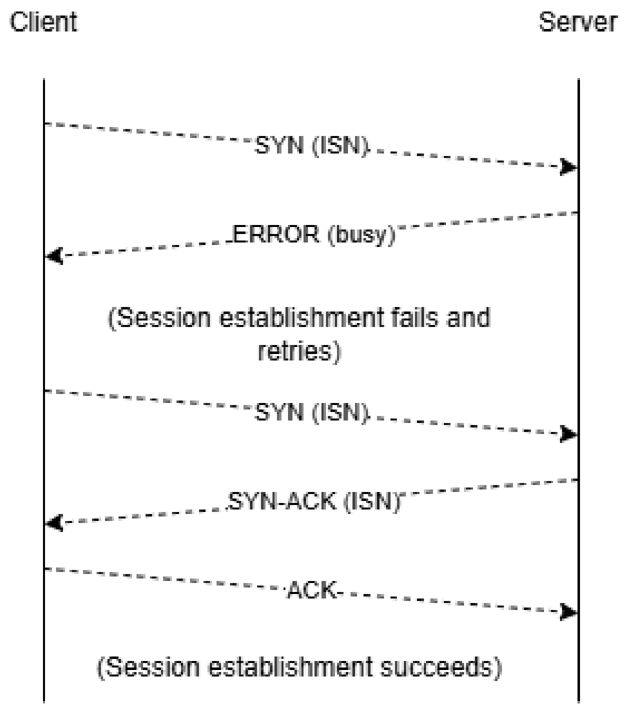
Mini-RFC Document for MCO1
Implementation of Reliable Data Transfer of UDP

I. Introduction

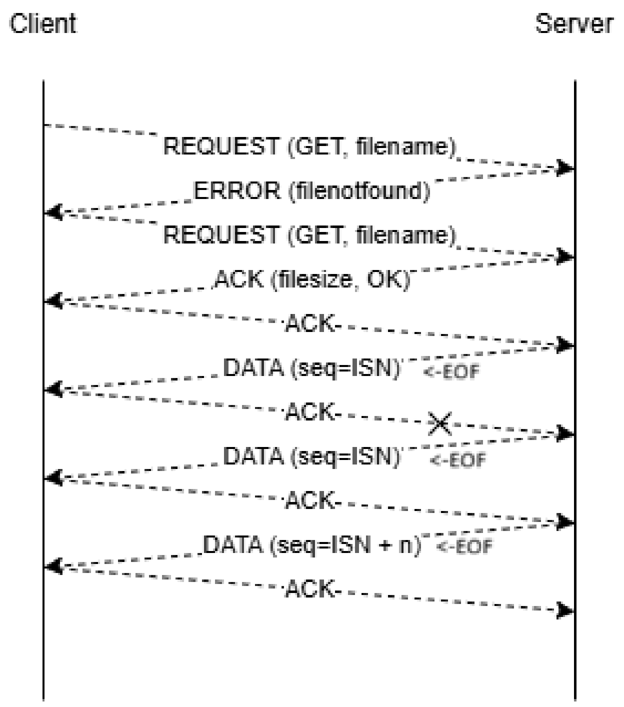
UDP is a connectionless transfer protocol. It has no flow control, packet ordering or acknowledgement. The objective of this MCO is to establish an application layer protocol on top of UDP specifically designed to take care of the previous constraints of the connectionless protocol.

II. Protocol Overview

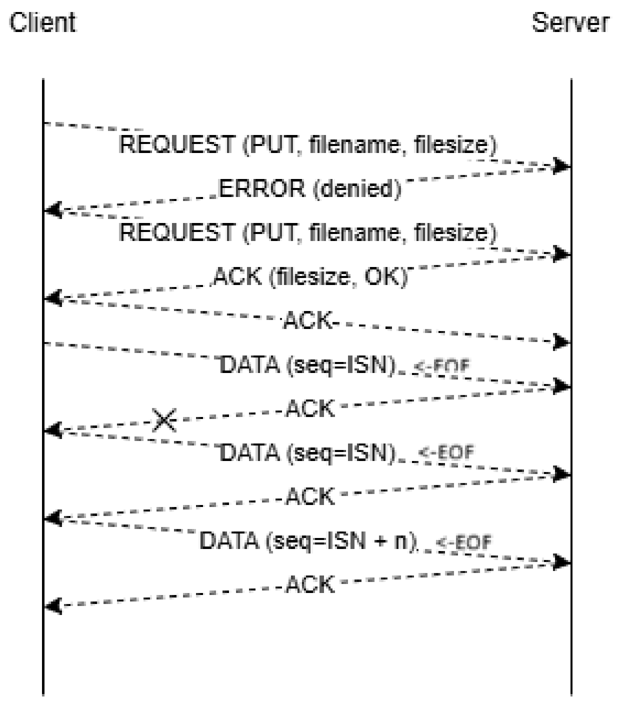
A. Session Error and Retry


Client
Server
SYN (ISN)
ERROR (busy)
(Session establishment fails and retries)
SYN (ISN)
SYN-ACK (ISN)
ACK
(Session establishment succeeds)

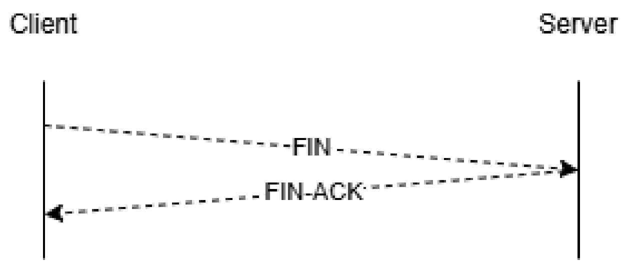
B. Download Error and Retry

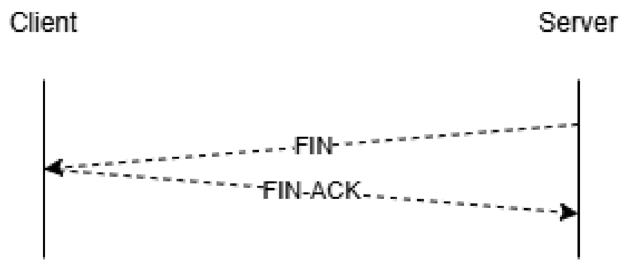


C. Upload Error and Retry



D. Session Termination from Client/Server





III. Packet Message Formats

A. 3 Way Handshake

The 3 way handshake taken from TCP contains SYN, SYN-ACK, and ACK. An initial sequence number (ISN), a 32 bit random integer, is generated by both the client and the server. The ACK contains the received SEQ and increments it by 1.

1.	**SYN:**	SYN SEQ=x
2.	**SYN-ACK:**	SYN-ACK SEQ=y ACK=x+1
3.	**ACK:**	ACK SEQ=y+1 ACK=x+1

B. Connection Termination

The connection termination contains FIN and FIN-ACK. It begins with a FIN then returns with a FIN-ACK.

1.	**FIN:**	FIN SEQ=x+1
2.	**FIN-ACK:**	FIN ACK=x+1

C. Error Handling

1.	**ERROR:**	ERROR type
		Type 0 = Timeout
		Type 1 = File not found
		Type 2 = Unexpected packets

D. Application Commands

These packets are used after the 3-Way Handshake to negotiate which file is being transferred and its size.

1. **GET**

a) Client Request: REQUEST GET <filename>
b) Server Response (Success): ACK filesize=<size> OK
c) Server Response (Failure): ERROR 1 (File not found)

2. **PUT**
    a) Client Request: REQUEST PUT <filename> <filesize>
    b) Server Response (Success): ACK filesize=<size> OK
    c) Server Response (Failure): ERROR 2

E. Data Packet

Data packets use a hybrid format: a UTF-8 encoded text header followed by a raw binary payload.

1. **DATA**: DATA SEQ=<seq> EOF=<eof>\n<binary_payload>
    a) EOF Types
        (1) Type 0 = More data to follow
        (2) Type 1 = EOF

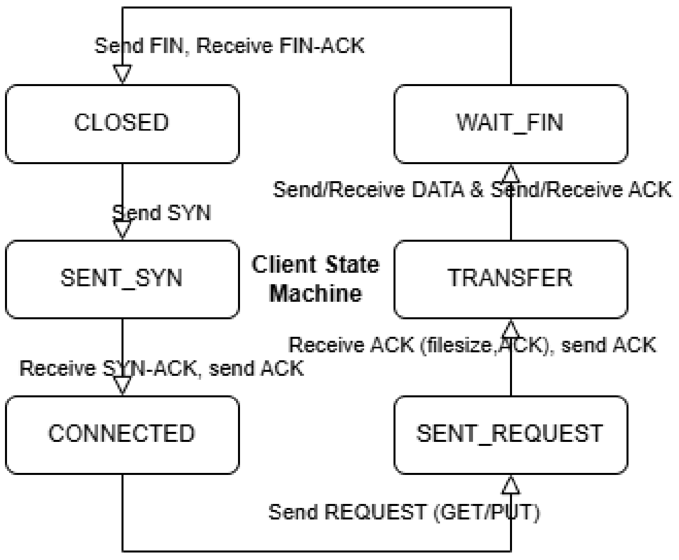
F. Acknowledgement

Sent by the receiver to confirm successful receipt of a specific data segment before the sender proceeds to the next block (Stop-and-Wait)

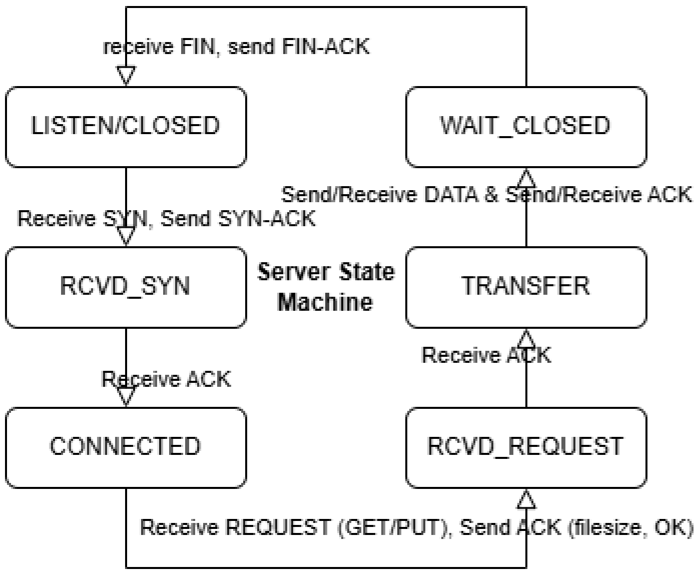
1. **ACK: ACK=<seq+1>**

IV. State Machines (Transition Diagram)

A. Client


Send FIN, Receive FIN-ACK
CLOSED
WAIT_FIN
Send/Receive DATA & Send/Receive ACK
Send SYN
SENT_SYN
Client State Machine
TRANSFER
Receive ACK (filesize,ACK), send ACK
Receive SYN-ACK, send ACK
CONNECTED
SENT_REQUEST
Send REQUEST (GET/PUT)

B. Server


receive FIN, send FIN-ACK
LISTEN/CLOSED
WAIT_CLOSED
Send/Receive DATA & Send/Receive ACK
Receive SYN, Send SYN-ACK
RCVD_SYN
Server State Machine
TRANSFER
Receive ACK
Receive ACK
CONNECTED
RCVD_REQUEST
Receive REQUEST (GET/PUT), Send ACK (filesize, OK)

V. Explanation of Reliability Mechanisms

A. Sequencing

Each packet is assigned a sequence number starting from the agreed ISN. Packets are sent and received in sequential order, with each subsequent packet incrementing the sequence number by one.

B. Acknowledgements

Upon receiving a data packet, the receiver sends an ACK containing the sequence number of the received packet, confirming successful delivery before the next packet is sent.

C. Retransmission

If an ACK is not received within a set timeout period, the sender assumes the packet was lost and retransmits it. The sender will not proceed to the next packet until the current one is acknowledged.

D. Ordered Delivery

Since only one packet is in flight at a time (Stop & Wait), packets are inherently delivered and processed in order. Once all packets are received, the receiver reassembles them by sequence number to reconstruct the original file.

VI. Error-handling

A. Error Codes

1. Code 0: Timeout

Given a configurable timeout, once the client requests for a file and the server does not reply within the timeout window a type 0 ERROR and the client has to ACK this. In the case that the server is down, the client-side will notify the client that the timeout window has passed and the connection will be closed. Once the server is back online, it will send the ERROR of type 0, and if the timeout window passes, it will close the half-open connection.

2. **Code 1: File Not Found**
   For when a client requests for a file name that doesn't exist in the server and/or the file size is different, then the server shall send a type 1 ERROR and must be ACKed by the client.
3. **Code 2: Unexpected Packets**
   The application will check incoming packets if they are within the expected range of the buffer. In the case where an incoming packet has a significantly different sequence number than the current, the server will send an ERROR with type 2. The client must ACK the error and proceed with retransmission if needed, else the connection will close both ways.

VII. File Transfer Operations

A. Download

To download a file, the client sends a REQUEST packet containing the filename to the server. If the file does not exist, the server responds with an ERROR (file not found) and the client may retry with a new request. If the file exists, the server responds with an ACK containing the file size, and the client replies with an ACK to confirm readiness, completing a triple handshake and officially beginning the transfer. The server then sends data packets sequentially, each tagged with an incrementing sequence number starting from the agreed ISN. The client sends an ACK for each received packet. If the server does not receive an ACK within the timeout period, it retransmits the current packet before proceeding. Since only one packet is in flight at a time, packets are received in order and no reassembly is needed. Once the final packet is acknowledged, the download operation is complete.

B. Upload

To upload a file, the client sends a REQUEST packet containing the filename and file size to the server. If the server cannot accept the file, it responds with an ERROR (denied) and the operation is aborted. If the server is ready, it responds with an ACK confirming the file size, and the client replies with an ACK to confirm readiness, completing a triple handshake and officially beginning the transfer. The client then sends data packets sequentially, each tagged with an incrementing sequence number starting from the agreed ISN. The server sends an ACK for each received packet. If the client does not receive an ACK within the timeout period, it retransmits the current packet before proceeding. Once the final packet is acknowledged by the server, the upload operation is complete.

VIII. End-of-File Signaling

To signal the end of a file transfer, the sender marks the final data packet with an EOF flag set to 1 in the packet header. All preceding packets carry EOF=0. Upon receiving a packet with EOF=1, the receiver recognizes it as the last packet, sends a final ACK, and considers the transfer complete. This provides an explicit and unambiguous end-of-file signal that works in tandem with the file size agreed upon during the REQUEST phase, ensuring both sides cleanly exit the data transfer phase without ambiguity.