ASSIGNMENT-3

GROUP-15 MEMBERS :
Adloori Chandana 220101008
Aetukuri Sri Durga 220101009
Alampally Khushi 220101010
Ramineni Hardhika 20123052

Q2) Real-Time Weather Monitoring System

This report details the development of a real-time weather monitoring system consisting of a server and multiple weather stations distributed across a city. The system is designed to handle high data throughput, adapt to varying network conditions, and efficiently manage data transmission.
Following are done on the server side of the system.

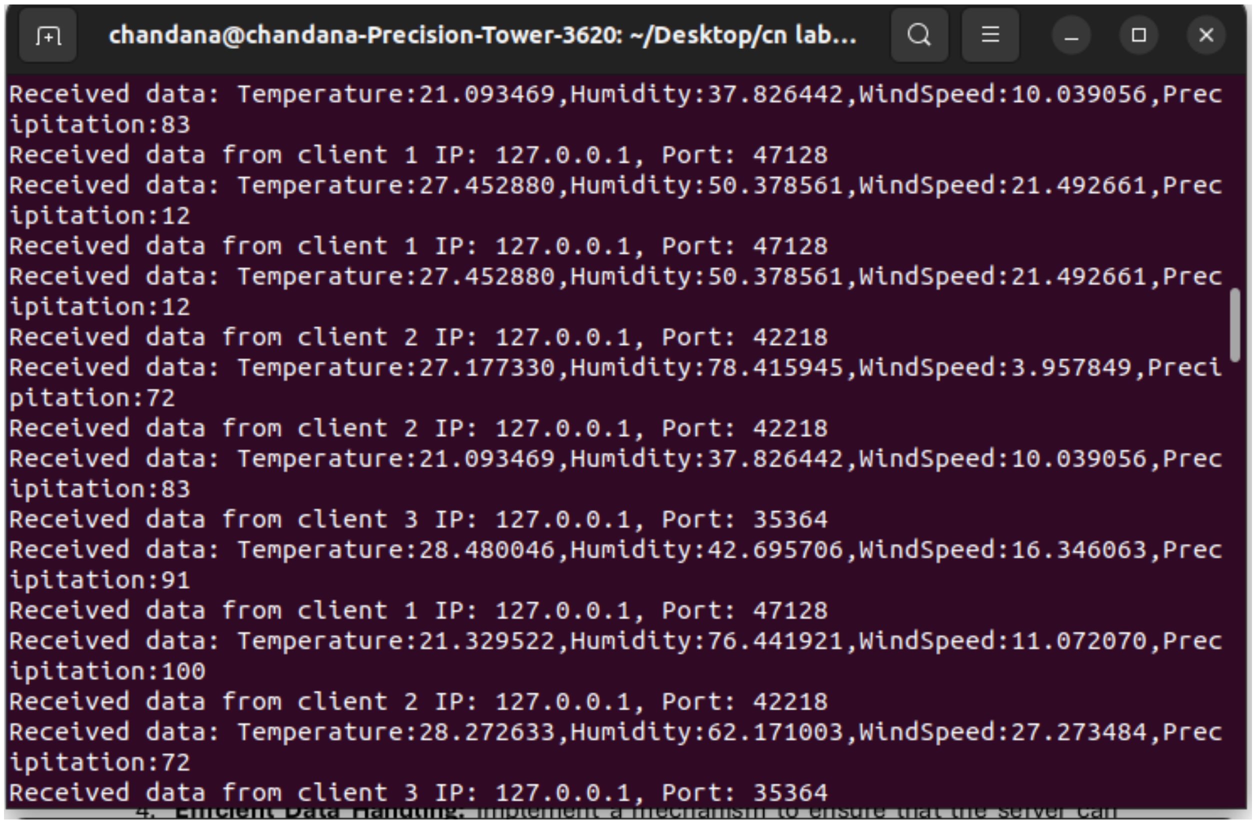
- The server code handles multiple concurrent connections from weather stations using a multi-threaded approach.
- It creates a socket, binds it to a port, and listens for incoming client connections.
- Each client is managed by a separate thread, which handles data reception, decompression, and acknowledgment based on a random probability.
- The server prints client information and the received data, ensuring thread safety with mutexes when accessing shared resources.
- It uses the zlib library for data decompression and incorporates basic error handling for socket operations and thread management.
- The server is designed to handle high client throughput efficiently.

The client code establishes a TCP connection to a server and simulates a weather station sending data with packet loss and congestion control.

- It compresses weather data before sending.
- uses a 50% packet loss probability to decide whether to actually send the packet or buffer it.

- The client implements a simplified version of **TCP Reno's congestion control algorithm**, adjusting the congestion window size (CWND) and slow-start threshold (SSTHRESH) based on packet acknowledgments and retransmissions.
- It handles acknowledgments with a timeout and retransmits packets if needed. The code also registers signal handlers for termination and prints transmitted packets statistics, including **packet loss, total packets sent, retransmissions and acknowledgments received.**

The following depicts that the server is capable of connecting to multiple clients and is collecting data from each.

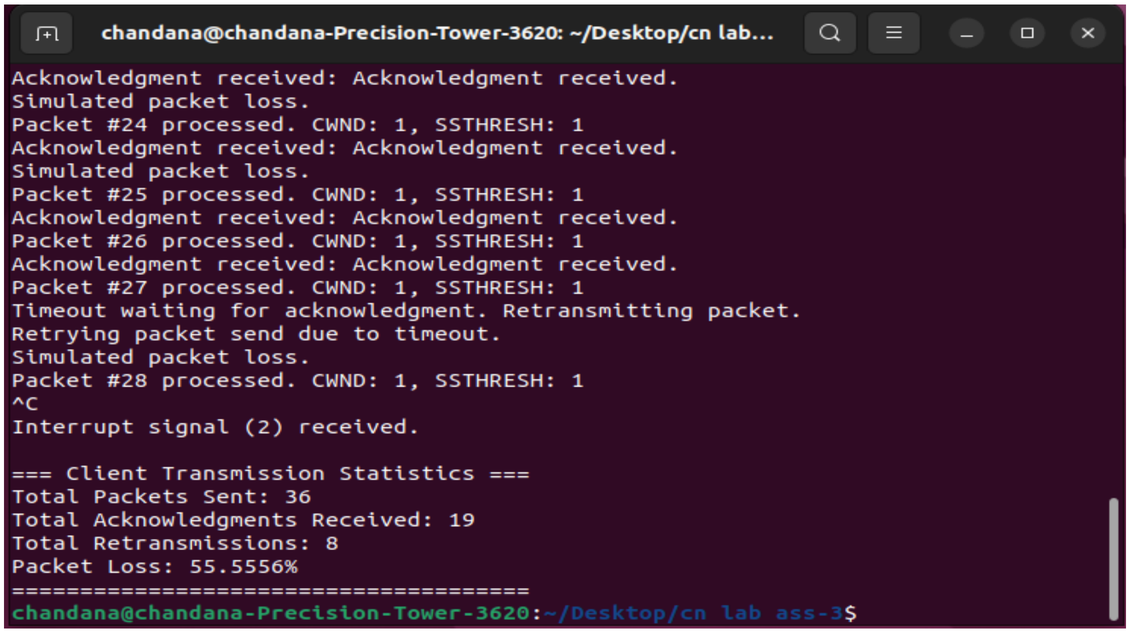


```
Received data: Temperature:21.093469,Humidity:37.826442,WindSpeed:10.039056,Prec
ipitation:83
Received data from client 1 IP: 127.0.0.1, Port: 47128
Received data: Temperature:27.452880,Humidity:50.378561,WindSpeed:21.492661,Prec
ipitation:12
Received data from client 1 IP: 127.0.0.1, Port: 47128
Received data: Temperature:27.452880,Humidity:50.378561,WindSpeed:21.492661,Prec
ipitation:12
Received data from client 2 IP: 127.0.0.1, Port: 42218
Received data: Temperature:27.177330,Humidity:78.415945,WindSpeed:3.957849,Preci
pitation:72
Received data from client 2 IP: 127.0.0.1, Port: 42218
Received data: Temperature:21.093469,Humidity:37.826442,WindSpeed:10.039056,Prec
ipitation:83
Received data from client 3 IP: 127.0.0.1, Port: 35364
Received data: Temperature:28.480046,Humidity:42.695706,WindSpeed:16.346063,Prec
ipitation:91
Received data from client 1 IP: 127.0.0.1, Port: 47128
Received data: Temperature:21.329522,Humidity:76.441921,WindSpeed:11.072070,Prec
ipitation:100
Received data from client 2 IP: 127.0.0.1, Port: 42218
Received data: Temperature:28.272633,Humidity:62.171003,WindSpeed:27.273484,Prec
ipitation:72
Received data from client 3 IP: 127.0.0.1, Port: 35364
```

4. Efficient Data Handling: Implement a mechanism to ensure that the server can

Packet stats of the three clients are as below

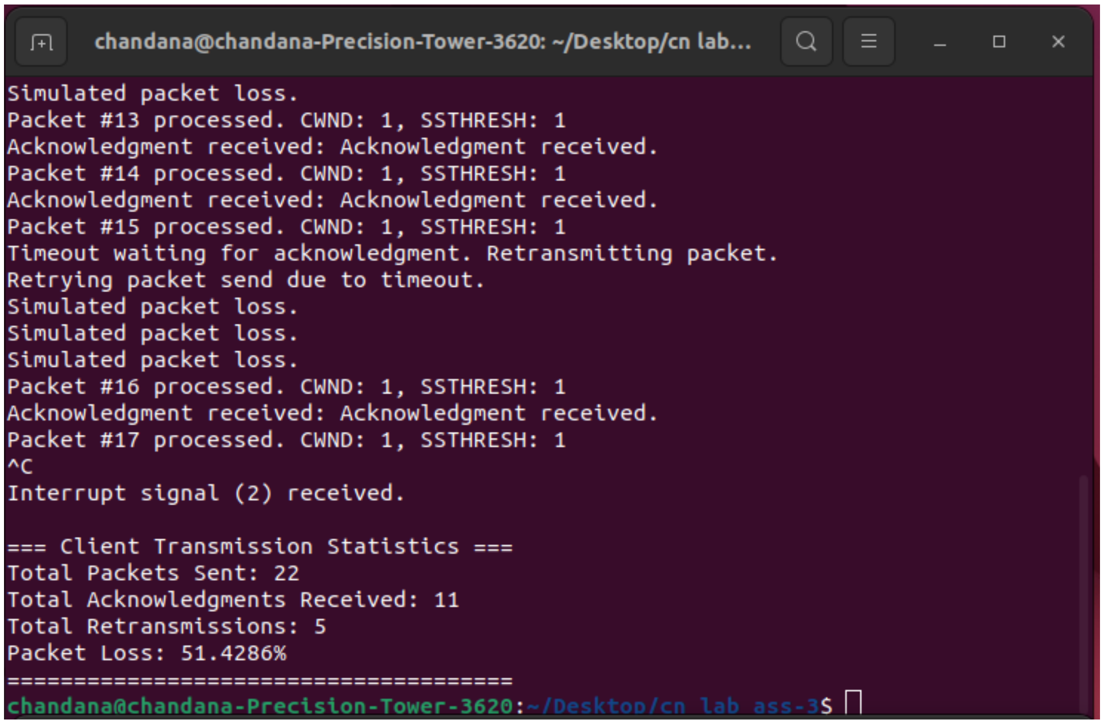
1.



```
Acknowledgment received: Acknowledgment received.
Simulated packet loss.
Packet #24 processed. CWND: 1, SSTHRESH: 1
Acknowledgment received: Acknowledgment received.
Simulated packet loss.
Packet #25 processed. CWND: 1, SSTHRESH: 1
Acknowledgment received: Acknowledgment received.
Packet #26 processed. CWND: 1, SSTHRESH: 1
Acknowledgment received: Acknowledgment received.
Packet #27 processed. CWND: 1, SSTHRESH: 1
Timeout waiting for acknowledgment. Retransmitting packet.
Retrying packet send due to timeout.
Simulated packet loss.
Packet #28 processed. CWND: 1, SSTHRESH: 1
^C
Interrupt signal (2) received.

=== Client Transmission Statistics ===
Total Packets Sent: 36
Total Acknowledgments Received: 19
Total Retransmissions: 8
Packet Loss: 55.5556%
======================================
chandana@chandana-Precision-Tower-3620:~/Desktop/cn lab ass-3$
```

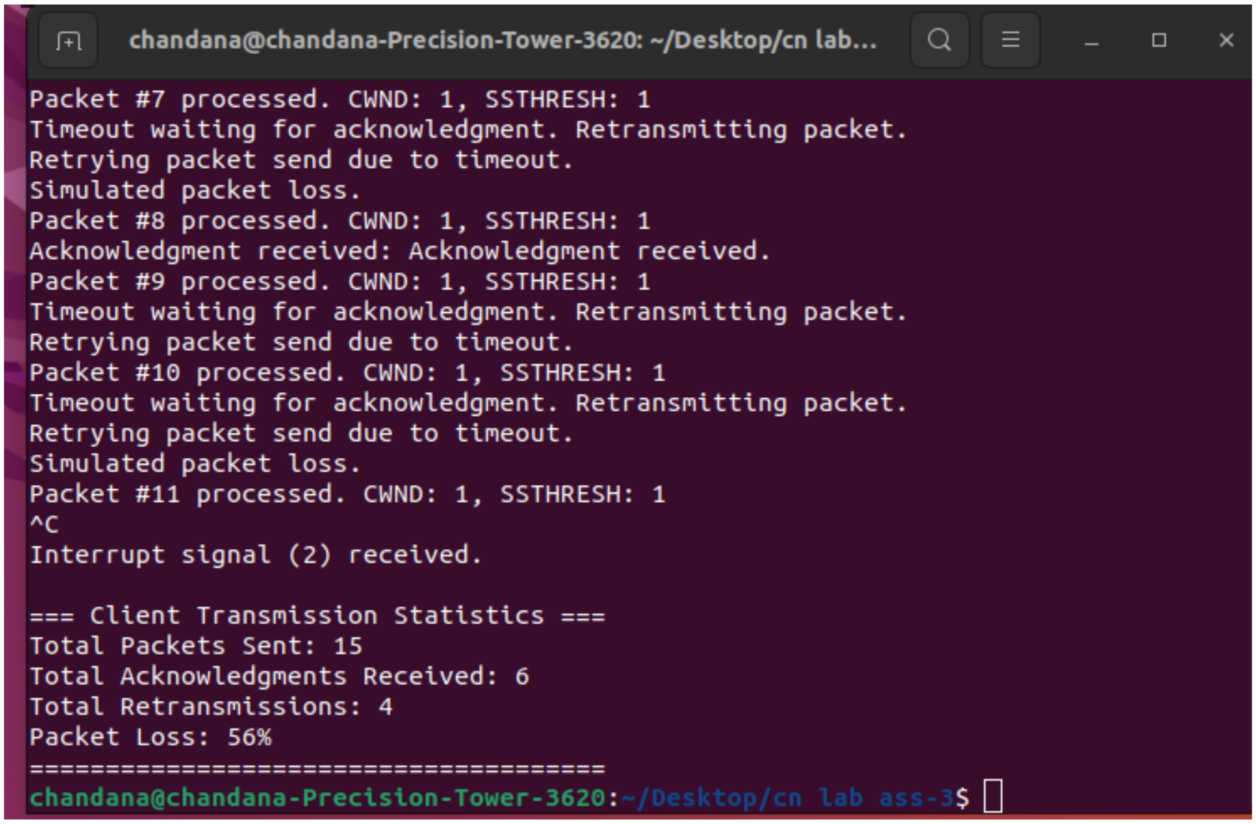
2.



```
Simulated packet loss.
Packet #13 processed. CWND: 1, SSTHRESH: 1
Acknowledgment received: Acknowledgment received.
Packet #14 processed. CWND: 1, SSTHRESH: 1
Acknowledgment received: Acknowledgment received.
Packet #15 processed. CWND: 1, SSTHRESH: 1
Timeout waiting for acknowledgment. Retransmitting packet.
Retrying packet send due to timeout.
Simulated packet loss.
Simulated packet loss.
Simulated packet loss.
Packet #16 processed. CWND: 1, SSTHRESH: 1
Acknowledgment received: Acknowledgment received.
Packet #17 processed. CWND: 1, SSTHRESH: 1
^C
Interrupt signal (2) received.

=== Client Transmission Statistics ===
Total Packets Sent: 22
Total Acknowledgments Received: 11
Total Retransmissions: 5
Packet Loss: 51.4286%
======================================
chandana@chandana-Precision-Tower-3620:~/Desktop/cn lab ass-3$
```

3.



```
Packet #7 processed. CWND: 1, SSTHRESH: 1
Timeout waiting for acknowledgment. Retransmitting packet.
Retrying packet send due to timeout.
Simulated packet loss.
Packet #8 processed. CWND: 1, SSTHRESH: 1
Acknowledgment received: Acknowledgment received.
Packet #9 processed. CWND: 1, SSTHRESH: 1
Timeout waiting for acknowledgment. Retransmitting packet.
Retrying packet send due to timeout.
Packet #10 processed. CWND: 1, SSTHRESH: 1
Timeout waiting for acknowledgment. Retransmitting packet.
Retrying packet send due to timeout.
Simulated packet loss.
Packet #11 processed. CWND: 1, SSTHRESH: 1
^C
Interrupt signal (2) received.

=== Client Transmission Statistics ===
Total Packets Sent: 15
Total Acknowledgments Received: 6
Total Retransmissions: 4
Packet Loss: 56%
======================================
chandana@chandana-Precision-Tower-3620:~/Desktop/cn lab ass-3$ 
```

Q1)**Server Code Description**

1. Server Initialization

The server is responsible for handling three communication modes: control commands, telemetry data, and file transfers. It listens on different ports for each mode, ensuring that the communication channels remain distinct.

UDP Socket for Control Commands:
The server creates a UDP socket to send low-latency control commands such as 'move', 'speed', and 'halt' to the drones.

The server binds to a specific port (e.g., 9000) for sending these commands to the drones.

TCP Socket for Telemetry Data:
The server opens a TCP socket on another port (e.g., 9001) to receive telemetry data from drones.

Telemetry data includes information like drone ID, GPS coordinates, battery status, and system health.

TCP Socket for File Transfer:
Another TCP socket listens on a third port (e.g., 9001) for file transfers. Files can include drone logs, images captured by drones, or updates to the drone's software.

The file transfer process involves chunking large files and ensuring that each chunk is transmitted securely using XOR encryption.

2. Control Command Handling

The server periodically sends encrypted control commands to the drones using the UDP socket.

Each control command consists of:

Drone ID: Identifying which drone to control.

Command Type: 'move', 'speed', or 'halt'.

Parameters: Direction and speed for 'move', or the value for 'speed'.

Command Encryption:The control command is XOR-encrypted using a shared key before sending. The encryption helps in securing the command against unauthorized access during transmission.

3. Receiving Telemetry Data

The server waits for connections from drones to send telemetry data. Once a connection is established, the server receives the telemetry packets from the drone.

The telemetry packet contains:

Drone ID: Identifying which drone is sending the data.

Timestamp: When the data was recorded.

GPS Coordinates: Latitude, longitude, and altitude.

Battery Status: Remaining battery percentage.

System Health Metrics: Processor usage, memory usage, etc.

- Each telemetry packet is XOR-encrypted by the drone before sending. The server decrypts the packet using the same XOR key after receiving it to retrieve the telemetry data.

4. Handling File Transfers

- The server initiates or responds to file transfer requests using the TCP socket designated for file transfers.
- Files are divided into chunks before sending to ensure that large files can be transmitted over the network without issues.
- The server requests the file, and once the drone starts sending it, the server receives each encrypted chunk and decrypts it on the fly.

Client (Drone) Code Description

1. Client Initialization

The drone (client) has the responsibility to receive control commands from the server, send telemetry data, and manage file transfers. Like the server, the client also uses distinct sockets for each communication mode.

UDP Socket for Control Commands:

The drone listens for control commands on a specific UDP port.Upon receiving a control command, the drone decrypts the command using XOR encryption and then executes the appropriate action.

TCP Socket for Telemetry Data:

The drone establishes a TCP connection with the server to send telemetry data at regular intervals.Telemetry data is collected, encrypted using XOR encryption, and then sent to the server via the established TCP connection.

TCP Socket for File Transfer:
The drone is prepared to send or receive files via another TCP socket when requested by the server.

Files are split into chunks and encrypted using XOR before transmission.

2. Receiving Control Commands

- The client listens on a specific UDP port for control commands from the server.
- Upon receiving an encrypted control command, the drone decrypts the command using the XOR key and parses the command.

Based on the command type, the drone performs actions such as:

- **Move**: Adjust the drone’s direction and speed based on the parameters.
- **Speed**: Change the current speed of the drone.
- **Halt**: Stop the drone immediately.

3. Sending Telemetry Data

The client gathers telemetry data at regular intervals. This includes:

Drone ID: Unique identifier for the drone.

Timestamp: Time of data collection.

Location: Current GPS coordinates (latitude, longitude, and altitude).

Telemetry Encryption:

The telemetry data is encrypted using XOR encryption to secure it during transmission.

The encrypted data is then sent via TCP to the server.

4. File Transfer

- When the server initiates a file transfer request, the drone reads the file (e.g., a log file) from its storage.
- The file is divided into smaller chunks to facilitate smooth transmission over the network.
- Each chunk is encrypted using XOR encryption and then transmitted to the server.

Encryption with XOR Cipher

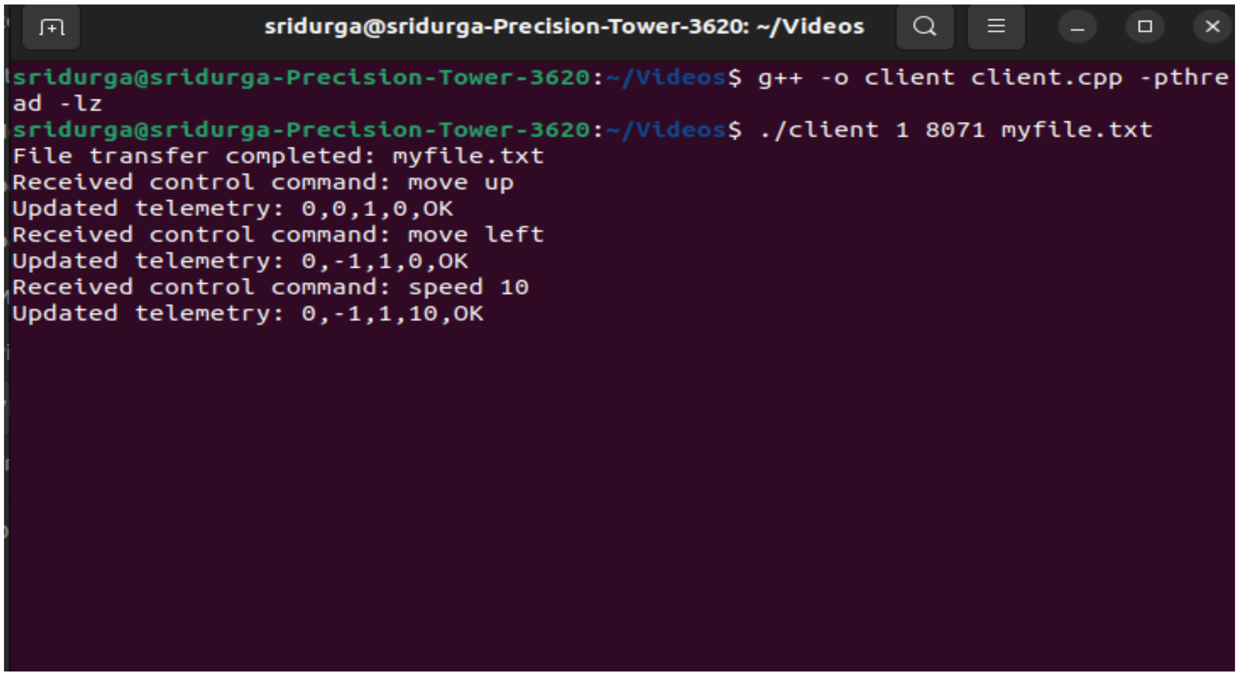
Both the server and drone client use a simple XOR cipher for encryption and decryption:

- **Encryption**: Each byte of the message is XORed with a secret key to produce an encrypted byte.

- **Decryption**: The same XOR operation is applied again using the secret key to retrieve the original data.

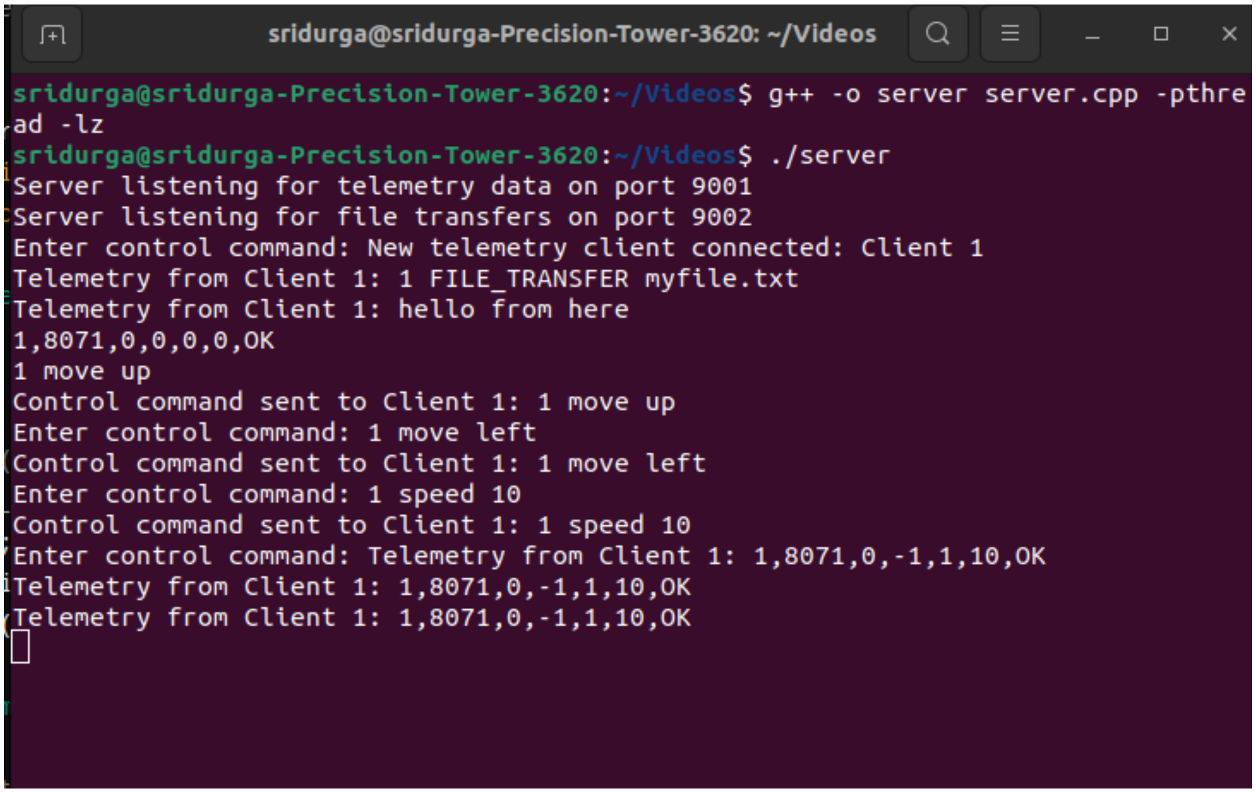
The XOR encryption ensures that messages, whether control commands, telemetry data, or files, remain secure during transmission.

1)Text file transfer



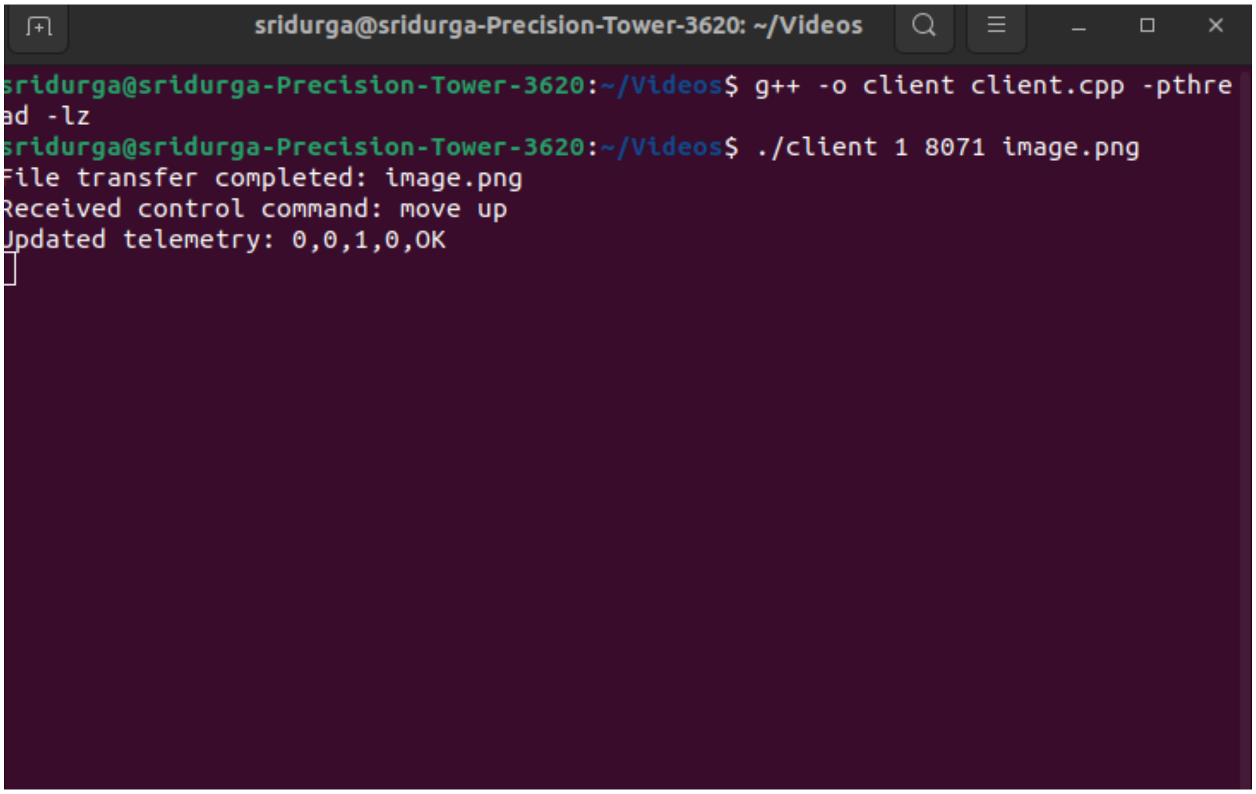
sridurga@sridurga-Precision-Tower-3620: ~/Videos

```
sridurga@sridurga-Precision-Tower-3620:~/Videos$ g++ -o client client.cpp -pthre
ad -lz
sridurga@sridurga-Precision-Tower-3620:~/Videos$ ./client 1 8071 myfile.txt
File transfer completed: myfile.txt
Received control command: move up
Updated telemetry: 0,0,1,0,OK
Received control command: move left
Updated telemetry: 0,-1,1,0,OK
Received control command: speed 10
Updated telemetry: 0,-1,1,10,OK
```

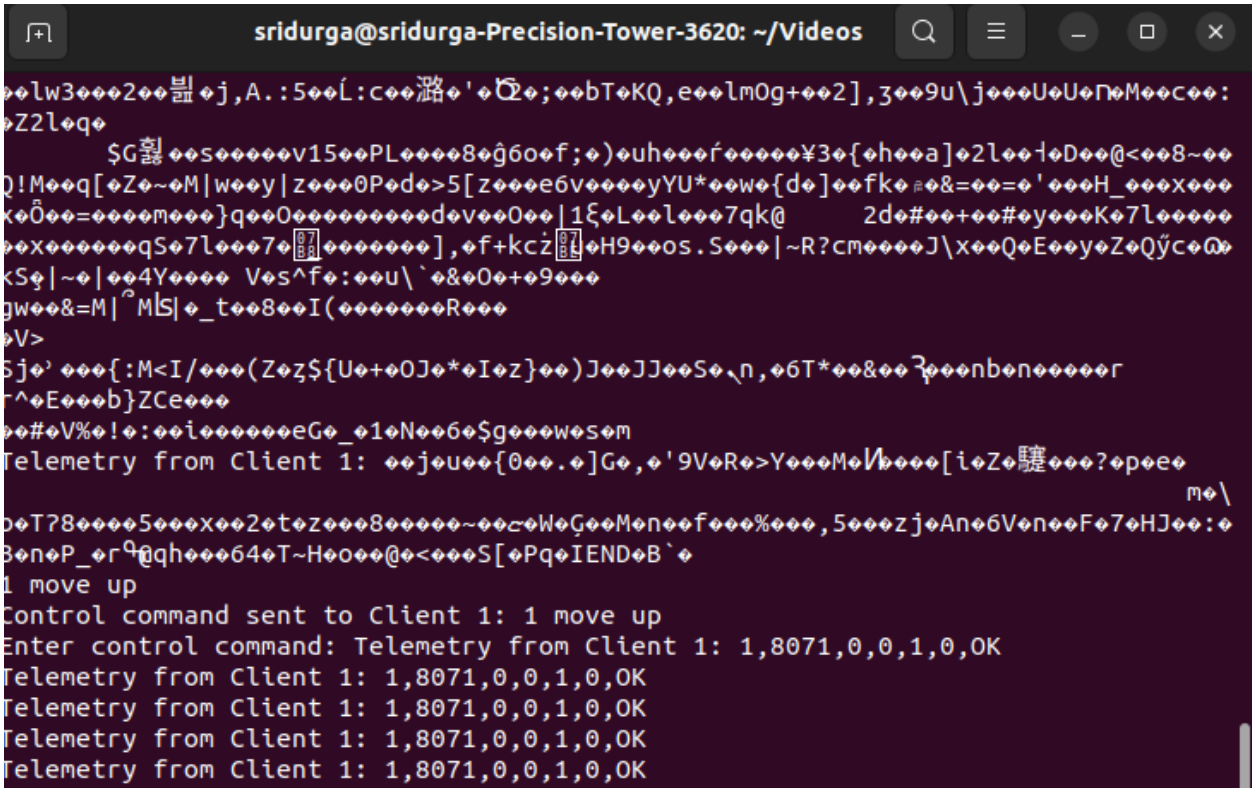


```
sridurga@sridurga-Precision-Tower-3620: ~/Videos
sridurga@sridurga-Precision-Tower-3620:~/Videos$ g++ -o server server.cpp -pthre
ad -lz
sridurga@sridurga-Precision-Tower-3620:~/Videos$ ./server
Server listening for telemetry data on port 9001
Server listening for file transfers on port 9002
Enter control command: New telemetry client connected: Client 1
Telemetry from Client 1: 1 FILE_TRANSFER myfile.txt
Telemetry from Client 1: hello from here
1,8071,0,0,0,0,OK
1 move up
Control command sent to Client 1: 1 move up
Enter control command: 1 move left
Control command sent to Client 1: 1 move left
Enter control command: 1 speed 10
Control command sent to Client 1: 1 speed 10
Enter control command: Telemetry from Client 1: 1,8071,0,-1,1,10,OK
Telemetry from Client 1: 1,8071,0,-1,1,10,OK
Telemetry from Client 1: 1,8071,0,-1,1,10,OK
```

2)Image file transfer:

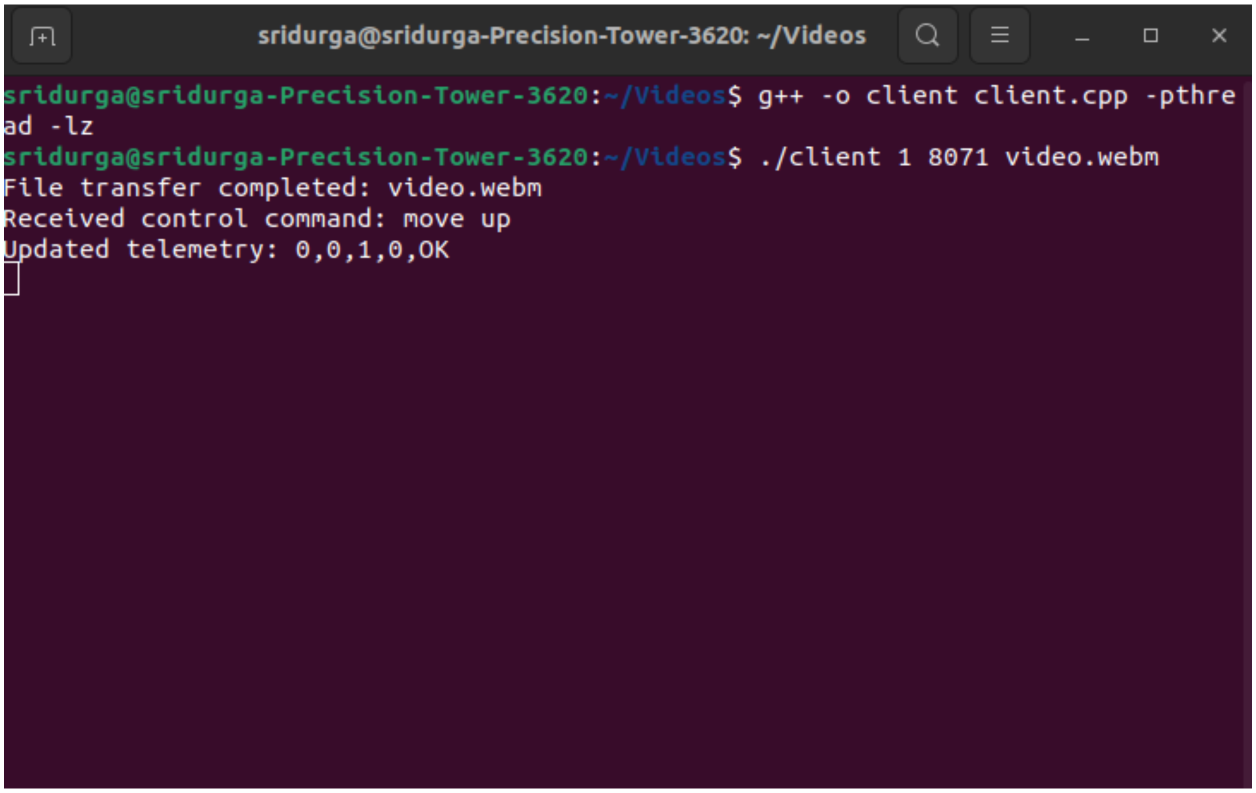


```
sridurga@sridurga-Precision-Tower-3620: ~/Videos
sridurga@sridurga-Precision-Tower-3620:~/Videos$ g++ -o client client.cpp -pthre
ad -lz
sridurga@sridurga-Precision-Tower-3620:~/Videos$ ./client 1 8071 image.png
File transfer completed: image.png
Received control command: move up
Updated telemetry: 0,0,1,0,OK
```

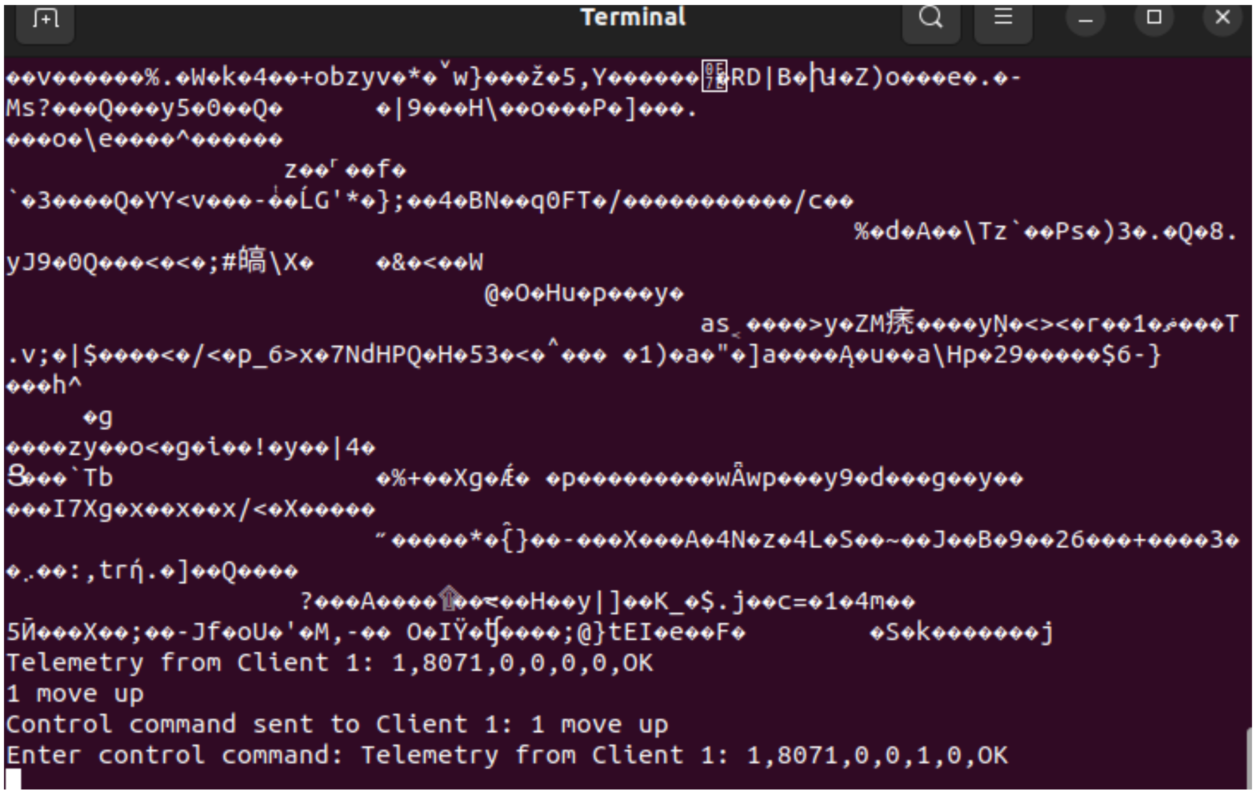


```
sridurga@sridurga-Precision-Tower-3620: ~/Videos
��lw3���2��븳�j,A.:5��Ĺ:c��潞�'�Ö2�;��bT�KQ,e��lmOg+��2],ʒ��9u\j���U�U�ᴨ�M��c��:
�Z2l�q�
       $G쿓��s������v15��PL����8�ĝ6o�f;�)�uh���ŕ������¥3�{�h��a]�2l��ᆟ�D��@<��8~��
Q!M��q[�Z�~�M|w��y|z���0P�d�>5[z���e6v�����yYU*��w�{d�]��fk��&=��=�'���H_���x���
x�Ő��=����m���}q��O����������d�v��O��|1ξ�L��l���7qk@      2d�#��+��#�y���K�7l�����
��x�������qS�7l���7�������],�f+kcż�H9���os.S���|~R?cm�����J\x��Q�E��y�Z�Qў c�Ꙍ�
kSę|~�|��4Y���� V�s^f�:��u\`�&�O�+�9���
gw��&=M|̂MЬ|�_t��8��I(�������R���
�V>
Sj�׳ ���{:M<I/���(Z�ʐ${U�+�OJ�*�I�z}��)J��JJ��S�ˎn,�6T*��&��ʕ���nb�n������r
r^�E���b}ZCe���
��#�V%�!�:��i�������eG�_�1�N��6�$g����w�s�m
Telemetry from Client 1: ��j�u��{0��.�]G�,�'9V�R�>Y���M�Ӣ���[i�Z�驥���?�p�e�
                                                                             m�\
b�T?8�����5���x��2�t�z���8������~��ᓄ�W�Ģ��M�n��f���%���,5���zj�An�6V�n��F�7�HJ��:�
B�n�P_�r٩@qh���64�T~H�o��@�<���S[�Pq�IEND�B`�
1 move up
Control command sent to Client 1: 1 move up
Enter control command: Telemetry from Client 1: 1,8071,0,0,1,0,OK
Telemetry from Client 1: 1,8071,0,0,1,0,OK
Telemetry from Client 1: 1,8071,0,0,1,0,OK
Telemetry from Client 1: 1,8071,0,0,1,0,OK
Telemetry from Client 1: 1,8071,0,0,1,0,OK
```

3)Video file transfer:



```
sridurga@sridurga-Precision-Tower-3620:~/Videos$ g++ -o client client.cpp -pthre
ad -lz
sridurga@sridurga-Precision-Tower-3620:~/Videos$ ./client 1 8071 video.webm
File transfer completed: video.webm
Received control command: move up
Updated telemetry: 0,0,1,0,OK
```



```
��v�������%.�W�k�4��+obzyv�*�ˇw}���ž�5,Y�������RD|B�|ㅂ�Z)o����e�.�-
Ms?���Q����y5�0��Q�      �|9���H\��o����P�]���.
����o�\e�����^������
                  z��ᒥ��f�
`�3����Q�YY<v���-��ĹG'*�};��4�BN��q0FT�/�����������/c��
                                                       %�d�A��\Tz`��Ps�)3�.�Q�8.
yJ9�0Q���<�<�;#暠\X�    �&�<��W
                              @�O�Hu�p���y�
                                            as ����>y�ZM痔����yŅ�<><�r��1�؞���T
.v;�|$����<�/<�p_6>x�7NdHPQ�H�53�<�ˆ��� �1)�a�"�]a����Ą�u��a\Hp�29�����$6-}
���h^
     �g
����zy���o<�g�i��!�y��|4�
Ȣ���`Tb                �%+��Xg�Ǽ� �p���������wȂwp����y9�d���g��y��
���I7Xg�x��x��x/<�X�����
                        ˝�����*�{̂}��-���X���A�4N�z�4L�S��~��J��B�9��26���+����3�
�.��:,trή.�]��Q����
                   ?���A����⇧��<��H��y|]��K_�$.j��c=�1�4m��
5Й���X��;��-Jf�oU�'�M,-�� O�IŸ�ʧ����;@}tEI�e��F�       �S�k�������j
Telemetry from Client 1: 1,8071,0,0,0,0,OK
1 move up
Control command sent to Client 1: 1 move up
Enter control command: Telemetry from Client 1: 1,8071,0,0,1,0,OK
```

For multiple clients we tried this code please refer to the multiple clients folder.