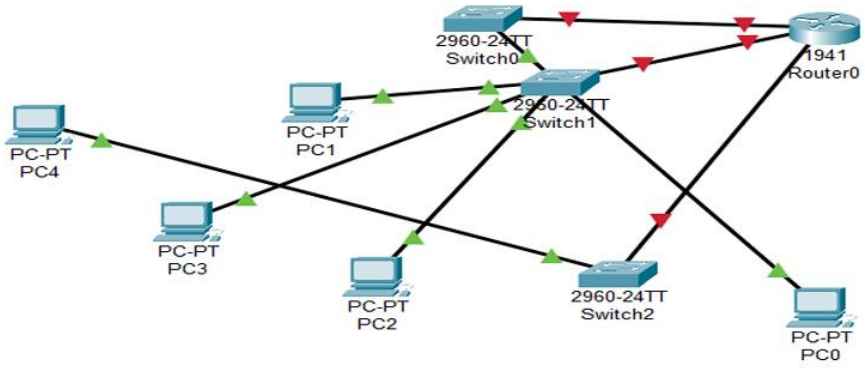
Practical Example: For a network with the address 192.168.40.128/26, there are 62 usable IP addresses available, with 192.168.40.129 and 192.168.40.130 being among them.

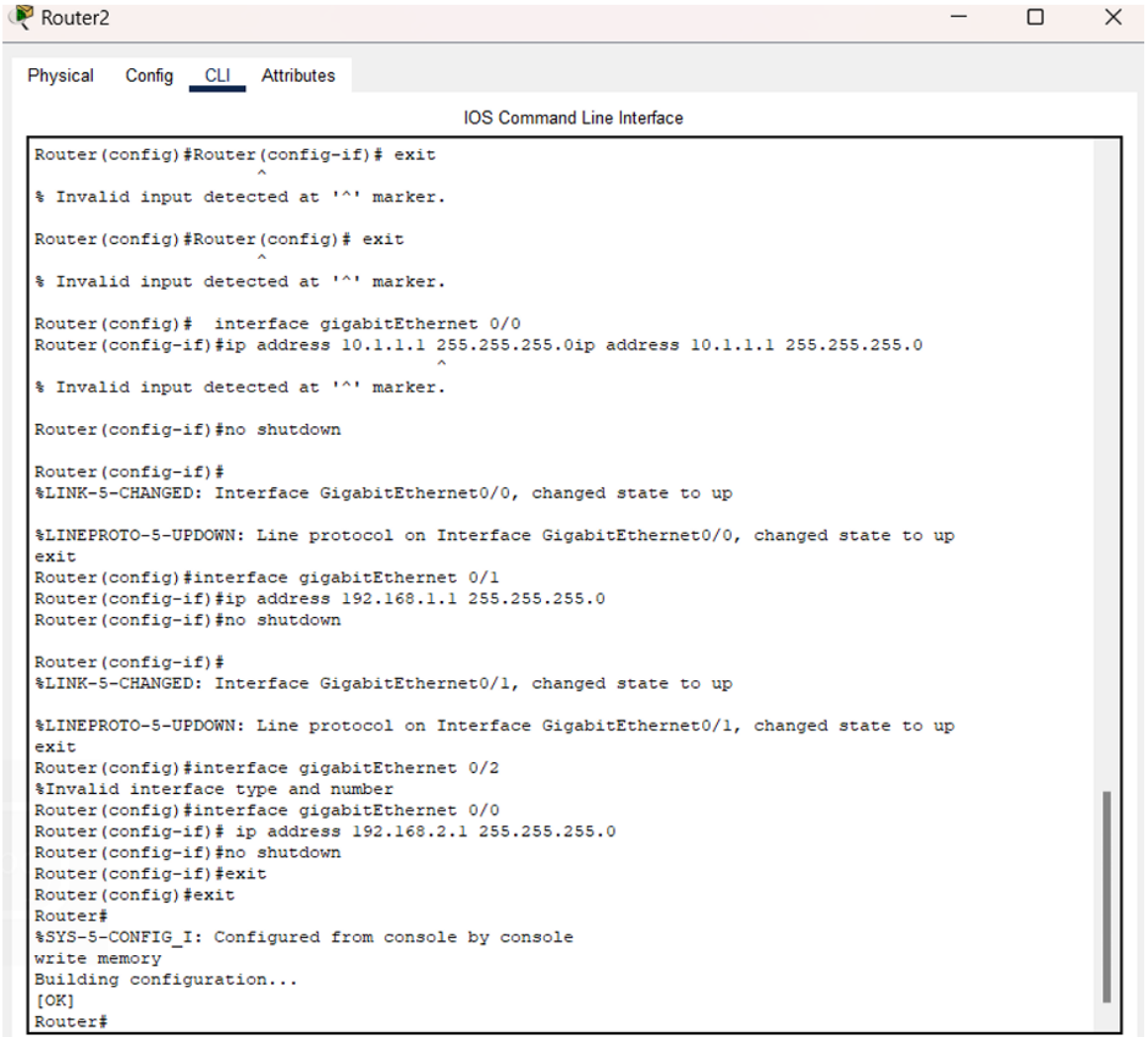
Reflection

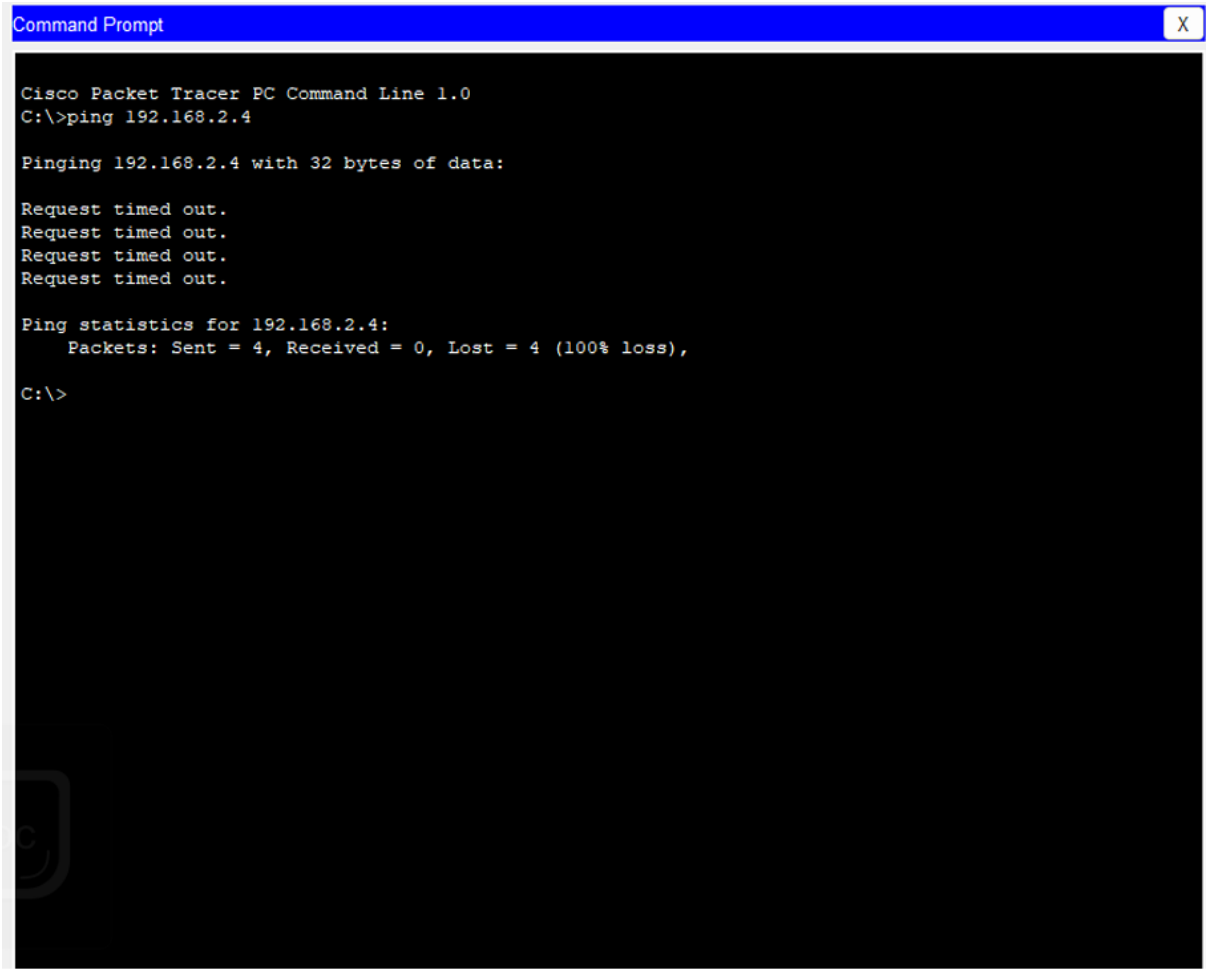
What I Learned: The module provided a comprehensive understanding of how the network layer manages data routing, packet switching, and logical addressing. I learned about the detailed functions of routers and the structure of IPv4 and IPv6 datagrams. The information on IP fragmentation and the transition mechanisms to IPv6 was particularly valuable.

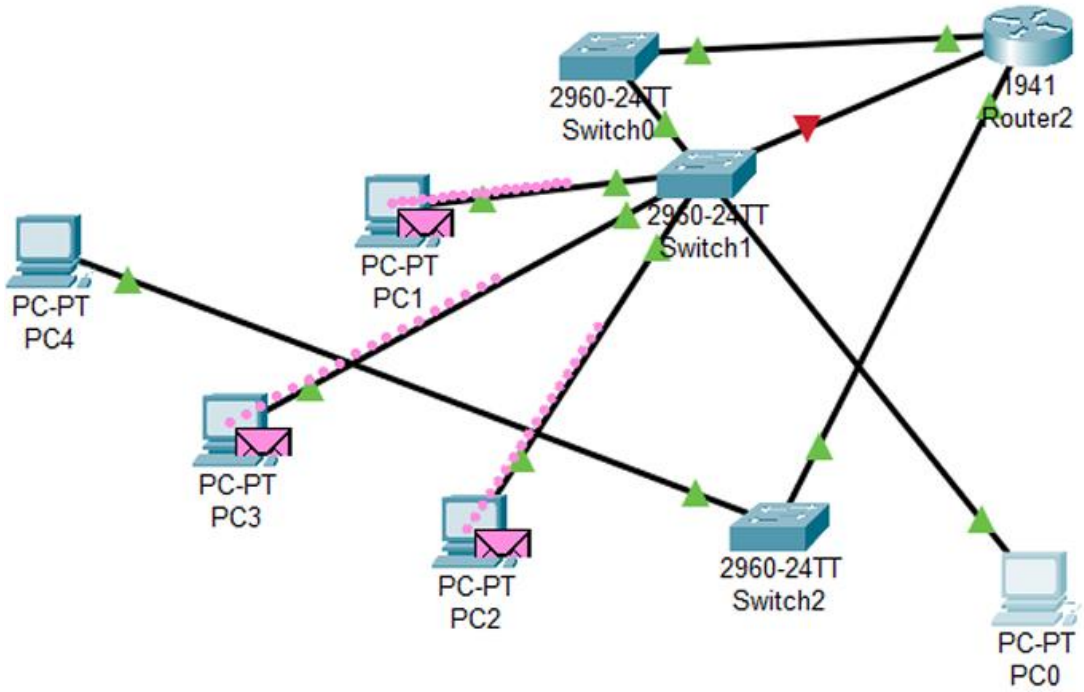
Usefulness: This knowledge is crucial for designing, managing, and troubleshooting network infrastructures. Understanding routing, addressing, and packet management helps ensure efficient data transmission and network performance. It's fundamental for tasks such as network configuration, optimization, and problem-solving.

Most Important Thing: The most significant takeaway was the detailed understanding of IP addressing and the differences between IPv4 and IPv6. This is essential for managing modern networks, particularly given the transition from IPv4 to IPv6 due to address space limitations. This knowledge is foundational for advanced networking concepts and real-world applications, including network design and cybersecurity.




Router2
Physical
Config
CLI
Attributes
IOS Command Line Interface
Router(config)#Router(config-if)# exit
^
% Invalid input detected at '^' marker.
Router(config)#Router(config)# exit
^
% Invalid input detected at '^' marker.
Router(config)# interface gigabitEthernet 0/0
Router(config-if)#ip address 10.1.1.1 255.255.255.0ip address 10.1.1.1 255.255.255.0
^
% Invalid input detected at '^' marker.
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0, changed state to up
exit
Router(config)#interface gigabitEthernet 0/1
Router(config-if)#ip address 192.168.1.1 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to up
exit
Router(config)#interface gigabitEthernet 0/2
%Invalid interface type and number
Router(config)#interface gigabitEthernet 0/0
Router(config-if)# ip address 192.168.2.1 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
write memory
Building configuration...
[OK]
Router#


Command Prompt
X
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.2.4
Pinging 192.168.2.4 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 192.168.2.4:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>


2960-24TT
Switch0
1941
Router2
2960-24TT
Switch1
PC-PT
PC1
PC-PT
PC4
PC-PT
PC3
PC-PT
PC2
2960-24TT
Switch2
PC-PT
PC0