

Solutions To Overcome Network Unreliability

SUBMITTED BY:

HASSAN AHMED 356963

MUHAMMAD AZEEM 346793

General Description:

This project involves the development of sender and receiver programs that facilitate the transfer of video files using the UDP protocol in Linux/GNU C. The sender program reads video files, divides them into data chunks, and sends these chunks as UDP segments. The receiver, on the other hand, is responsible for accepting, reordering, and saving these segments to reconstruct the original file. The system employs enhanced mechanisms over standard UDP to achieve greater reliability.

Client Module:

The sender's primary function is to transmit a video file via UDP sockets. It segments the file into 500-byte packets, incorporating retransmission for packets not acknowledged by the receiver. The transmission employs a sliding window protocol with a window size of five UDP segments, interspaced by a 0.03-second delay. The sender also listens on a specific port for acknowledgments and signals from the receiver, using a single UDP socket for both sending and receiving data. The process concludes with the sender shutting down after successful file transfer.

Server:

The receiver binds to a specified UDP port and receives the file in 500-byte segments. Each packet is sequentially numbered to facilitate reordering and accurate reconstruction of the file at the receiver's end. The receiver operates within the same window size and delay constraints as the sender. After the complete reception of the file, the receiver saves it under a different filename and then terminates.

Design Considerations

Addressing UDP's Inherent Unreliability:

UDP, known for its speed, lacks in-built mechanisms to ensure data integrity and order. This is typically not an issue for applications like streaming or VoIP, where speed is prioritized over accuracy. However, for file transfers, these limitations are significant.

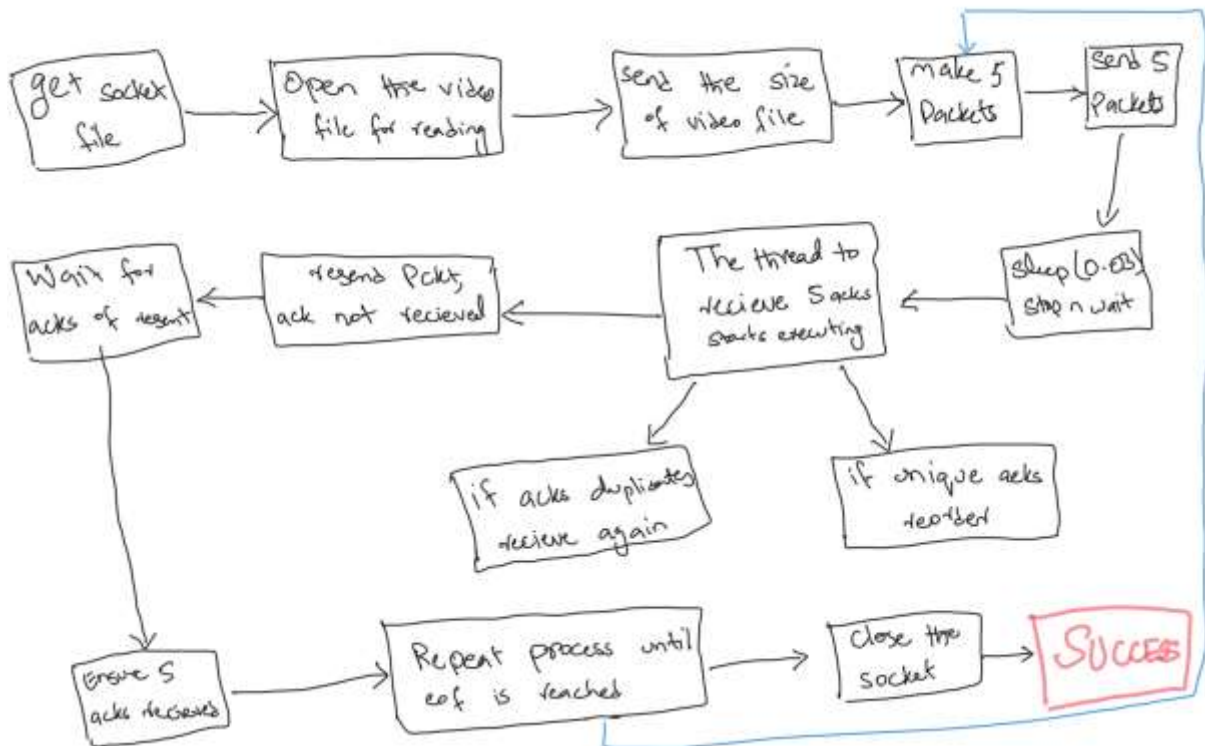
Enhancing UDP Reliability:

To adapt UDP for reliable video file transfer, several strategies are employed:

1. **Sequence Numbering:** Packets are sequentially numbered to maintain order.
2. **Selective Retransmission:** Packets without acknowledgments are resent.
3. **Windowed Transmission:** A window of five segments is sent at a time, followed by a pause for acknowledgments and potential retransmission.
4. **Reordering at Receiver:** The receiver uses the sequence numbers to reorder the packets correctly.

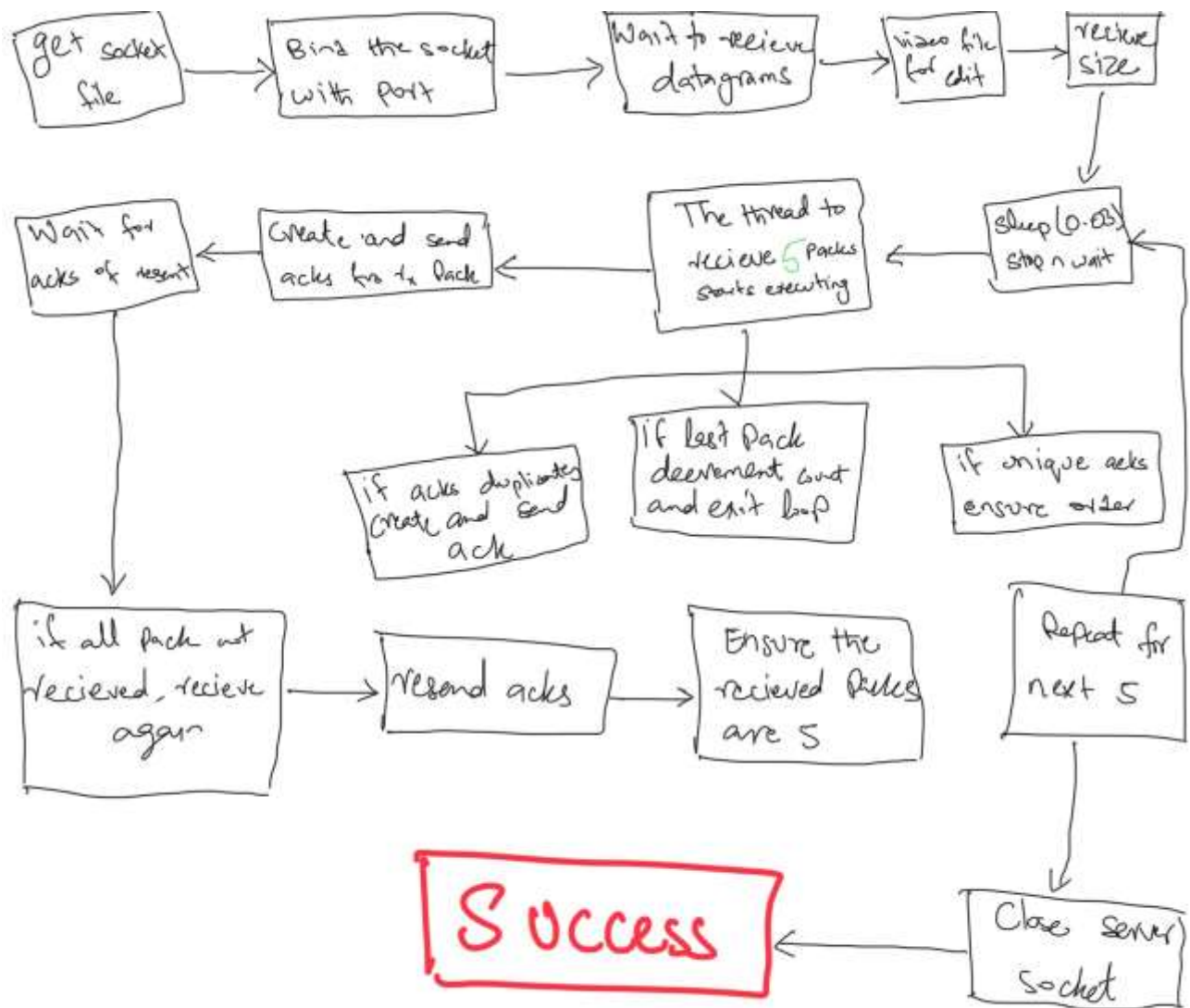
Client/Sender Implementation (udp_client.c):

The sender initiates a UDP socket and reads the video file, first sending the file size to the receiver. It then enters a loop where it segments the file, sends packets, waits for acknowledgments, and handles retransmissions. This loop continues until the entire file is successfully transferred.



Server/Receiver Implementation (udp_server.c):

The receiver sets up a UDP socket, binds it, and then awaits data from the sender. It first receives the file size, then proceeds to receive packets, send acknowledgments, and write the data to an output file. The packets are written in the correct sequence, ensuring the integrity of the video file.



Output

Client:

Connection Initialized:

```

(sh4d0w@sh4d0w)-[~/Desktop/reliable_udp]
$ gcc udp_client.c -lpthread -o client

(sh4d0w@sh4d0w)-[~/Desktop/reliable_udp]
$ ./client 127.0.0.1
Size of Video File: 1175017 bytes
Sending packet 0
Sending packet 1
Sending packet 2
Sending packet 3
Sending packet 4
Sending missing packet: 0
Sending missing packet: 1
Sending missing packet: 2
Sending missing packet: 3

```

Process Completed

```

File  Actions  Edit  View  Help
Sending packet 3
Sending packet 4
Ack Received: 0
Ack Received: 1
Ack Received: 2
Ack Received: 3
Ack Received: 4
Sending packet 0
Sending packet 1
Sending packet 2
Sending packet 3
Sending packet 4
Ack Received: 0
Ack Received: 1
Ack Received: 2
Ack Received: 3
Ack Received: 4
End of file reached.
Sending packet 0
Sending packet 1
Ack Received: 0
Ack Received: 1

File transfer completed successfully!

(sh4d0w@sh4d0w)-[~/Desktop/reliable_udp]
$

```

Server:

Connection Initialized:

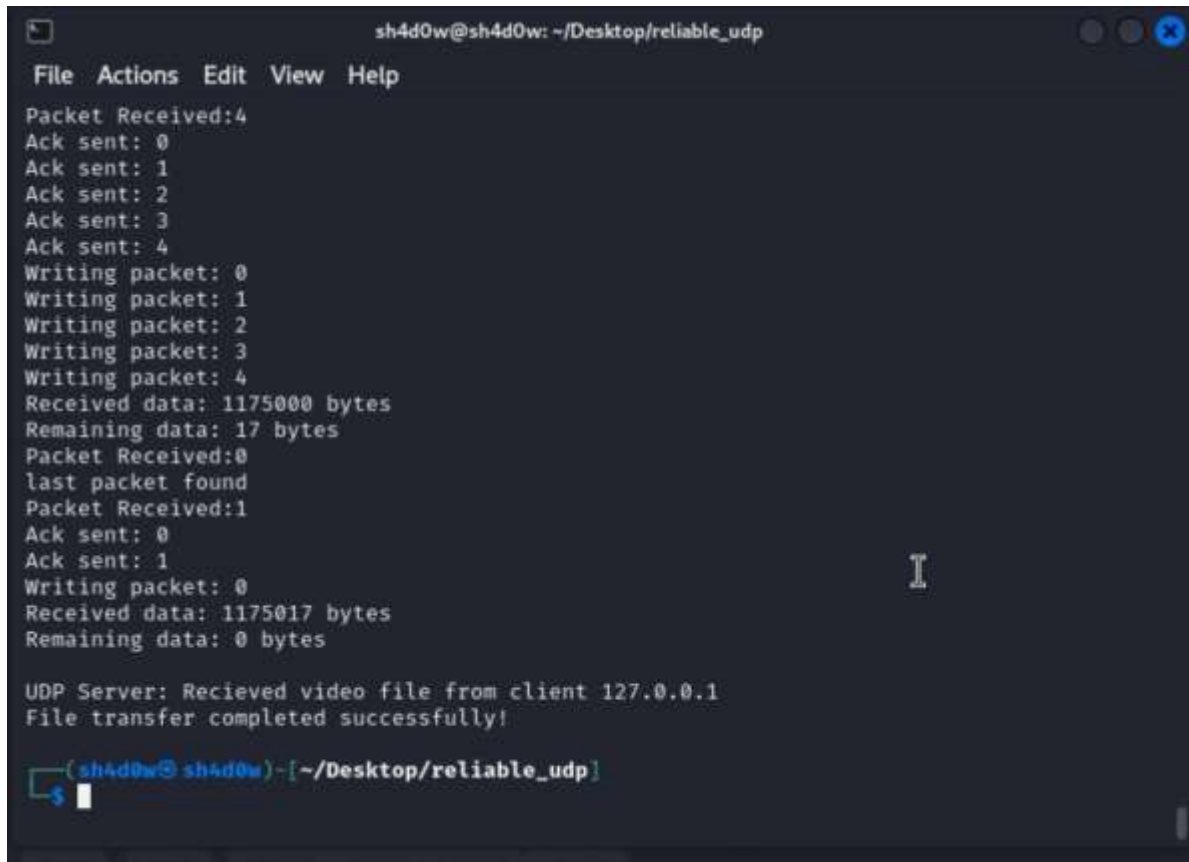
```

(sh4d0w@sh4d0w)-[~/Desktop/reliable_udp]
$ gcc udp_server.c -lpthread -o server

(sh4d0w@sh4d0w)-[~/Desktop/reliable_udp]
$ ./server
UDP Server: Waiting to recieve datagrams ...

```

Process Completed



```
sh4d0w@sh4d0w: ~/Desktop/reliable_udp
File Actions Edit View Help
Packet Received:4
Ack sent: 0
Ack sent: 1
Ack sent: 2
Ack sent: 3
Ack sent: 4
Writing packet: 0
Writing packet: 1
Writing packet: 2
Writing packet: 3
Writing packet: 4
Received data: 1175000 bytes
Remaining data: 17 bytes
Packet Received:0
last packet found
Packet Received:1
Ack sent: 0
Ack sent: 1
Writing packet: 0
Received data: 1175017 bytes
Remaining data: 0 bytes

UDP Server: Recieved video file from client 127.0.0.1
File transfer completed successfully!

(sh4d0w@sh4d0w)~[~/Desktop/reliable_udp]
$
```

File Explorer:



```
(sh4d0w@sh4d0w)~[~/Desktop/reliable_udp]
$ ls
README.md  client  media  output_video.mp4  project.pdf  server  udp_client.c  udp_server.c

(sh4d0w@sh4d0w)~[~/Desktop/reliable_udp]
$
```