1. Message Formats

1) FIRST 클라이언트가 처음 업로드할 때 보내는 메시지.

[FIRST <ClientID> <FileName> <FileSize>]

의미)

처음 업로드를 시작할 때 사용

서버는 해당 클라이언트용 디렉토리가 없으면 새로 만들고

파일이 이미 존재하면 offset 체크 후 이어받을 준비

2) RESUME 업로드가 중간에 끊겼을 때 클라이언트가 재접속하며 보냄

[RESUME <ClientID> <FileName>]

의미)

클라이언트가 이어서 보내고 싶다고 알림

서버는 현재 저장된 파일의 크기(offset)를 응답으로 전송

3) ACK

[ACK <StoredOffset>]

의미)

서버가 저장해둔 파일 크기(offset)를 알려줌

클라이언트는 여기부터 이어서 전송 시작

4) DATA

[DATA <ChunkSize>₩n
<binary payload>]

의미)

실제 파일 데이터 보내는 부분

ChunkSize 만큼 데이터를 보낸다는 의미

서버는 해당 데이터를 append

5) FIN

[FIN]

의미)

클라이언트가 파일 데이터를 모두 보냈음을 서버에게 알리는 메시지

6) COMPLETE

[COMPLETE]

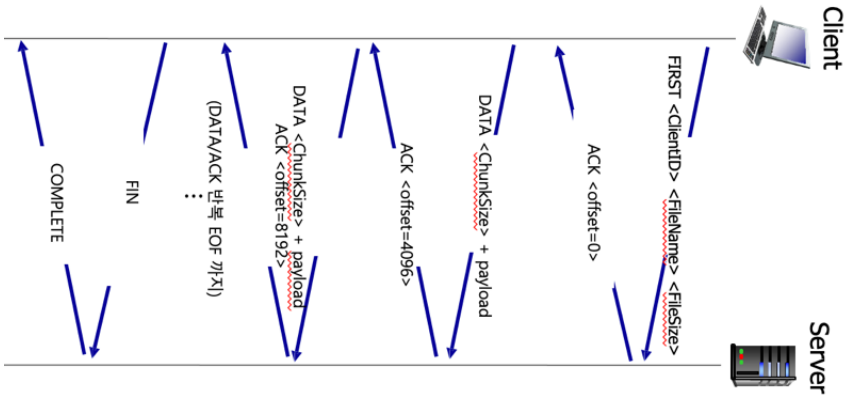
의미)

서버가 업로드 전체를 성공적으로 받았다는 의미

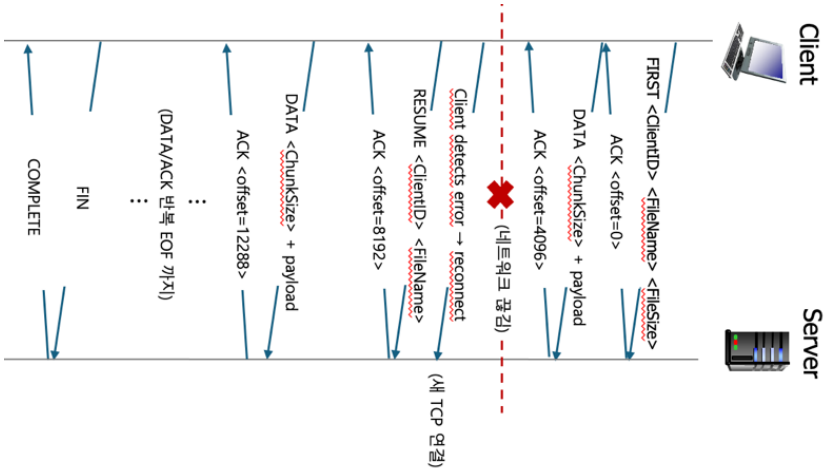
클라이언트는 프로그램 종료

2. Sequence Diagram

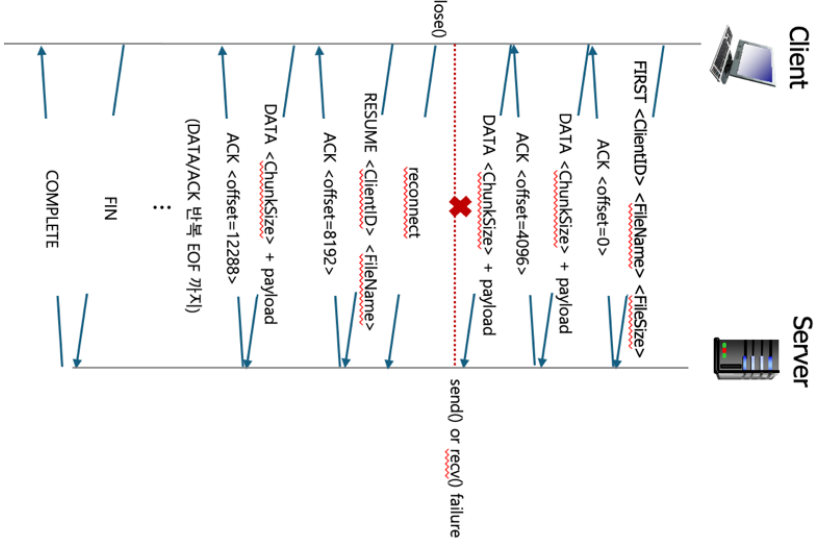
1) 정상 업로드



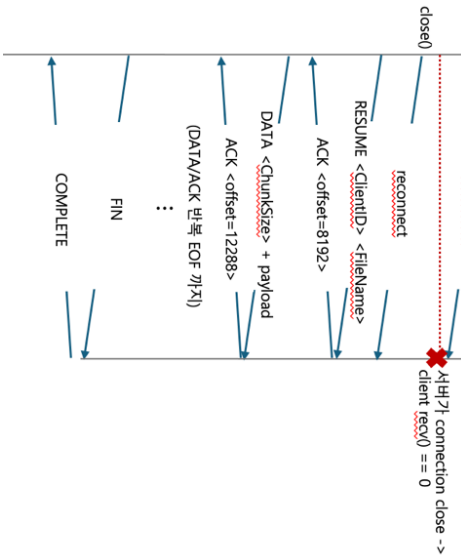
2) 업로드 중단 → 재접속 → Resume



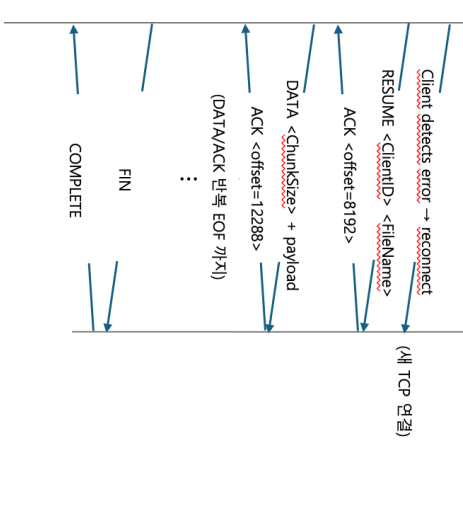
3) send()/recv() 즉시 실패



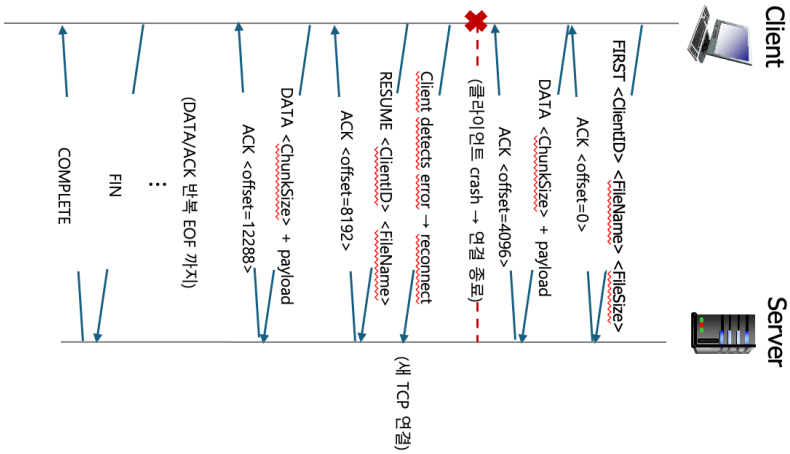
4) peer closed connection (read() == 0)



5) 서버 Crash → 재시작 → Resume



6) 클라이언트 Crash → 재시작 → Resume



3) Error Handling and Policies

1) write()/read() or send()/recv() failure

: Client detects the failure immediately from the return value of send/recv or write/read.

Close the socket.

Reconnect to server.

Send: RESUME <ClientID> <FileName>

Receive: ACK <offset>

Resume upload from that offset.

2) peer closed connection (like read() == 0)

: Client receives read() == 0, meaning the peer performed a close().

Close the socket.

Reconnect to server.

Send: RESUME <ClientID> <FileName>

Receive: ACK <offset>

Resume upload from that offset.

3) Connection reset

: send()/recv() returns -1

Close the socket.

Reconnect to server.

Send: RESUME <ClientID> <FileName>

Receive: ACK <offset>

Resume upload from that offset.

4) Server process crashes and restarts

: Client receives send/recv failure or connection drop while sending data.

Client closes the socket.

Client reconnects when the server is available again.

Client sends RESUME <ClientID> <FileName>.

Server checks the stored file and replies with ACK <offset>.

Client seeks to offset and continues uploading.

5) Client process crashes and restarts

: The client itself restarts, and the previous TCP connection no longer exists.

After restarting, the client reconnects to the server.

Instead of sending FIRST, it sends RESUME <ClientID> <FileName>.

Server replies with ACK <offset> based on stored data.

Client seeks to offset and continues sending from that position.

6) Multiple interruptions during a single upload

: Any of the previous failure types (1~5) repeatedly occur.

For every interruption:

1. Detect the failure.
2. Close the socket.
3. Reconnect to the server.
4. Client sends: RESUME <ClientID> <FileName>
5. Server replies: ACK <offset>
6. Client seeks to offset and resumes uploading.

This cycle continues until FIN/COMPLETE is reached.