1. Description :

The reason of this PBL is to Implementation of Network Protocols such as TCP, UDP, or Ethernet

in MATLAB

2. Introduction to User Datagram Protocol (UDP) :

The User Datagram Protocol (UDP) is a connectionless transport layer protocol used in

computer communication networks. It is part of the Internet Protocol (IP) suite and is often

employed when low overhead and reduced latency are more critical than guaranteed delivery and

error recovery. Here are key features and characteristics of the User Datagram Protocol (UDP) in

computer communication networks:

1. Connectionless Protocol:

• UDP is a connectionless protocol, meaning it does not establish a connection before

sending data and does not guarantee the reliable delivery of data.

• There is no three-way handshake as seen in connection-oriented protocols like TCP.

2. Minimal Header Overhead:

• UDP has a minimal header compared to connection-oriented protocols like TCP.

This results in lower overhead and reduced processing requirements.

• The UDP header includes source and destination port numbers, length information,

a checksum for error detection, and the data payload.

3. Unreliable Delivery:

• UDP provides best-effort delivery, meaning it does not guarantee that data will be

delivered to the destination, and there is no mechanism for acknowledgment or

retransmission.

• Applications using UDP must handle any necessary error detection and correction

independently.

4. Low Latency:

• Because of its connectionless nature and minimal overhead, UDP is suitable for

applications where low latency is crucial.

• Real-time applications such as voice over IP (VoIP), online gaming, and streaming

media often use UDP to minimize delays.

5. No Flow Control:

• UDP does not implement flow control mechanisms to manage the rate of data

transmission. This lack of flow control allows applications to send data at their

desired pace without being regulated by the protocol.

6. Broadcast and Multicast Support:

• UDP supports both broadcast and multicast communication. This is beneficial for

scenarios where a single message needs to be sent to multiple recipients

simultaneously.

7. Examples of UDP Applications:

• DNS (Domain Name System) uses UDP for name resolution queries.

• DHCP (Dynamic Host Configuration Protocol) employs UDP for obtaining IP

addresses and configuration information.

• Streaming media, real-time communication applications, and online gaming often

use UDP for its low latency characteristics.

8. Checksum for Error Detection:

• UDP includes a simple checksum field to detect errors in the transmitted data.

However, UDP does not provide mechanisms for error recovery.

In summary, UDP is a lightweight, connectionless transport layer protocol that prioritizes

low latency and reduced overhead. It is suitable for applications where occasional data loss is

acceptable, and real-time communication is more critical than guaranteed delivery. Developers

need to implement their own error-handling mechanisms if required, as UDP does not provide

built-in error recovery features.



2. Advantages of UDP :

UDP (User Datagram Protocol) offers several advantages over TCP (Transmission Control

Protocol) in certain scenarios due to its simplicity and lower overhead. Here are some key

advantages of UDP:

1. Low Overhead:

• UDP has a smaller header size compared to TCP, which reduces the amount of

overhead per packet. This makes UDP more efficient for applications that require

minimal protocol overhead, especially for small data transfers or real-time

communication.

2. No Connection Establishment:

• UDP is connectionless, meaning it does not require a connection setup process like

TCP's three-way handshake. This results in lower latency for data transmission

since there is no initial negotiation required before sending data.

• For applications where connection setup time is a concern, such as real-time

streaming, gaming, or VoIP (Voice over Internet Protocol), UDP's lack of

connection establishment overhead can be advantageous.

3. No Flow Control or Congestion Control:

• Unlike TCP, UDP does not implement flow control or congestion control

mechanisms. This means that UDP does not regulate the rate at which data is sent

or adjust transmission rates based on network conditions.

• For applications where low-latency and real-time data delivery are prioritized over

reliability and congestion avoidance, such as online gaming or live video streaming,

UDP's lack of flow and congestion control can be advantageous.

4. Simplicity:

• UDP is simpler and easier to implement compared to TCP, as it does not involve

complex connection management, sequencing, or retransmission mechanisms.

• This simplicity makes UDP well-suited for applications that require minimal

protocol overhead and can tolerate packet loss or out-of-order delivery, such as

real-time multimedia streaming or simple client-server communication.

5. Broadcast and Multicast Support:

• UDP supports broadcast and multicast communication, allowing a single packet to

be sent to multiple recipients simultaneously.

• This feature is useful for applications that need to distribute data to multiple

recipients efficiently, such as multimedia streaming, online gaming, or network

monitoring.

3. Disadvantages of UDP :

While UDP (User Datagram Protocol) offers advantages in terms of simplicity and low

overhead, it also comes with several disadvantages, particularly in scenarios where reliability and

ordered delivery are essential. Here are some key disadvantages of UDP:

1. Unreliable Data Delivery:

• UDP does not guarantee delivery of packets. Unlike TCP, which includes

mechanisms for acknowledging and retransmitting lost packets, UDP simply sends

packets without any assurance of delivery.

• This lack of reliability can result in packet loss, duplication, or out-of-order

delivery, especially in networks with high congestion or packet loss rates.

2. No Flow Control or Congestion Control:

• UDP does not implement flow control or congestion control mechanisms. This

means that it does not regulate the rate at which data is sent or adjust transmission

rates based on network conditions.

• In congested networks, UDP packets may be dropped or delayed, leading to

degraded performance or loss of data.

3. No Error Detection or Correction:

• UDP does not include error detection or correction mechanisms like TCP's

checksum and sequence number verification.

• As a result, UDP packets may be corrupted during transmission, and there is no

mechanism to detect or recover from errors. Applications must handle error

detection and recovery independently, if required.

4. No Connection Establishment:

• UDP is connectionless, meaning it does not require a connection setup process like

TCP's three-way handshake. While this reduces latency, it also means that there is

no initial negotiation or authentication between sender and receiver.

• This lack of connection establishment can make UDP susceptible to various

security threats, such as IP spoofing, denial-of-service attacks, and unauthorized

access.

5. Limited Support for Multicast and Broadcast:

• While UDP supports multicast and broadcast communication, it lacks built-in

mechanisms for managing group membership, reliability, and flow control in

multicast scenarios.

• Applications relying on UDP for multicast or broadcast communication may

require additional protocols or mechanisms to ensure efficient and reliable delivery

to multiple recipients.

4. Applications of UDP :

User Datagram Protocol (UDP) is often chosen for applications that prioritize low latency

and can tolerate occasional data loss. UDP's connectionless nature and minimal overhead make it

suitable for scenarios where speed and responsiveness are crucial. Here are some common

applications that use UDP:

1. Real-Time Communication:

• VoIP (Voice over IP): VoIP applications, such as voice and video calls, often use

UDP due to its low latency. While occasional packet loss might occur, the focus is

on real-time communication.

2. Streaming Media:

• Video Streaming: UDP is commonly used for streaming video content, especially

in live broadcasts or real-time events, where low latency is more critical than

guaranteed delivery.

3. Online Gaming:

• Multiplayer Online Games:Online gaming applications often use UDP for its low

latency and responsiveness. Real-time interactions between players benefit from

the quick transmission of game data.

4. DNS (Domain Name System):

• DNS Queries:DNS typically uses both UDP and TCP, but DNS queries, especially

for simple requests, are often handled using UDP due to its lower overhead and

faster response times.

5. DHCP (Dynamic Host Configuration Protocol):

• IP Address Allocation: DHCP, which is used for dynamically assigning IP

addresses to devices on a network, uses UDP during the initial configuration phase.

6. SNMP (Simple Network Management Protocol):

• Network Monitoring:SNMP, used for managing and monitoring network devices,

often utilizes UDP for certain operations where real-time data updates are essential.

7. Wake-on-LAN:

• Remote Device Wakeup: Wake-on-LAN, a protocol used to remotely wake up a

device over a network, often relies on UDP for its simplicity and efficiency.

8. TFTP (Trivial File Transfer Protocol):

• Simple File Transfers: TFTP, a lightweight file transfer protocol, uses UDP for

its simplicity and minimal overhead, although it lacks the reliability features of

TCP.

9. NTP (Network Time Protocol):

• Time Synchronization: NTP, used for synchronizing the clocks of devices on a

network, can use both UDP and TCP. UDP is often preferred for its low overhead.

5. Introduction to Transmission Control Protocol (TCP) :

The Transmission Control Protocol (TCP) is a connection-oriented transport layer protocol

in computer communication networks. It plays a crucial role in providing reliable and ordered

delivery of data between communicating devices. TCP is part of the Internet Protocol (IP) suite

and is widely used for various applications, including web browsing, file transfer, and email. Here

are key features and aspects of TCP in computer communication networks:

1. Connection-Oriented Communication:

• TCP establishes a connection before data exchange begins. This involves a three-

way handshake process between the sender and receiver to set up parameters for

reliable communication.

2. Reliable Data Delivery:

• TCP ensures reliable and ordered delivery of data. It uses acknowledgments,

sequence numbers, and retransmission mechanisms to guarantee that data reaches

the destination correctly and in the proper order.

3. Flow Control:

• TCP implements flow control mechanisms to manage the rate of data transmission.

This prevents congestion and ensures that the sender does not overwhelm the

receiver or the network with data. Window-based flow control is commonly used.

4. Connection Establishment and Termination:

• Before data exchange, TCP performs a connection establishment process using a

three-way handshake (SYN, SYN-ACK, ACK).

• After data exchange is complete, TCP ensures a graceful connection termination

using a four-way handshake (FIN, ACK, FIN, ACK).

5. Full Duplex Communication:

• TCP supports full-duplex communication, allowing data to be transmitted in both

directions simultaneously. This is achieved by having separate sequence numbers

for data sent in each direction.

6. Acknowledgments and Retransmissions:

• TCP uses acknowledgments (ACKs) to confirm the receipt of data. If an

acknowledgment is not received within a specified timeout period, the sender

retransmits the data.

• Selective acknowledgment (SACK) may be used to inform the sender about

specific packets that need retransmission.

7. Congestion Control:

• TCP incorporates congestion control mechanisms to adapt to network conditions

and prevent congestion collapse. This includes adjusting the transmission rate

based on perceived network congestion.

8. Three-Way Handshake:

• The three-way handshake involves the following steps:

1. SYN (Synchronize):Initiates a connection and includes an initial sequence

number.

2. SYN-ACK (Synchronize-Acknowledge): Acknowledges the SYN and

includes its own initial sequence number.

3. ACK (Acknowledge): Confirms the receipt of the SYN-ACK and

completes the connection setup.

9. Four-Way Handshake for Connection Termination:

• The four-way handshake involves the following steps:

1. FIN (Finish): Initiates the connection termination process.

2. ACK (Acknowledge): Acknowledges the FIN from the other side.

3. FIN (Finish): Initiates the termination on the other side.

4. ACK (Acknowledge): Acknowledges the FIN from the other side,

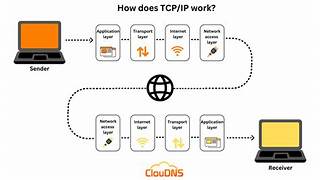
completing the termination.

TCP is well-suited for applications that require reliable, ordered data delivery, such as web

browsing, email, and file transfer. While it introduces additional overhead compared to

connectionless protocols like UDP, its features make it a fundamental protocol for various

communication scenarios on the Internet.



6. Advantages of TCP :

TCP (Transmission Control Protocol) offers several advantages over other protocols,

particularly in scenarios where reliable, ordered, and error-checked delivery of data is crucial. Here

are some key advantages of TCP:

1. Reliable Data Delivery:

• TCP ensures reliable delivery of data by employing mechanisms such as

acknowledgments, sequence numbers, and retransmissions.

• It uses acknowledgments to confirm the receipt of data segments, and retransmits

any lost or corrupted segments until they are successfully received by the

destination.

• This reliability ensures that data arrives intact and in the correct order, making TCP

suitable for applications that require guaranteed delivery, such as file transfer and

web browsing.

2. Ordered Delivery:

• TCP maintains the order of data segments sent by the sender and reassembles them

in the correct order at the receiver.

• This ordered delivery ensures that data is delivered to the application layer in the

same sequence as it was sent, which is essential for applications that rely on the

correct ordering of data, such as streaming media or database transactions.

3. Flow Control:

• TCP implements flow control mechanisms to prevent the sender from

overwhelming the receiver with data.

• It uses a sliding window approach, where the receiver advertises its available buffer

space to the sender, regulating the rate of data transmission.

• This flow control mechanism ensures efficient data transfer without causing

congestion or packet loss, optimizing network performance.

4. Congestion Control:

• TCP employs congestion control algorithms to prevent network congestion and

ensure fair sharing of network resources among competing flows.

• It dynamically adjusts the transmission rate based on network conditions, such as

packet loss, latency, and available bandwidth, to avoid network congestion and

optimize throughput.

• TCP's congestion control mechanisms help maintain network stability and prevent

the degradation of service quality during periods of high traffic.

5. Error Detection and Correction:

• TCP includes error detection mechanisms, such as checksums, to detect errors in

transmitted data segments.

• In the event of data corruption or loss, TCP retransmits the affected segments until

they are successfully received and acknowledged by the destination.

• This error detection and correction capability ensure the integrity of transmitted

data and minimize the likelihood of data corruption or loss during transmission.

6. Connection-Oriented Communication:

• TCP establishes a connection between the sender and receiver before data exchange

begins, ensuring reliable communication.

• The connection setup process involves a three-way handshake, which includes

SYN (synchronize), SYN-ACK (synchronize-acknowledge), and ACK

(acknowledge) segments.

• This connection-oriented communication model provides a reliable and secure

channel for data transmission, protecting against data interception and tampering.

7. Disadvantages of TCP :

While TCP (Transmission Control Protocol) offers numerous advantages, it also has some

disadvantages, particularly in scenarios where low latency or minimal overhead are prioritized

over reliability. Here are some key disadvantages of TCP:

1. High Overhead:

• TCP has higher overhead compared to other protocols like UDP (User Datagram

Protocol) due to its extensive header size and connection management

mechanisms.

• The TCP header includes various control fields, such as sequence numbers,

acknowledgment numbers, and control flags, which increase the size of each

packet.

• This high overhead can lead to increased network congestion and reduced

efficiency, especially for applications that require minimal protocol overhead,

such as real-time communication or low-bandwidth networks.

2. Connection Setup Time:

• TCP requires a connection setup process before data exchange begins, involving a

three-way handshake (SYN, SYN-ACK, ACK).

• This connection establishment process introduces additional latency, especially

for short-lived connections or applications that require rapid data transmission.

• In scenarios where connection setup time is critical, such as real-time streaming

or online gaming, TCP's connection-oriented communication model may be a

disadvantage.

3. No Support for Multicast:

• TCP does not support multicast communication, where a single packet is sent to

multiple recipients simultaneously.

• Multicast communication is more efficient than sending individual packets to

each recipient, especially for applications like multimedia streaming or online

gaming that require real-time delivery to multiple users.

• TCP-based applications requiring multicast support may need to use additional

protocols or workarounds to achieve efficient multicast communication.

4. Limited Error Recovery Options:

• While TCP includes mechanisms for detecting and recovering from errors, its

error recovery options are somewhat limited compared to other protocols.

• TCP primarily relies on retransmitting lost packets to recover from errors, which

can introduce additional latency and congestion, especially in high-loss or high-

latency networks.

• In scenarios where rapid error recovery is crucial, such as real-time

communication or mission-critical applications, TCP's error recovery mechanisms

may not be sufficient.

5. Performance Degradation in Congested Networks:

• TCP's congestion control mechanisms can lead to performance degradation in

highly congested networks.

• During periods of network congestion, TCP reduces its transmission rate to

alleviate congestion, which can result in decreased throughput and increased

latency.

• In scenarios where maintaining high throughput and low latency is essential, such

as real-time communication or high-performance computing, TCP's congestion

control mechanisms may be a disadvantage.

8. Applications of TCP :

TCP (Transmission Control Protocol) is widely used in a variety of applications where reliable,

ordered, and error-checked communication is essential. Here are some common applications of

TCP:

1. Web Browsing (HTTP):

• TCP is the underlying protocol for Hypertext Transfer Protocol (HTTP), which is

used for accessing and retrieving web pages on the World Wide Web.

• When you visit a website using a web browser, the browser establishes a TCP

connection to the web server to request and receive HTML, CSS, JavaScript, and

other web content.

2. File Transfer (FTP, SFTP):

• TCP is used in File Transfer Protocol (FTP) and Secure FTP (SFTP) for

transferring files between computers over a network.

• FTP and SFTP clients establish TCP connections to FTP or SFTP servers to

upload, download, or manage files securely.

3. Email (SMTP, IMAP, POP3):

• TCP is utilized in various email protocols, including Simple Mail Transfer

Protocol (SMTP), Internet Message Access Protocol (IMAP), and Post Office

Protocol version 3 (POP3).

• SMTP is used for sending email messages between mail servers, while IMAP and

POP3 are used for retrieving email messages from a mail server to a client device.

4. Remote Login (SSH):

• TCP is employed in Secure Shell (SSH) for securely accessing and managing

remote systems over a network.

• SSH clients establish TCP connections to SSH servers to log in remotely, execute

commands, transfer files, and manage network devices securely.

5. VoIP (Voice over Internet Protocol):

• TCP is used in VoIP applications for transmitting voice and multimedia data

packets over IP networks.

• VoIP clients establish TCP connections to VoIP servers or peers to initiate and

maintain voice calls, video calls, and other real-time communication sessions.

6. Database Access (MySQL, PostgreSQL):

• TCP is employed in database management systems like MySQL, PostgreSQL,

and others for connecting client applications to database servers.

• Database clients establish TCP connections to database servers to perform

queries, updates, inserts, and other database operations securely and reliably.

7. Remote Desktop (RDP, VNC):

• TCP is used in Remote Desktop Protocol (RDP) and Virtual Network Computing

(VNC) for accessing and controlling remote desktops and computers over a

network.

• RDP and VNC clients establish TCP connections to RDP or VNC servers to view

and interact with remote desktops, transfer files, and perform administrative tasks

remotely.

8. Instant Messaging and Chat (XMPP, IRC):

• TCP is utilized in instant messaging protocols like Extensible Messaging and

Presence Protocol (XMPP) and Internet Relay Chat (IRC) for real-time messaging

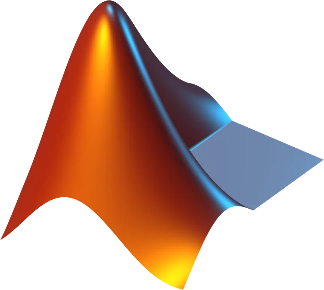
and chat applications.

• Instant messaging clients establish TCP connections to XMPP or IRC servers to

exchange messages, join chat rooms, and participate in group conversation.

**9.Difference between UDP and TCP**

|  |  |  |
| --- | --- | --- |
| **Basis** | **Transmission Control Protocol (TCP)** | **User Datagram Protocol (UDP)** |
| Reliability | TCP is reliable as it guarantees the delivery of data to the destination router. | The delivery of data to the destination cannot be guaranteed in UDP. |
| Error checking mechanism | TCP provides extensive error-checking mechanisms. | UDP has only the basic error-checking mechanism using checksums. |
| Acknowledgment | An acknowledgment segment is present. | No acknowledgment segment. |
| Sequence | Sequencing of data is a feature of Transmission Control  Protocol (TCP). this means that packets arrive in order at the receiver. | There is no sequencing of data in UDP. If the order is required, it has to be managed by the application layer. |
| Speed | TCP is comparatively slower than UDP. | UDP is faster, simpler, and more efficient than TCP. |
| Retransmission | Retransmission of lost packets is possible in TCP, but not in UDP. | There is no retransmission of lost packets in the User Datagram Protocol (UDP). |
| Header Length | TCP has a (20-60) bytes variable length header. | UDP has an 8 bytes fixed-length header. |
| Weight | TCP is heavy-weight. | UDP is lightweight. |
| Handshaking Techniques | Uses handshakes such as SYN, ACK, SYN-ACK | It’s a connectionless protocol i.e. No handshake |
| Broadcasting | TCP doesn’t support Broadcasting. | UDP supports Broadcasting. |



**10.SOFTWARE USED**

**MATLAB:**

**1. Programming Language:**

MATLAB provides an easy-to-understand programming language with syntax that is conducive to

Mathematical and scientific computing.

It supports various programming paradigms, including procedural, functional, and object-oriented

programming.

MATLAB code is interpreted, which allows for rapid prototyping and quick development cycles.

**2. Numerical Computing:**

MATLAB's core strength lies in its numerical computing capabilities.

It provides built-in functions and libraries for performing a wide range of mathematical operations,

Including linear algebra, optimization, statistics, signal processing, and differential equations solving.

MATLAB's matrix-based approach to computation makes it particularly efficient for handling large-scale

numerical problems.

**3. Data Analysis and Visualization:**

MATLAB offers powerful tools for importing, manipulating, analyzing, and visualizing data.

It supports importing data from various file formats, such as spreadsheets, text files, and databases.

MATLAB's plotting functions enable users to create publication-quality plots, charts, graphs, and

animations to visualize data effectively.

**4. Toolboxes:**

MATLAB's functionality can be extended using toolboxes, which are collections of specialized functions

and algorithms for specific application areas.

MathWorks provides a wide range of toolboxes covering domains such as image processing, control

systems, machine learning, signal processing, and more.

These toolboxes allow users to access advanced features and algorithms tailored to their specific needs.

**5. Interactivity:**

MATLAB's interactive environment enables users to execute commands and scripts, visualize data, and

explore algorithms in real-time.

The command window allows for quick experimentation and exploration of MATLAB functions and

features.

MATLAB also supports the creation of graphical user interfaces (GUIs) for building custom interactive

applications.

**6. Deployment:**

MATLAB code can be deployed to various platforms and formats for sharing or integration into larger

systems.

MATLAB Compiler allows users to convert MATLAB code into standalone executables or shared

libraries for deployment on systems without MATLAB installed.

MATLAB code can also be deployed to web applications, hardware devices, and cloud platforms.

**7. Integration:**

MATLAB can be integrated with other programming languages and tools, enabling seamless

interoperability.

MATLAB supports calling external functions and libraries written in languages such as C/C++, Java,

Python, and .NET.

MATLAB also provides APIs and interfaces for interacting with external software, databases,

and hardware devices.

**8. Community and Support:**

MATLAB has a large and active user community, with resources such as documentation, forums, and

online tutorials available for learning and troubleshooting.

MathWorks offers comprehensive technical support, training courses, and consulting services for

MATLAB users.

MATLAB Central, MathWorks' online community platform, provides a wealth of user-contributed code,

examples, and resources for MATLAB users.

Overall, MATLAB is a versatile and powerful platform that offers a wide range of capabilities for

numerical computing, data analysis, visualization, and application development. Its ease of use,

extensive functionality,

and strong community support make it a popular choice for researchers, engineers, educators, and

students worldwide.

11. MATLAB code for UDP :

i. UDP Server:

try

% Create a UDP object with Ip Address ‘0.0.0.0’, port ‘12345’, local port ‘54321' and

terminator to indicate the end of a message (''):

udpServer = udp('0.0.0.0', 12345, 'LocalPort', 54321, 'Terminator', '');

% Open the UDP connection

fopen(udpServer);

% Display message indicating the server is waiting for messages

disp('UDP Server is waiting for messages...');

% Infinite loop to continuously check for incoming data

while true

% Check if there's data available to read

if udpServer.BytesAvailable > 0

% Read the available data from the UDP buffer

dataReceived = fread(udpServer, udpServer.BytesAvailable);

% Process the received data (add your processing logic here)

disp(['Received data: ' char(dataReceived')]);

% No acknowledgment is sent back to the client in this version

end

% Pause for 1 second before the next iteration (adjust as needed)

pause(1);

end

catch serverException

% Handle any errors that may occur during execution

disp(['Error in UDP server: ' serverException.message]);

end

% Close the UDP server (not reached in this example)

fclose(udpServer);

delete(udpServer);

ii. UDP Client :

try

% Create a UDP object with Ip Address ‘192.168.7.103’, port ‘54321’, local port ‘12345' and

terminator to indicate the end of a message (''):

udpClient = udp(‘192.168.7.103’, 54321, 'LocalPort', 12345, 'Terminator', '');

% Open the UDP connection

fopen(udpClient);

% Define the data to be sent

dataToSend = 'Liverpool is the best club';

% Send the data to the server

fwrite(udpClient, dataToSend);

% Pause for 1 second to allow time for the server to process

pause(1);

% Check if there's acknowledgment data available to read

if udpClient.BytesAvailable > 0

% Read the acknowledgment data from the UDP buffer

acknowledgmentReceived = fread(udpClient, udpClient.BytesAvailable);

% Display acknowledgment received from the server

disp(['Acknowledgment from server: ' char(acknowledgmentReceived')]);

else

% If no acknowledgment data is available, display a message

disp('No acknowledgment available.');

end

% Close the UDP connection

fclose(udpClient);

delete(udpClient);

catch clientException

% Handle any errors that may occur during execution

disp(['Error in UDP client: ' clientException.message]);

end



iii. Output :

12. MATLAB Code for TCP :

i. TCP Server :

% Create a TCP object with Ip Address ‘0.0.0.0’, port ‘12345’, and network role as Server.

tcpServer = tcpip('0.0.0.0', 12345, 'NetworkRole', 'server');

fopen(tcpServer);

% Loop until data is available

while tcpServer.BytesAvailable == 0

pause(0.1); % Adjust the pause duration based on your requirements

end

% Receive data

dataReceived = fread(tcpServer, tcpServer.BytesAvailable);

disp(['Received data: ' char(dataReceived')]);

% Process the data (add your processing logic here)

% Send acknowledgment

acknowledgment = 'Data received successfully!';

fwrite(tcpServer, acknowledgment);

% Close the server

fclose(tcpServer);

ii. TCP Client :

% Create a TCP object with Ip Address ‘0.0.0.0’, port ‘12345’, and network role as Client

tcpClient = tcpip(‘192.168.7.103’, 12345, 'NetworkRole', 'client');

fopen(tcpClient);

% Send data

dataToSend = 'Hello, Server!';

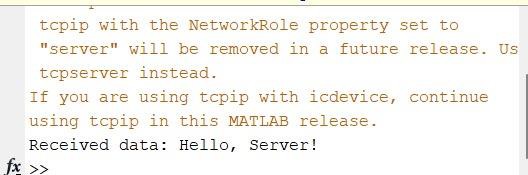
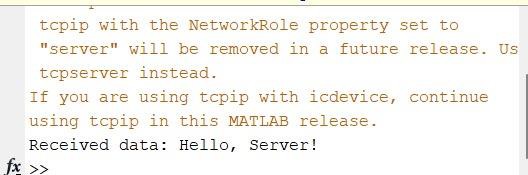
fwrite(tcpClient, dataToSend);

% Loop until acknowledgment is available

while tcpClient.BytesAvailable == 0

pause(0.1); % Adjust the pause duration based on your requirements

end

% Receive acknowledgment

acknowledgmentReceived = fread(tcpClient, tcpClient.BytesAvailable);

disp(['Acknowledgment from server: ' char(acknowledgmentReceived')]);

% Close the client

fclose(tcpClient);

iii. OUTPUT :

13. CONCLUSION :

With the help of this PBL we have learnt about,

i. User Datagram Protocol (UDP) :

The User Datagram Protocol (UDP) is a connectionless transport layer protocol used in

computer communication networks. It is part of the Internet Protocol (IP) suite and is often

employed when low overhead and reduced latency are more critical than guaranteed delivery and

error recovery.

ii. Transmission Control Protocol (TCP) :

The Transmission Control Protocol (TCP) is a connection-oriented transport layer protocol

in computer communication networks. It plays a crucial role in providing reliable and ordered

delivery of data between communicating devices. TCP is part of the Internet Protocol (IP) suite

and is widely used for various applications, including web browsing, file transfer, and email.

iii. We have Implemented UDP and TCP Protocols in MATLAB

iv. Therefore we have satisfied the course outcome 4 ( CO4 ) :

Articulate the protocols and functions associated with the transport layer services.