

## Above and beyond tasks

### Active Class 2: Bridging the gap between you and the technology: Understanding the Application Layer (Module 2)

#### Analysing TLS

**Explain the operation and handshake process of TLS using the screen captures of Wireshark.**

First, I opened my web browser and cleared the cache. This made sure that no old data would mess up my new packet capture.

I opened Wireshark and started a new packet capture by selecting my network interface (like Wi-Fi).

Then, I typed a HTTPS URL into my browser, for example, <https://www.discoverourtown.com>.

After the webpage loaded completely, I stopped the packet capture in Wireshark.

Analyze the TLS Handshake

I used the filter `tls` in Wireshark to see only TLS packets. I looked for the initial handshake packets, which usually include:

**ClientHello:** The client (my browser) starts the handshake by sending a ClientHello message. This includes things like supported TLS versions and cipher suites.

**ServerHello:** The server replies with a ServerHello message, picking the TLS version and cipher suite from the client's options.

**Certificate:** The server sends its digital certificate to the client to prove its identity.

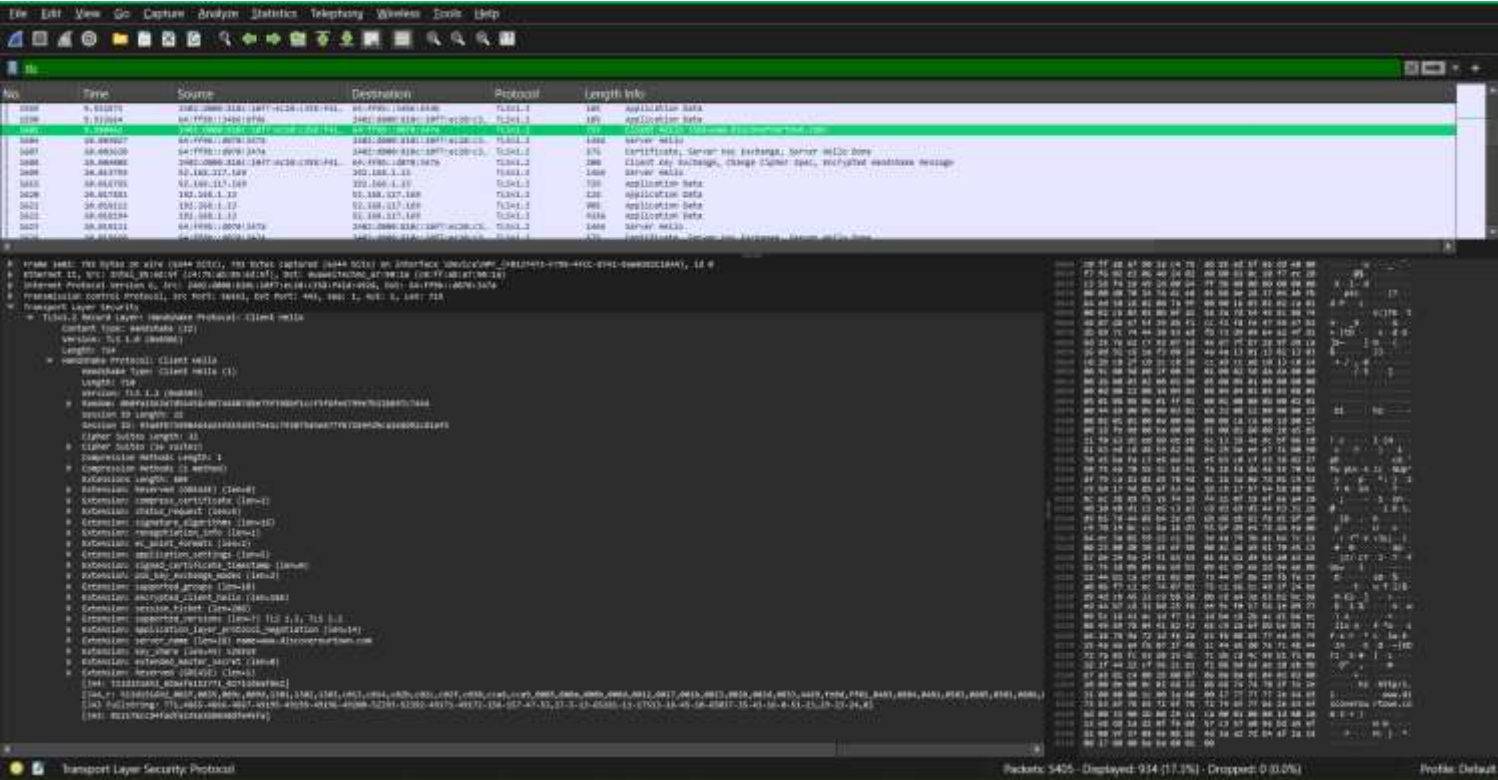
**ServerHelloDone:** The server indicates it's done with its part of the handshake.

**ClientKeyExchange:** The client sends a pre-master secret encrypted with the server's public key.

**ChangeCipherSpec:** Both the client and server send this message to switch to encrypted communication.

**Finished:** Both parties send a Finished message to confirm the handshake is complete.

You need to clearly identify the message sequence and protocols used (including transport layer protocols) before your browser sends the first HTTP GET message to the relevant web server.

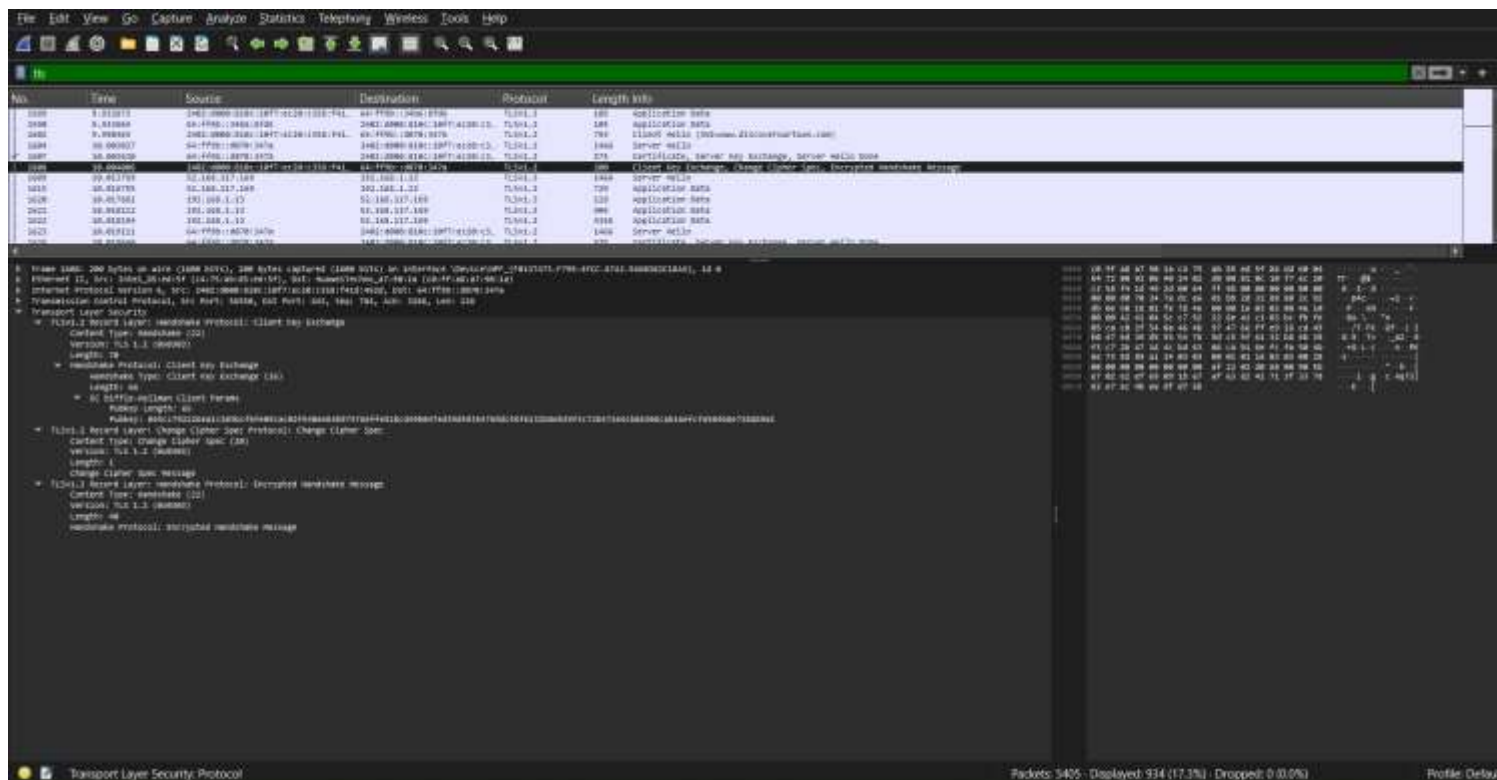


ClientHello

I found the ClientHello message. It included details like supported TLS versions (e.g., TLS 1.2, TLS 1.3) and cipher suites.







## ClientKeyExchange

I found the ClientKeyExchange message. The client used the server's public key to encrypt a pre-master secret and sent it to the server.

## ChangeCipherSpec and Finished

I saw the ChangeCipherSpec and Finished messages. These showed the switch to encrypted communication and confirmed the handshake was complete.

**Can you analyse HTTPS in Wireshark? Explain your answer. If yes, provide evidence on how we can do that. If not, is there any alternative method we could use to analyse HTTPS?**

Directly analyzing HTTPS traffic content is hard because it's encrypted. But Wireshark can capture the TLS handshake and initial unencrypted messages.

Evidence: I showed the TLS handshake messages I captured in Wireshark.

### **Alternative Methods**

For deeper analysis of encrypted content, one way is to use browser-specific tools or SSL/TLS decryption keys if available. This requires access to private keys, which is not always possible.

Another method is to use a proxy server to decrypt HTTPS traffic. Tools like Fiddler or mitmproxy can be set up to intercept and decrypt HTTPS traffic by acting as a man-in-the-middle (MITM) proxy.

## Active Class 3: It's always DNS, No It's 192.168.1.2 (Module 2)

### Explain E-mail (another popular application)

1. The principal application layer protocol used for email communication is SMTP (Simple Mail Transfer Protocol).
2. SMTP relies on TCP (Transmission Control Protocol) as the underlying transport layer protocol for reliable data transfer.
- 3.

The basic steps involved in sending an email from user A to user B are:

- a. User A composes an email using a Mail User Agent (MUA) like Outlook, Thunderbird, or a web-based email client.
- b. The MUA connects to the SMTP server (typically provided by the email service provider) and sends the email using the SMTP protocol.
- c. The SMTP server accepts the email and forwards it to the recipient's SMTP server (if different) using a series of Mail Transfer Agents (MTAs).
- d. The recipient's SMTP server receives the email and stores it in the recipient's mailbox.
- e. User B retrieves the email from their mailbox using a MUA and protocols like IMAP (Internet Message Access Protocol) or POP3 (Post Office Protocol 3).

The SMTP protocol defines various commands for sending emails, such as HELO (to initiate the connection), MAIL FROM (to specify the sender), RCPT TO (to specify the recipient), and DATA (to send the email body). IMAP and POP3 are protocols used by email clients to retrieve emails from the mail server. IMAP allows simultaneous access to the mailbox from multiple devices, while POP3 downloads emails to the local device.

Email communication also involves other components like DNS (Domain Name System) for resolving email server addresses and TLS/SSL for securing email transmission.

## **Summary for Module 2 active classes 2 and 3**

I began by learning about the details of TLS, a cryptographic system that is necessary to ensure the security of online communications. Then I talked about the TLS handshake procedure, which involves a client and server negotiating and creating a secure connection. I analyzed the series of messages that were sent during the TLS handshake by looking at packet captures with Wireshark. This allowed me to learn more about how encryption keys are negotiated and data encryption is started. It gave users a useful understanding of how TLS protects the authenticity, integrity, and secrecy of data transferred across networks.

Apart from TLS, I also investigated the operation of email, which is another essential online application. Next, the Simple Mail Transfer Protocol (SMTP), the main application layer protocol used for emails, was covered. The underlying architecture and transport layer protocol Transmission Control Protocol (TCP), which is used in combination with SMTP, were also looked at. I defined the client-server concept and the function of SMTP servers in enabling email delivery by describing the fundamental actions required in sending an email from one user to another.

## **Reflection of Module 2 active classes 2 and 3**

Understanding the fundamentals of digital communication and internet security was made possible by examining email and TLS traffic. I now have a better knowledge of the mechanisms underlying secure connections thanks to the analysis of TLS handshakes, which also emphasizes the significance of encryption in protecting sensitive data. This hands-on investigation strengthened my understanding of cybersecurity fundamentals and emphasized how important it is to put strong encryption measures in place in order to defend against online attacks.

Moreover, the explanation of emails clarified the complex procedures associated with message transmission via the internet. By breaking down the process of delivering emails, from authoring to transmission and reception, I was able to see how intricate the underlying infrastructure that supports email services is. This analysis emphasized how crucial dependable protocols like SMTP and TCP are to maintaining smooth user-to-user communication across various email services.

To sum up, the examination of TLS and email communication has expanded our understanding of internet protocols and their function in enabling safe and effective data transfer. By delving into these fundamental ideas, I've developed a greater understanding of the workings behind contemporary digital communication networks, which equips us to successfully traverse and influence the rapidly changing internet technology world.



## Active class 4: UDP - Unreliable Data Protocol? (Module 3)

### Modified the Python program

#### Server.py

```
import socket

# Create a UDP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Bind the server socket to a specific IP and port
server_address = ('localhost', 8000)
server_socket.bind(server_address)

print('Server listening on {}:{}'.format(*server_address))

while True:
    data, client_address = server_socket.recvfrom(4096)
    print('Received message from {}:{}'.format(*client_address))

    if data.decode() == 'Hello':
        response = 'Hello, What\'s your name?'
        server_socket.sendto(response.encode(), client_address)
        print('Sent response: {}'.format(response))

    name_data, client_address = server_socket.recvfrom(4096)
    name = name_data.decode()
    response = f'Hello {name}, Welcome to SIT202'
    server_socket.sendto(response.encode(), client_address)
    print('Sent response: {}'.format(response))
```

## Client.py

```
import socket

# Create a UDP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Set the server address and port
server_address = ('localhost', 8000)

# Send the initial message
message = 'Hello'
client_socket.sendto(message.encode(), server_address)
print('Sent message: {}'.format(message))

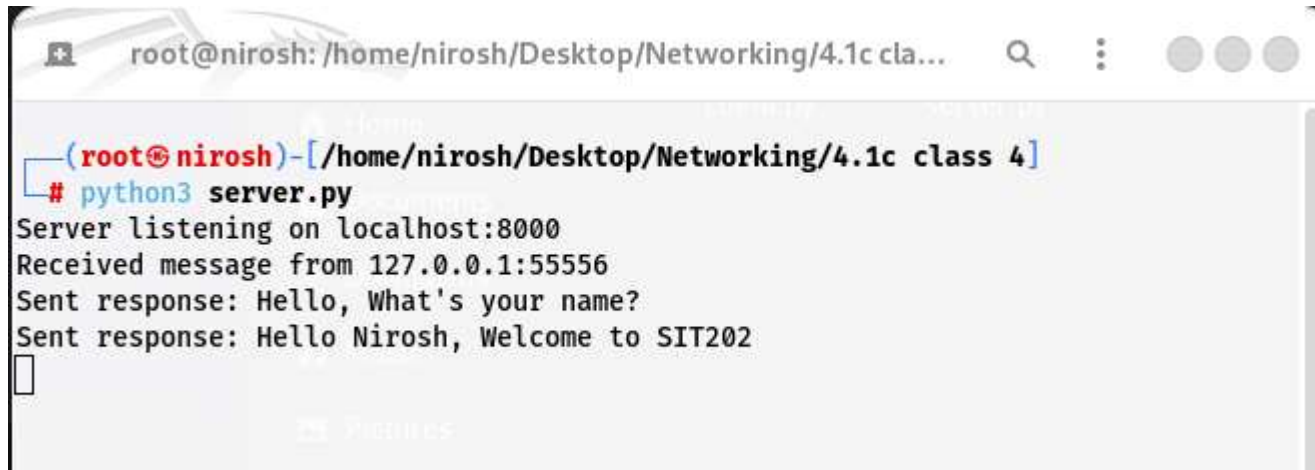
# Receive the response from the server
data, server_address = client_socket.recvfrom(4096)
response = data.decode()
print('Received response: {}'.format(response))

# Get the name from the user and send it to the server
if response == 'Hello, What\'s your name?':
    name = input('Enter your name: ')
    client_socket.sendto(name.encode(), server_address)
    print('Sent name: {}'.format(name))

    # Receive the response from the server
    data, server_address = client_socket.recvfrom(4096)
    response = data.decode()
    print('Received response: {}'.format(response))

# Close the socket
client_socket.close()
```

## Server-side terminal

A screenshot of a terminal window titled 'root@nirosh: /home/nirosh/Desktop/Networking/4.1c cla...'. The terminal shows the execution of 'python3 server.py'. The output indicates the server is listening on localhost:8000, receives a message from 127.0.0.1:55556, and sends two responses: 'Hello, What's your name?' and 'Hello Nirosh, Welcome to SIT202'.

```
(root@nirosh)-[/home/nirosh/Desktop/Networking/4.1c class 4]
# python3 server.py
Server listening on localhost:8000
Received message from 127.0.0.1:55556
Sent response: Hello, What's your name?
Sent response: Hello Nirosh, Welcome to SIT202
█
```

## Client-side terminal

A screenshot of a terminal window titled 'root@nirosh: /home/nirosh/Desktop/Networking/4.1c cla...'. The terminal shows the execution of 'python3 client.py'. The output shows the client sending 'Hello', receiving 'Hello, What's your name?', entering the name 'Nirosh', sending it, and receiving the final response 'Hello Nirosh, Welcome to SIT202'.

```
(root@nirosh)-[/home/nirosh/Desktop/Networking/4.1c class 4]
# python3 client.py
Sent message: Hello
Received response: Hello, What's your name?
Enter your name: Nirosh
Sent name: Nirosh
Received response: Hello Nirosh, Welcome to SIT202

(root@nirosh)-[/home/nirosh/Desktop/Networking/4.1c class 4]
# █
```

## Active class 5: How can I transport my application data reliably? (Module 3)

### Discuss the differences

1.

#### Stop-and-Wait:

When using stop-and-wait, the sender sends one packet, then holds off on sending another until they receive an acknowledgment (ACK). The packet is sent again if the acknowledgment is not received in a certain period of time. Due to the sender's need to wait for an acknowledgment after each packet, this method may result in considerable latency, despite its simplicity.

Consider this as sending one package at a time and delaying sending the next until you get confirmation (such as a signed receipt). It is quite slow, yet it is easy to understand. Just picture a delivery person standing by for each and every box!

This is Good for simple, low-bandwidth situations where slowness isn't a big deal.

#### Go-Back-N:

Go-Back-N limits the number of unacknowledged packets that can be in the pipeline (determined by a window size) and lets the sender send many packets before requiring an acknowledgment. The sender is required to retransmit both the lost packet and any succeeding packets in the event of an error. This technique is more efficient than Stop-and-Wait, but it may be wasteful if failures happen frequently as it may result in several packets being retransmitted needlessly.

The delivery person can accept as many items as they like in this situation without having to wait for approval. The person must return and deliver all subsequent packages in case the first one gets misplaced. Although this is quicker than stop-and-wait, a large number of dropped packets may cause bandwidth to be wasted.

This is a good balance between simplicity and efficiency for moderate data transfers.

#### Selective Repeat:

Like Go-Back-N, Selective Repeat allows numerous packets to be delivered before requiring an acknowledgment. However, when an error occurs, only the particular packet that was lost or incorrect is retransmitted, as opposed to retransmitting all following packets. Because Selective Repeat reduces unnecessary retransmissions, it is therefore more effective than Go-Back-N when mistakes are present.

It's similar to having an intelligent mailbox that knows how to arrange packages.

Packages can be continuously dropped off by the delivery person, and the mailbox will organize them. This is the most efficient approach because only the missing packets are resent. Its implementation is the most complicated, though.

This is Best for high-speed, reliable data transfers where efficiency is crucial.

## 2.

TCP congestion control is crucial because it prevents the network from being overloaded with data at once, which can result in packet loss, higher latency, and worse network performance overall. TCP controls congestion via a number of ways, including:

### **Slow Start**

A TCP connection helps to carefully investigate the network capacity by starting with a small congestion window (cwnd) and increasing it exponentially with each acknowledgment received.

### **Congestion Avoidance**

To prevent overwhelming the network, the congestion window increases more slowly when it hits a threshold.

### **Congestion Avoidance**

Through the retransmission of lost packets and the reduction of the congestion window to prevent new congestion, these processes aid in the speedy recovery from packet losses.

Imagine if every delivery person attempted to utilize the same road at the same time. stuck in the traffic! When too much data tries to get across the internet at once, this is what occurs. Similar to traffic lights, congestion management in TCP ensures smooth data flow by preventing overload. In order to prevent overloading, the system, it keeps an eye on the network and modifies the transmission pace.

Through the use of these techniques, TCP guarantees steady and dependable data delivery by preventing network congestion and ensuring effective network operation.

## Summary for Module 3 active classes 4 and 5

The goal of the Python program change was to improve an earlier client-server UDP communication setup that I had put in place. The prior `client.py` and `server.py` scripts needed to be updated in order to enable a more interactive communication. The protocol was expanded to include a "Hello" and "Hello, What's your name?" exchange of greeting messages, after which the client sends the server their name and receives a welcome message.

The differences in operation between the three main protocols—Stop-and-wait, Go-back-N, and Selective Repeat—were then the focus of the protocol discussion. Every protocol was examined in light of its distinct operational features, benefits, and limitations. In addition, I've covered which of these protocols are most suited, taking into account elements like ease of use, effectiveness, and error-recovery potential.

Investigating TCP congestion control then showed how important it is to preserving the dependability and stability of networks. The conversation emphasized how important congestion management systems are to avoiding system failure, guaranteeing equal distribution of resources, and preventing network overload. One of the key components of TCP's congestion control method is its adaptive nature, which allows it to dynamically modify transmission rates in response to network conditions.

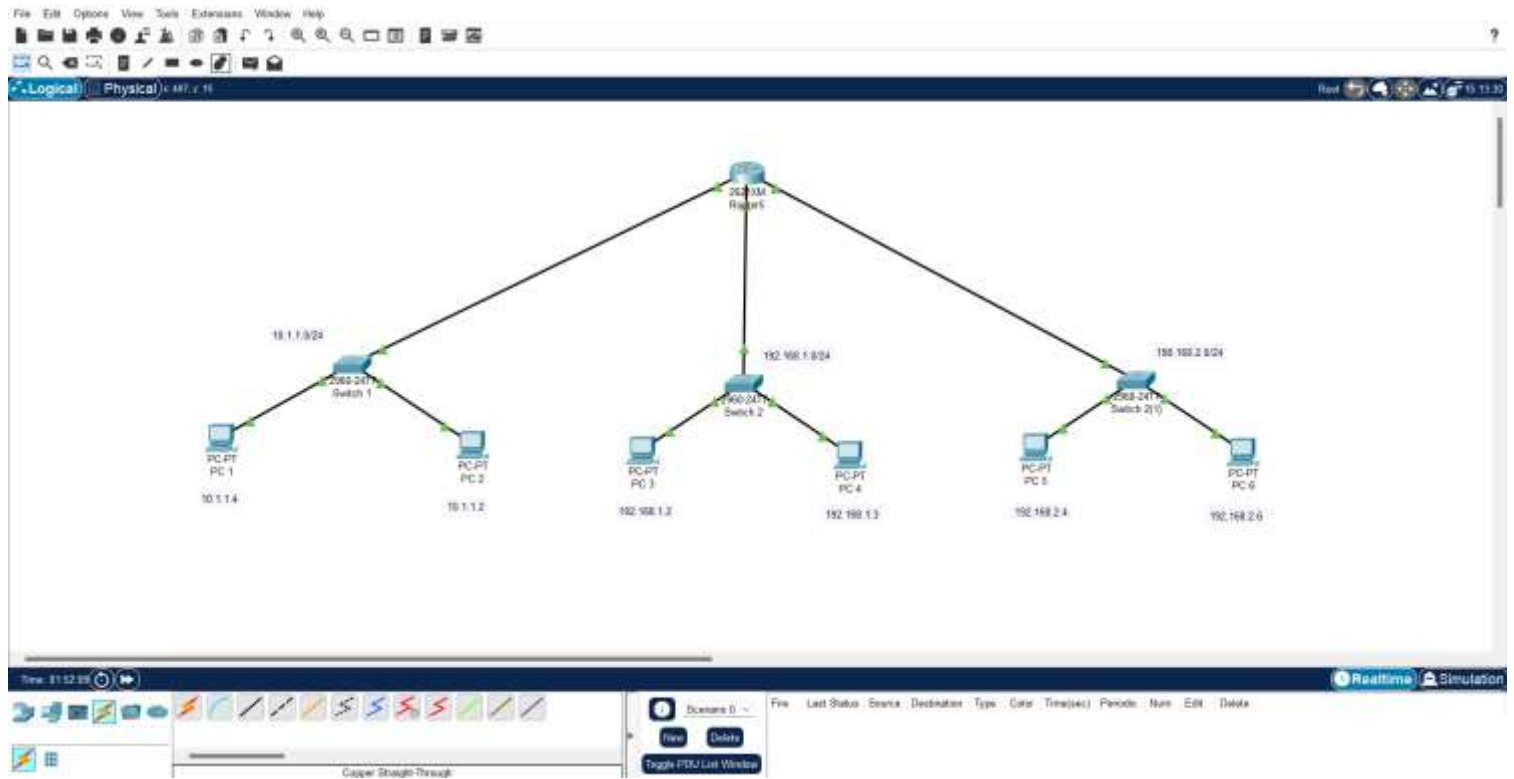
## Reflection of Module 3 active classes 4 and 5

Modifying Python applications offered useful insights into socket programming by stressing message exchange protocols and client-server connectivity. Through the use of UDP sockets to create a simple network communication system, this hands-on activity enhanced knowledge. By examining the operational details and practical applications of protocols, the conversation about them improved understanding. A more detailed understanding of the advantages and disadvantages of each protocol was made possible by contrasting and comparing them, highlighting the importance of choosing the right protocol for a given set of communication requirements.

Examining TCP congestion management brought to light the difficulties in managing network resources in distributed systems, in addition to explaining its complicated mechanics. Gaining insight into the reasoning behind congestion management algorithms highlighted the importance of TCP in providing dependable and effective data transfer across networks, enhancing knowledge of network communication concepts.

## Active class 6: How do I get from My home to Your home - Journey of an IP packet (Module 4)

This is the network implementation I did



I implemented a new LAN and added PC5 and PC6 to it. It has a subnet mask of 192.168.2.0/24. Below are the configurations for the LAN3. I statically gave the new LAN their IP addresses. The PC5 was given the IP address of 192.168.2.4 as asked in the assignment.

PC 5

Physical

Config

Desktop

Programming

Attributes

IP Configuration

X

Interface

FastEthernet0

IP Configuration

DHCP

Static

IPv4 Address

192.168.2.4

Subnet Mask

255.255.255.0

Default Gateway

192.168.2.1

DNS Server

0.0.0.0

IPv6 Configuration

Automatic

Static

IPv6 Address

/

Link Local Address

FE80::209:7CFF:FEC8:4211

Default Gateway

DNS Server

802.1X

Use 802.1X Security

Authentication

MD5

Username

Password



## PC6

The screenshot shows the configuration window for PC6, with the 'Desktop' tab selected. The 'IP Configuration' section is active, showing settings for the 'FastEthernet0' interface. The 'Static' option is selected for both IPv4 and IPv6 configurations. The IPv4 address is 192.168.2.6, the subnet mask is 255.255.255.0, the default gateway is 192.168.2.1, and the DNS server is 0.0.0.0. The IPv6 address is empty, the link local address is FE80::210:11FF:FE82:68C4, and the default gateway and DNS server are empty. The '802.1X' section is also visible, with 'Use 802.1X Security' unchecked, 'Authentication' set to MD5, and 'Username' and 'Password' fields empty.

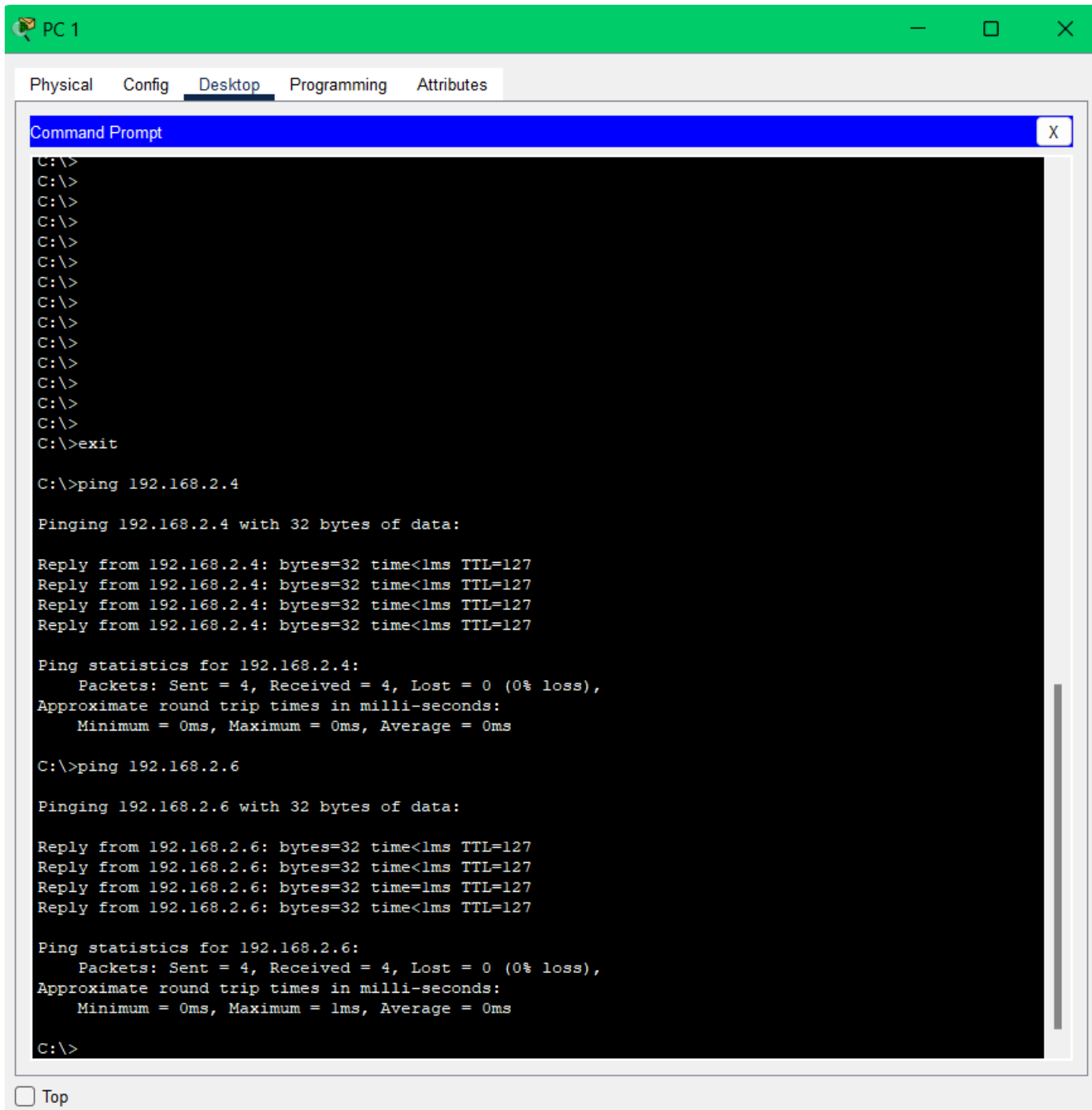
Interface	FastEthernet0
<b>IP Configuration</b>	
<input type="radio"/> DHCP	<input checked="" type="radio"/> Static
IPv4 Address	192.168.2.6
Subnet Mask	255.255.255.0
Default Gateway	192.168.2.1
DNS Server	0.0.0.0
<b>IPv6 Configuration</b>	
<input type="radio"/> Automatic	<input checked="" type="radio"/> Static
IPv6 Address	
Link Local Address	FE80::210:11FF:FE82:68C4
Default Gateway	
DNS Server	
<b>802.1X</b>	
<input type="checkbox"/> Use 802.1X Security	
Authentication	MD5
Username	
Password	

To connect to this LAN3 I had to add another switch and change the router from 1941 to 2621XM.

After changing the router, I had to add the NM-1E that features a single Ethernet port that can connect a LAN backbone which can also support either six PRI connections to aggregate ISDN lines, or 24 synchronous/asynchronous ports.

I also added another PC(PC6) in the LAN3 network.

Here's ping from **PC1 to PC5** and then from **PC1 to PC6**



The screenshot shows a desktop environment for PC1 with a green title bar. The desktop has tabs for Physical, Config, Desktop (selected), Programming, and Attributes. A Command Prompt window is open, displaying the following text:

```
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>exit

C:\>ping 192.168.2.4

Pinging 192.168.2.4 with 32 bytes of data:

Reply from 192.168.2.4: bytes=32 time<1ms TTL=127
Reply from 192.168.2.4: bytes=32 time<1ms TTL=127
Reply from 192.168.2.4: bytes=32 time<1ms TTL=127
Reply from 192.168.2.4: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.2.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 192.168.2.6

Pinging 192.168.2.6 with 32 bytes of data:

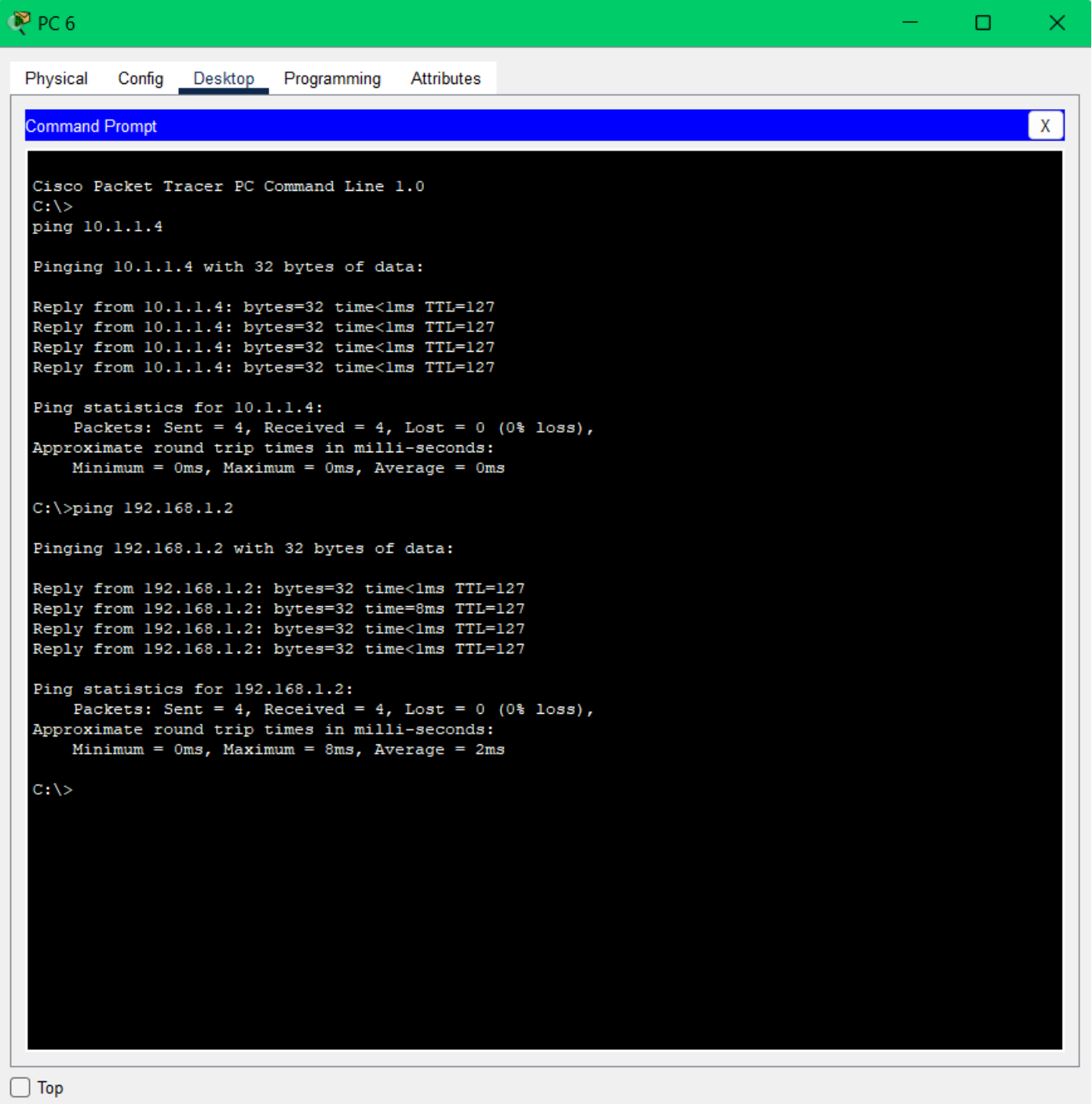
Reply from 192.168.2.6: bytes=32 time<1ms TTL=127
Reply from 192.168.2.6: bytes=32 time<1ms TTL=127
Reply from 192.168.2.6: bytes=32 time=1ms TTL=127
Reply from 192.168.2.6: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.2.6:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>
```

At the bottom left of the desktop, there is a checkbox labeled "Top".

To ensure connectivity from LAN3 to other LANS, I pinged from **PC6(LAN3)** to **PC1(LAN1)** and **PC3(LAN2)**



The screenshot shows a Cisco Packet Tracer interface for PC6. The 'Desktop' tab is selected, displaying a 'Command Prompt' window. The command prompt shows the execution of two ping commands. The first command is 'ping 10.1.1.4', which results in four successful replies with 32 bytes of data, a time of less than 1ms, and a TTL of 127. The statistics show 4 packets sent, 4 received, and 0% loss. The second command is 'ping 192.168.1.2', which also results in four successful replies with 32 bytes of data, a time of less than 1ms, and a TTL of 127. The statistics show 4 packets sent, 4 received, and 0% loss. The command prompt is titled 'Command Prompt' and has a close button (X) in the top right corner. The PC6 window has a green title bar and tabs for 'Physical', 'Config', 'Desktop', 'Programming', and 'Attributes'.

```
Cisco Packet Tracer PC Command Line 1.0
C:\>
ping 10.1.1.4

Pinging 10.1.1.4 with 32 bytes of data:

Reply from 10.1.1.4: bytes=32 time<1ms TTL=127
Reply from 10.1.1.4: bytes=32 time<1ms TTL=127
Reply from 10.1.1.4: bytes=32 time<1ms TTL=127
Reply from 10.1.1.4: bytes=32 time<1ms TTL=127

Ping statistics for 10.1.1.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<1ms TTL=127
Reply from 192.168.1.2: bytes=32 time=8ms TTL=127
Reply from 192.168.1.2: bytes=32 time<1ms TTL=127
Reply from 192.168.1.2: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 8ms, Average = 2ms

C:\>
```

Since the PC6 is from LAN3 and it is pinging properly to PC3 and PC1 from LAN2 and LAN1 respectively, it shows proper connectivity.

## **Summary for Module 4 active class 6**

I added a new PC (PC5) with the IP address 198.168.2.4 to the network established in Activity 3 as part of the "Above and Beyond" activity for Active Class 6. This required setting PC5's IP address in accordance with how connected it is to the current network. Furthermore, I verified that PC1 and PC5 were properly connected by using the "ping" command to test connection. In addition to giving me hands-on experience expanding network capabilities, this effort helped me gain a deeper understanding of IP addressing and network topology design.

## **Reflection of Module 4 active class 6**

The Above and Beyond assignment gave participants the chance to put the networking principles they had learnt in Active Class 6 activities to use and develop further. I was able to solidify my grasp of IP addressing, subnetting, and network communication protocols by adding more devices and checking network connectivity. As I navigated the complexity of network design and configuration, this task also challenged me to use my critical thinking and problem-solving skills.

All things considered, the Above and Beyond challenge helped me cement my understanding of network layer operations and was a useful exercise in practical application.

## Summary and reflection of Module 2

### Summary

I learned a lot about network application architectures in Module 2, with a particular emphasis on the Peer-to-Peer (P2P) and Client-Server models. I studied their architectures, their communication protocols, and the challenges of maintaining these systems. The module distinguished clearly between the benefits and drawbacks of each design by highlighting the situations in which it is most useful.

**Client-Server Architecture:** Clients ask servers for resources or services, and servers respond. This is known as the client-server architecture. For instance, your browser (client) asks a web server for web pages when you visit a website. For centralized control and management, like databases or web services, this design works well.

**Peer-to-peer (P2P) Architecture:** P2P networks, on the other hand, divide up the work among peers, with each peer having the ability to function as a client or a server. File-sharing networks such as BitTorrent serve as an example, where users download files from one another instead of a central server. Although this approach is robust and scalable, its decentralized structure can make it more difficult to administer.

In addition, I looked into inter-process communication (IPC), focusing on the function of sockets and APIs in enabling communication among processes on a network. Understanding how networked applications maintain data interchange and communication—a crucial component of designing and debugging network-based systems—was made possible thanks to this part.

**Sockets:** On a network, sockets serve as endpoints for data transmission and reception. For example, your email client connects to the email server via a socket when you send an email. Sockets are essential to network programming because they guarantee dependable data delivery.

The HTTP protocol was a large portion of the module. I gained knowledge about the way web clients and servers communicate, the statelessness of HTTP, and the differences between non-persistent and persistent connections. This information laid a strong basis for web development and network administration and is essential for comprehending how web pages are requested and served.

**HTTP Protocol:** I gained knowledge about how web clients and servers communicate, how stateless HTTP is, and how non-persistent and persistent connections differ. In order to improve performance, persistent connections reuse the same connection for many requests, whereas non-persistent connections create a new connection for each HTTP request/response combination, such as when loading individual images on a webpage.

Furthermore, the Domain Name System (DNS) was thoroughly covered throughout the module. I looked at its hierarchy and dispersed structure as well as the procedures that translate an IP address from a hostname. It was also interesting to understand the distinctions between recursive and iterative DNS inquiries, as this provided insights into the effective and precise resolution of DNS queries.

**DNS:** DNS converts domain names that are readable by humans, such as `www.example.com`, into IP addresses, such as `192.0.2.1`. This is an iterative or recursive process that involves making hierarchical queries to DNS servers.

In a recursive inquiry, the DNS server fully resolves the query on behalf of the client; in an iterative question, the DNS server responds by referring the user to another server.

In addition to these fundamental subjects, the session covered the fundamentals of network security, emphasizing the value of encrypting data transfers and guarding against frequent dangers such as DNS spoofing and man-in-the-middle attacks. This component is essential for creating safe network applications and preserving the confidentiality and integrity of data.

Network security involved knowing how to safeguard data transfer through authentication and encryption. For example, HTTPS (HTTP Secure) encrypts data between the client and server using SSL/TLS protocols, guaranteeing data integrity and privacy.

All in all, this module has proven to be really helpful. I now have a strong foundation in internet architecture and protocols, which is helpful for creating effective network applications and comprehending how people interact with the online. My recently gained knowledge is directly useful for activities like online communication optimization, secure and dependable network service implementation, and network troubleshooting. My comprehension has greatly improved as a result of this lesson, which has also helped me get ready for more difficult subjects in networking and application development.

## **Reflection**

Gaining a thorough understanding of various network architectures and the HTTP protocol is the module's most important lesson. Particularly helpful have been learning about P2P and Client-Server architectures as well as the HTTP protocol's functioning in a client-server context. This information offers a deeper understanding of the interactions between different network components and the data transfer process across the internet.

The course expanded on my prior understanding of fundamental computer networks by providing a more thorough look at data transport mechanisms, web client-server interactions, and host inter-process communication. It seems like piecing together a complicated puzzle because these pieces fit together perfectly with what I previously understood.

This lesson, in my opinion, was created by the course developers to give students a basic understanding of how the internet functions. Anyone working in information technology or computer science has to know this information. It gives me the abilities I need to optimize online communications, solve network problems, and put safe, dependable network services into place. Not only is academic knowledge stressed, but real-world application in practical settings is as well. A vital first step in becoming a skilled network engineer or web developer is completing this subject.

The introduction to network security was also a useful addition. Protecting data integrity and privacy requires an understanding of fundamental network security concepts like authentication and encryption. This information is especially crucial because network application security is getting harder to achieve and cyber threats are always changing.

My understanding was solidified by the practical exercises and hands-on experiences that were provided throughout the module. Socket programming, network configuration, DNS lookups, and other tasks helped close the knowledge gap between theoretical concepts and practical applications. These exercises improved my knowledge of technology and gave me greater confidence when it came to managing network-related tasks.

The gap between theory and practice was filled up by tasks like configuring DNS servers, simulating HTTP queries, and creating programs to create socket connections. These exercises improved my technical proficiency and gave me more self-assurance when it came to managing network-related chores.

To sum up, this module has improved my understanding of network architecture and internet protocols considerably. It has given me a solid foundation for a future in network engineering or web development and equipped me for more advanced networking and application development issues. The knowledge I've acquired from this course will be extremely helpful as I continue to research networking and work toward becoming a skilled and productive professional.

#### **Some external resources I used**

1. GeeksforGeeks. (2022, September 5). *Application layer in OSI model*. GeeksforGeeks.  
<https://www.geeksforgeeks.org/application-layer-in-osi-model/>
2. Wikipedia contributors. (2024, May 18). *Application layer*. Wikipedia.  
[https://en.wikipedia.org/wiki/Application\\_layer](https://en.wikipedia.org/wiki/Application_layer)
3. Kirvan, P., & Froehlich, A. (2022, March 22). *application layer*. Networking.  
<https://www.techtarget.com/searchnetworking/definition/Application-layer>

# Module 3 Summary and Reflection

## Summary

### Understanding of TCP / IP model

#### Transport Layer Overview:

This module covers the manner in which data is transmitted across networks reliably, and focuses on the role of Transmission Control Protocol (TCP) options vs. User Datagram Protocol (UDP) options.

The transport vs. network layer: The module starts off by explaining what each of the layers are responsible for. The network layer routes packets to and from different hosts, while the transport layer facilitates communication between processes that are running on those hosts. This can be drawn as follows taking the analogy at your home, where transport layer will make sure mother hands over messages (letters) to its children(processes) directly in house(host), and network layer is post which will deliver letters between houses.

#### TCP and UDP Protocols

Transmission Control Protocol /65535

Reliable – provides a guaranteed delivery mechanism using an in-order protocol.

Contains mechanisms to control congestion, check flow and establish a connection.

This is perfect for embeddings which require high amount of stability like web browsing over emails.

#### UDP (User Datagram Protocol)

provides an unreliable, message-oriented communication protocol requiring no implementation of message orientation or reliabilityundanceReadStreampeersoctets flow control.

Lesser latency as no connection establishment is involved.

Use this when speed is an issue and losing a few packets of data will not make your application mind. It's ideal in situations where you may be streaming multimedia.

#### Multiplexing and Demultiplexing

##### Multiplexing

The process of taking data from more than one socket and turning it into a “message” with some transport headers on top.

These headers ensure that each data segment being transmitted can be addressed correctly at the receiving end.



## **Demultiplexing**

The process in reverse, where headers are used by the transport layer on the receiver to pass down data appropriately to application processes.

## **Connection-Oriented vs Connectionless**

### **Connectionless Communication (UDP)**

With connectionless communication (UDP), data is delivered without first establishing a connection. This method works well for quick, effective transmission in situations where a small amount of data loss is acceptable.

### **Connection-Oriented Communication (TCP)**

Prior to data transfer, a dependable link is made, guaranteeing error-free and well-organized delivery. In order to establish a connection, a four-way handshake and a three-way handshake are required.

## **TCP Functions**

The segment structure of TCP, the usage of sequence and acknowledgment numbers, and the methods for opening and closing connections all contribute to an explanation of how the protocol functions. To create a connection, the TCP three-way handshake uses the SYN, SYN-ACK, and ACK messages. To close a connection, the four-way handshake uses the FIN and ACK messages.

## **Practical Examples**

The module provides practical examples, such as:

### **TCP's Three-Way Handshake**

- Step 1: The client sends a SYN (synchronize) message to the server.
- Step 2: The server responds with a SYN-ACK message.
- Step 3: The client sends an ACK (acknowledgment) message to establish the connection.

## **UDP in Action**

Streaming services like Netflix use UDP to ensure fast delivery of data packets, where occasional loss of data is not critical.

## Additional Concepts

### Round Trip Time (RTT) and Timeout Calculations

TCP calculates the timeout interval using estimated RTT and variance (DevRTT), ensuring efficient retransmission of lost packets.

Example Formula:  $\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$ .

### ACK Generation and Retransmission

TCP uses cumulative acknowledgments and fast retransmit strategies to handle lost packets and ensure data integrity.

The transport layer in the TCP/IP model is crucial for managing data transfer between devices on a network. Here are the primary uses and functionalities of the transport layer:

### Reliable Data Transfer (TCP)

- **Ensures Data Integrity:** By using mechanisms like error detection and correction, TCP ensures that data sent from one device is received accurately by another.
- **In-Order Delivery:** TCP guarantees that data packets arrive in the same order they were sent, which is essential for applications like web browsing and email.
- **Retransmission of Lost Packets:** If a data packet is lost during transmission, TCP retransmits the packet, ensuring that no data is lost.
- **Example:** When downloading a file, TCP ensures the complete file is received correctly, even if some packets need to be retransmitted.

### Connection Establishment and Termination (TCP)

- **Three-Way Handshake:** TCP uses a three-way handshake (SYN, SYN-ACK, ACK) to establish a connection, ensuring both the sender and receiver are ready for data transfer.
- **Four-Way Handshake:** To terminate a connection, TCP uses a four-way handshake (FIN, ACK), gracefully closing the communication channel.
- **Example:** Initiating and ending a video call, where a stable connection is required to start and finish the call properly.

## Flow Control (TCP)

- **Manages Data Flow:** TCP uses flow control mechanisms to prevent the sender from overwhelming the receiver with too much data at once.
- **Example:** In an online meeting, flow control ensures the video and audio data are transmitted smoothly without overloading the network or the receiver's device.

## Congestion Control (TCP)

- **Adjusts Transmission Rate:** TCP dynamically adjusts the data transmission rate based on network congestion, preventing network overload and ensuring efficient data transfer.
- **Example:** Streaming a high-definition video, where TCP adjusts the data rate to avoid buffering due to network congestion.

## Unreliable, Low-Latency Data Transfer (UDP)

- **Faster Data Transmission:** UDP provides quick data transmission without the overhead of connection establishment, making it suitable for time-sensitive applications.
- **No Guaranteed Delivery:** Unlike TCP, UDP does not guarantee that all packets will be delivered or in order, which is acceptable for certain applications.
- **Example:** Live video streaming or online gaming, where speed is more critical than ensuring every packet arrives in order.

## Multiplexing and Demultiplexing

- **Handles Multiple Connections:** The transport layer manages data from multiple applications by adding headers to distinguish between different data streams.
- **Directs Data to Correct Application:** On the receiving end, it uses these headers to deliver data to the correct application process.
- **Example:** Browsing the web while listening to music online, where the transport layer ensures both activities receive their respective data streams without interference.

## Port Number Management

- **Identifies Applications:** The transport layer uses port numbers to identify different applications and services on a device.
- **Directs Traffic:** Ensures that data meant for a specific application is correctly directed to it.
- **Example:** An email application using port 25 for sending emails (SMTP) and port 110 for receiving emails (POP3).

By understanding these uses, it's clear how the transport layer is essential for efficient, reliable, and accurate data transfer across networks, enabling various applications and services to function seamlessly.

### Some external resources I referred to

1. GeeksforGeeks. (2023a, July 21). *TCP/IP model*. GeeksforGeeks. <https://www.geeksforgeeks.org/tcp-ip-model/>
2. *The transport layer in TCP/IP model*. (n.d.). <https://www.tutorialspoint.com/The-Transport-Layer-in-TCP-IP-Model>
3. Wikipedia contributors. (2024b, May 19). *Transport layer*. Wikipedia. [https://en.wikipedia.org/wiki/Transport\\_layer](https://en.wikipedia.org/wiki/Transport_layer)
4. *Transmission Control Protocol (TCP) (article)* | Khan Academy. (n.d.). Khan Academy. <https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/transmission-control-protocol--tcp>

## Reflection

### What is the most important thing you learnt in this module?

The ability to fully understand TCP and UDP's functions and roles in data transmission is the module's most important takeaway. I now recognize the significance of TCP's dependability features, which are essential for data integrity-requiring applications and include flow control and congestion control. On the other hand, seeing how quick and easy UDP is to use makes it an excellent choice for real-time applications such as streaming videos.

### How does this relate to what you already know?

This module expands on what I already know about the OSI model, especially with regard to the transport layer's function in maintaining end-to-end communication. By stressing the real-world uses and technical specifics of TCP and UDP, it deepens my comprehension.

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

The importance of the transport layer in network communications is highlighted by the course team's attention on it. Developing reliable network applications and resolving network problems require a deep understanding of this layer. In the fields of computer science and network engineering, where dependable and efficient data transfer is critical, this knowledge is fundamental.

I learned more about the transport layer and its crucial function in network communications from this module. I used to have a rudimentary understanding of the functions of the transport layer, but now I understand the distinctions between UDP (User Datagram Protocol) and TCP (Transmission Control Protocol) and why each is crucial for certain applications.

Accurate and sequential data transmission is guaranteed by TCP. For applications where data integrity is critical, this is vital. Sending an email, for instance, requires that the message be received exactly as sent. TCP does this by establishing a connection through a three-way handshake (SYN, SYN-ACK, ACK) prior to the start of data transfer. This connection-oriented strategy works similarly to a phone call in which both sides signal that they are prepared to speak before beginning.

Another scenario where TCP's dependability is crucial is web browsing. The browser uses TCP to request data from the server when I load a webpage, making sure that every component is supplied appropriately. By modifying the data transfer rate, avoiding overload, and guaranteeing smooth delivery, TCP controls network congestion.

Since UDP is all about speed, it's suitable for real-time applications where speed is more important than accuracy, such as online gaming or live video streaming. Retransmitting missing data causes delays that are more evident in a live feed than a few lost frames. UDP makes it possible for the stream to run smoothly, which improves the watching experience.

Minimal delay is necessary for quick action in online gaming. Because UDP has a minimal overhead, games may send short, frequent packets quickly, allowing player actions to be reflected in the game instantly. Even though a few packets may be dropped, the game can handle this without having a big impact on the gameplay.

Data from several applications is combined into a single stream during multiplexing, then at the destination, it is demultiplexed to divide the data back into independent streams. The effective use of network resources is ensured by this procedure. Multiplexing guarantees that, for instance, concurrently downloading a file and streaming a movie are handled effectively and sent to the appropriate apps on my device.

Three-way handshakes are used to establish connections while four-way handshakes are used to close them under TCP's connection management protocol. By doing this, dependable communication over arguably unstable networks is ensured. Every data packet is tracked and received in the correct order thanks to sequence and acknowledgment numbers. TCP's retransmission capabilities preserve communication integrity by sending the lost data again in the event of a packet loss.

Comprehending TCP and UDP facilitates the diagnosis of network problems. Tools that detect packet loss or congestion can be used to modify TCP settings or choose a better protocol if a file download is taking a long time.

In addition to improving my theoretical understanding, this module has given me useful insights into how networks operate. This will make a big difference in how network applications are developed and optimized, helping to make sure they satisfy the necessary reliability and performance requirements.

# Summary and Reflection of Module 4

## Summary

Segments must be transported from the transmitting host to the receiving host via the network layer. Segments are delivered to the transport layer protocol after being encapsulated into datagrams and passed to the link layer. All Internet-connected devices, such as hosts and routers, have this layer.

## Routing and Forwarding

**Forwarding:** This local procedure involves sending packets from the input link of a router to the relevant output link. The forwarding mechanism chooses which router outbound connection will deliver a packet closer to Los Angeles, for instance, if it arrives at the router from a computer in New York that is intended for a server in Los Angeles.

**Routing:** This mechanism on the network as a whole decides how packets get from their source to their destination. In order to determine the most effective path, routing uses algorithms like BGP (Border Gateway Protocol) and OSPF (Open Shortest Path First). It's similar to figuring out the finest highways and routes to take when traveling from New York to Los Angeles.

## Data Plane and Control Plane

### Planes of Data and Control

**Data plane:** It is responsible for managing the packet forwarding process. For instance, the data plane utilizes a forwarding table to determine which output port to send a packet to when it reaches a router. In order to maintain smooth traffic flow, this process moves quite quickly.

**Control Plane:** Oversees decisions and logic for the entire network. The control plane modifies the forwarding tables of all impacted routers, for instance, if a new route proves to be more effective as a result of modifications in network traffic. By doing this, packets are guaranteed to travel the best route throughout the network.

## Models of Network Services

Various service models are provided via network architectures:

**Best-Effort Service:** This model, which is commonly used on the Internet, does not ensure delivery, timing, order, or bandwidth. An email sent, for instance, is sent via the network as best it can, but there's no assurance on when it will reach or even in what sequence.

## Router Design

Several parts make up a generic router:

Accept incoming packets on the input ports. A router that is connected to several PCs, for example, will have multiple input ports that handle data from various sources.

Input ports are connected to output ports via switching fabric. It makes sure packets go quickly from input to output by acting as a fast internal network inside the router.

Packets should be sent to their next location via output ports. For instance, a packet that is meant for a server in a different city leaves through the proper output port after being processed by the switching fabric.

Routing Processor: Manages operations of the control plane, such as the update of routing tables in response to network conditions.

## Subnetting and IP Addressing

**IP addressing:** A 32-bit (in IPv4) address is assigned to each device connected to a network. A computer might, for instance, have the IP address 192.168.1.1.

**Subnetting:** Subdivides a huge network into more manageable sub-networks. One example of an IP address that defines a subnet with up to 256 addresses is 192.168.1.0/24.

**Subnet Mask:** Assists in distinguishing between the network and host portions of an IP address. For instance, when 192.168.1.1 is assigned the subnet mask 255.255.255.0, the first three octets (192.168.1) identify the network, and the final octet (1) identifies the particular host.

**CIDR Notation:** Allows flexible IP address assignment. For example, 192.168.1.0/24 means the first 24 bits are the network part, and the remaining 8 bits can be used for hosts.

## NAT and DHCP

The Dynamic Host Configuration Protocol, or DHCP, allocates IP addresses to devices dynamically. For instance, DHCP chooses an IP address from the pool of addresses available to assign to a laptop when it connects to a Wi-Fi network.

A single public IP address can be shared by several devices connected to the same local network thanks to NAT (Network Address Translation). When many devices, such as laptops and phones, connect to the Internet through a single IP address, a home router can utilize network address translation (NAT) to make it look as though they are all using the same IP address.



## IPv6

IPv6 overcomes IPv4's drawbacks, such as the address space exhaustion. It makes use of 128-bit addresses, which greatly expands the pool of potential addresses. An IPv6 address, for instance, appears like this: 2001:0db8:85a3:0000:0000:8a2e:0370:7334. This increases the number of addresses available while also streamlining certain aspects of routing and address setting.

It is essential to comprehend these ideas in order to create, manage, or debug a network. They offer the fundamental information needed to guarantee effective and safe data transfer across networks.

### A thorough comprehension of network operations

1. **Effective Data Transmission:** To ensure effective data transmission across networks, one must have a thorough understanding of the network layer and its constituent elements, such as forwarding and routing. For instance, I can enhance data flow and lower latency by optimizing network pathways by understanding how packets are routed from one router to another.
2. **Improved Troubleshooting Skills:** I can identify and resolve network problems more quickly now that I have a firm understanding of how the data and control planes function. To find and fix the bottleneck, for example, if there is a delay in data transmission, I can examine the routing tables and forwarding procedures.

### Useful Implementation in Network Architecture and Administration

3. **Efficient Network Design:** Subnetting and IP addressing expertise enable effective network administration and design. For instance, I may divide a network into manageable subnets and improve performance and security by utilizing CIDR notation and subnet masks. This is especially helpful in large enterprises where effective management of network traffic is required.
4. **Dynamic IP Management:** To manage dynamic IP addresses and guarantee uninterrupted connectivity, one must comprehend DHCP and NAT. For instance, DHCP can streamline network management in a corporate setting by automatically allocating IP addresses to devices as soon as they connect to the network. By hiding internal IP addresses, NAT improves network security while preserving public IP addresses.

### Changing to Meet the Needs of Modern Networking

5. **IPv6 Proficiency:** Understanding the differences and benefits of IPv6 is essential as we go from IPv4 to IPv6. IPv6 has a far bigger address space than IPv4, which helps to overcome its constraints, including address exhaustion. I can manage and create scalable, future-proof networks because I understand IPv6.
6. **Network service model implementation:** Different network service models offer varying degrees of Quality of Service (QoS). Examples of these models are best-effort, IntServ, and DiffServ. I can implement the proper service model based on the network's requirements by comprehending these models. For instance, IntServ can be used in a VoIP service to ensure the dependability and quality of voice data transmission.

## Practical Uses

**7. Practical Experience and Empirical Models:** Participating in exercises and simulations strengthens theoretical understanding and gives practical experience. For instance, setting up subnets, evaluating packet flow, and building routing tables in a controlled environment get me ready for situations where these abilities are required in the real world.

**8. Foundational Information for sophisticated Topics:** This module provides the groundwork for topics in networking that are more sophisticated. For instance, knowing the fundamentals of forwarding and routing helps me when I study more about advanced routing protocols like OSPF and BGP.

All things considered, this module is essential for anyone working in networking or IT. It offers fundamental information and useful abilities that are directly related to network management, design, and troubleshooting. Whether it's maximizing network performance, guaranteeing effective IP address management, or being ready for IPv6 in the future, the ideas covered in this module are essential for a successful networking career.

## External resources I referred to

1. *Control Plane vs. Data Plane: What Are The Differences?* / Splunk. (n.d.). Splunk.  
[https://www.splunk.com/en\\_us/blog/learn/control-plane-vs-data-plane.html](https://www.splunk.com/en_us/blog/learn/control-plane-vs-data-plane.html)
2. GeeksforGeeks. (2021, July 27). *Difference between Control Plane and Data Plane*. GeeksforGeeks.  
<https://www.geeksforgeeks.org/difference-between-control-plane-and-data-plane/>
3. Courtois, S. (2023, July 17). *Data Plane vs. Control Plane: What's the Difference?* SnapLogic.  
<https://www.snaplogic.com/blog/data-plane-vs-control-plane-whats-the-difference>
4. Kong. (n.d.). *Control Plane vs. Data Plane: What's the Difference?* Kong Inc.  
<https://konghq.com/blog/learning-center/control-plane-vs-data-plane>

## Reflection

### Important Thing I Learned in This Module:

The operation of the data plane within the network layer is the most significant thing I have learned. Having a thorough understanding of each router's packet forwarding process aids in ensuring effective network routing. The relevance of their different responsibilities is shown, for instance, when one considers how quickly the data plane operates (in nanoseconds) as opposed to the control plane (in milliseconds).

### How This Relates to What I Already Know:

My knowledge of computer networks is strengthened by this lesson, which digs deeper into the technical facets of data management and transmission. For example, although I was familiar with IP addresses, I now comprehend the specifics of CIDR and subnetting, which are critical for effective network architecture and IP address management.

### Why My Course Team Wants Me to Learn This:

In order for us to have a strong foundation in networking ideas, the course team wants students to master this material. Any IT professional has to know this stuff since it helps us troubleshoot and optimize network performance. For instance, establishing and maintaining safe, effective networks in both home and business settings requires a solid understanding of DHCP and NAT.

Personal Learning Experience: Playing through different simulations, such setting up routing tables and examining packet flow, helped me to better comprehend theory. I had trouble understanding subnetting concepts at first, but I was able to get past this obstacle by using more materials and practical experience. For instance, establishing subnet masks and allocating IP addresses during lab exercises aided in my comprehension. I learned the value of paying close attention to details and the significance of every parameter in network operations through misconfigurations in routing simulations.

As I think back on this module, I see how important these ideas are to networking in the real world. My future profession will benefit greatly from the technical knowledge I gained here, since it will equip me with the necessary abilities to manage, construct, and debug intricate networks. Through my comprehension of the finer points of the network layer, I am better equipped to guarantee secure and effective data transfer in any kind of IT environment. For instance, network performance and security can be greatly improved by understanding how to control IP address allocation and optimize router configurations.