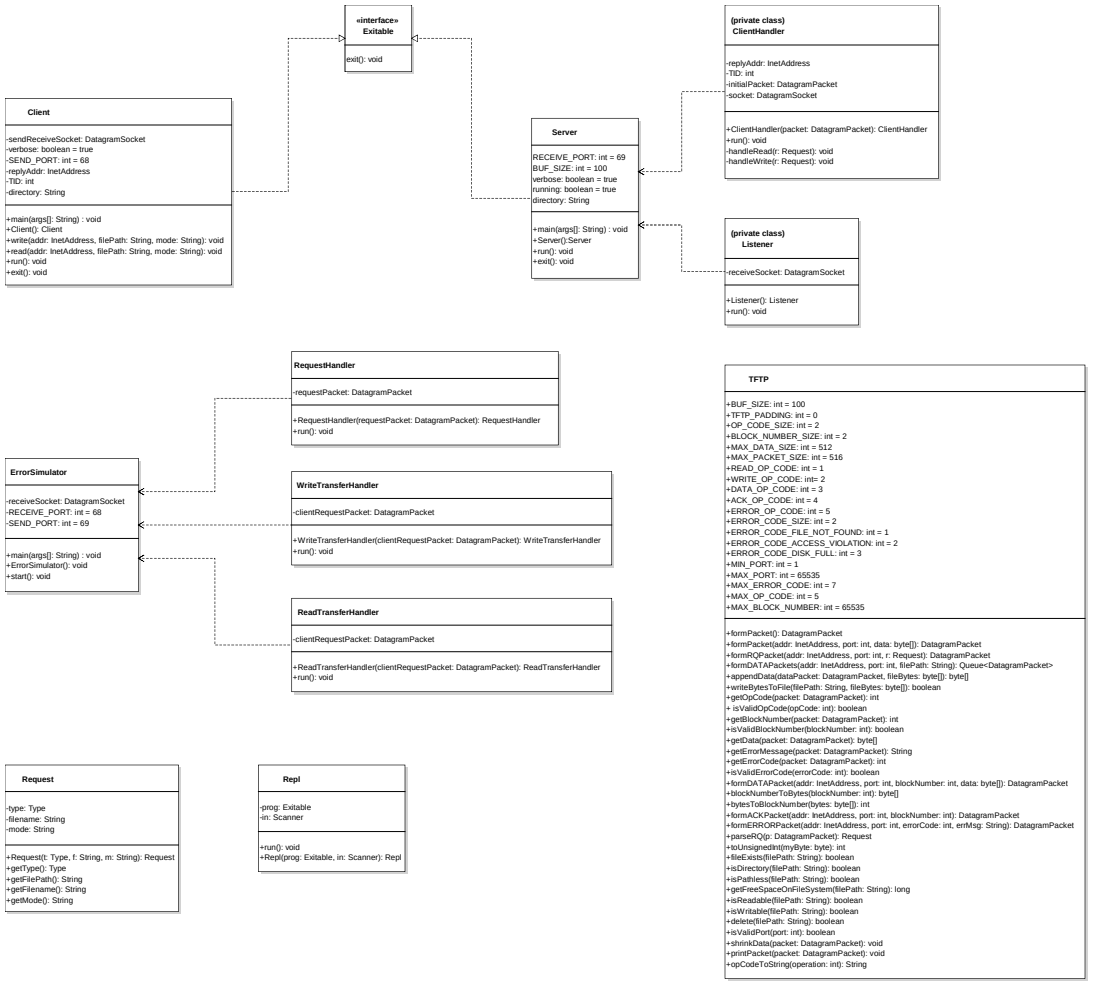
Diagrams for Iteration #2

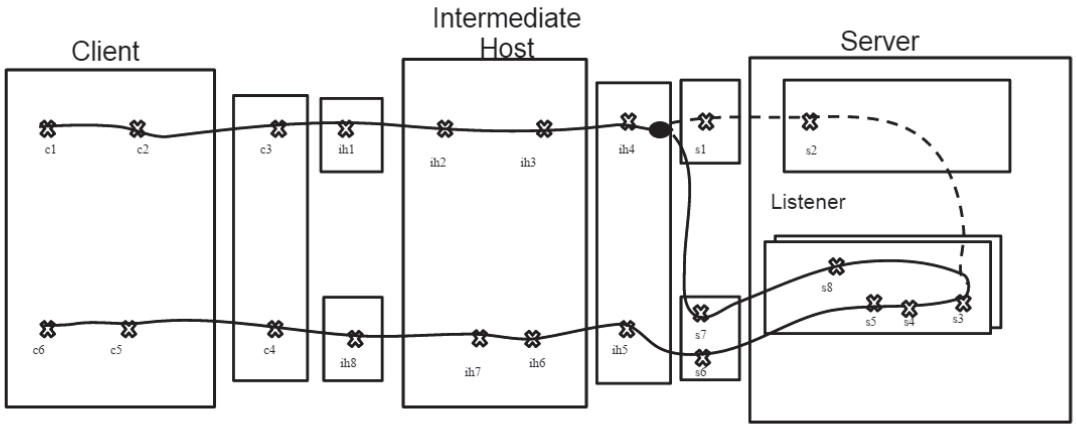
-including class diagram, unchanged UCM diagrams, and timing diagram for error code 1, 2, 3

By: Team 4

Date: 19th May, 2015


«interface»
Exitable
exit(): void
Client
-sendReceiveSocket: DatagramSocket
-verbose: boolean = true
-SEND_PORT: int = 68
-replyAddr: InetAddress
-TID: int
-directory: String
+main(args[]: String) : void
+Client(): Client
+write(addr: InetAddress, filePath: String, mode: String): void
+read(addr: InetAddress, filePath: String, mode: String): void
+run(): void
+exit(): void
Server
RECEIVE_PORT: int = 69
BUF_SIZE: int = 100
verbose: boolean = true
running: boolean = true
directory: String
+main(args[]: String) : void
+Server():Server
+run(): void
+exit(): void
(private class)
ClientHandler
-replyAddr: InetAddress
-TID: int
-initialPacket: DatagramPacket
-socket: DatagramSocket
+ClientHandler(packet: DatagramPacket): ClientHandler
+run(): void
-handleRead(r: Request): void
-handleWrite(r: Request): void
(private class)
Listener
-receiveSocket: DatagramSocket
+Listener(): Listener
+run(): void
RequestHandler
-requestPacket: DatagramPacket
+RequestHandler(requestPacket: DatagramPacket): RequestHandler
+run(): void
ErrorSimulator
-receiveSocket: DatagramSocket
-RECEIVE_PORT: int = 68
-SEND_PORT: int = 69
+main(args[]: String) : void
+ErrorSimulator(): void
+start(): void
WriteTransferHandler
-clientRequestPacket: DatagramPacket
+WriteTransferHandler(clientRequestPacket: DatagramPacket): WriteTransferHandler
+run(): void
ReadTransferHandler
-clientRequestPacket: DatagramPacket
+ReadTransferHandler(clientRequestPacket: DatagramPacket): ReadTransferHandler
+run(): void
TFTP
+BUF_SIZE: int = 100
+TFTP_PADDING: int = 0
+OP_CODE_SIZE: int = 2
+BLOCK_NUMBER_SIZE: int = 2
+MAX_DATA_SIZE: int = 512
+MAX_PACKET_SIZE: int = 516
+READ_OP_CODE: int = 1
+WRITE_OP_CODE: int= 2
+DATA_OP_CODE: int = 3
+ACK_OP_CODE: int = 4
+ERROR_OP_CODE: int = 5
+ERROR_CODE_SIZE: int = 2
+ERROR_CODE_FILE_NOT_FOUND: int = 1
+ERROR_CODE_ACCESS_VIOLATION: int = 2
+ERROR_CODE_DISK_FULL: int = 3
+MIN_PORT: int = 1
+MAX_PORT: int = 65535
+MAX_ERROR_CODE: int = 7
+MAX_OP_CODE: int = 5
+MAX_BLOCK_NUMBER: int = 65535
+formPacket(): DatagramPacket
+formPacket(addr: InetAddress, port: int, data: byte[]): DatagramPacket
+formRQPacket(addr: InetAddress, port: int, r: Request): DatagramPacket
+formDATAPackets(addr: InetAddress, port: int, filePath: String): Queue<DatagramPacket>
+appendData(dataPacket: DatagramPacket, fileBytes: byte[]): byte[]
+writeBytesToFile(filePath: String, fileBytes: byte[]): boolean
+getOpCode(packet: DatagramPacket): int
+ isValidOpCode(opCode: int): boolean
+getBlockNumber(packet: DatagramPacket): int
+isValidBlockNumber(blockNumber: int): boolean
+getData(packet: DatagramPacket): byte[]
+getErrorMessage(packet: DatagramPacket): String
+getErrorCode(packet: DatagramPacket): int
+isValidErrorCode(errorCode: int): boolean
+formDATAPacket(addr: InetAddress, port: int, blockNumber: int, data: byte[]): DatagramPacket
+blockNumberToBytes(blockNumber: int): byte[]
+bytesToBlockNumber(bytes: byte[]): int
+formACKPacket(addr: InetAddress, port: int, blockNumber: int): DatagramPacket
+formERRORPacket(addr: InetAddress, port: int, errorCode: int, errMsg: String): DatagramPacket
+parseRQ(p: DatagramPacket): Request
+toUnsignedInt(myByte: byte): int
+fileExists(filePath: String): boolean
+isDirectory(filePath: String): boolean
+isPathless(filePath: String): boolean
+getFreeSpaceOnFileSystem(filePath: String): long
+isReadable(filePath: String): boolean
+isWritable(filePath: String): boolean
+delete(filePath: String): boolean
+isValidPort(port: int): boolean
+shrinkData(packet: DatagramPacket): void
+printPacket(packet: DatagramPacket): void
+opCodeToString(operation: int): String
Request
-type: Type
-filename: String
-mode: String
+Request(t: Type, f: String, m: String): Request
+getType(): Type
+getFilePath(): String
+getFilename(): String
+getMode(): String
Repl
-prog: Exitable
-in: Scanner
+run(): void
+Repl(prog: Exitable, in: Scanner): Repl

UCM Read Request:

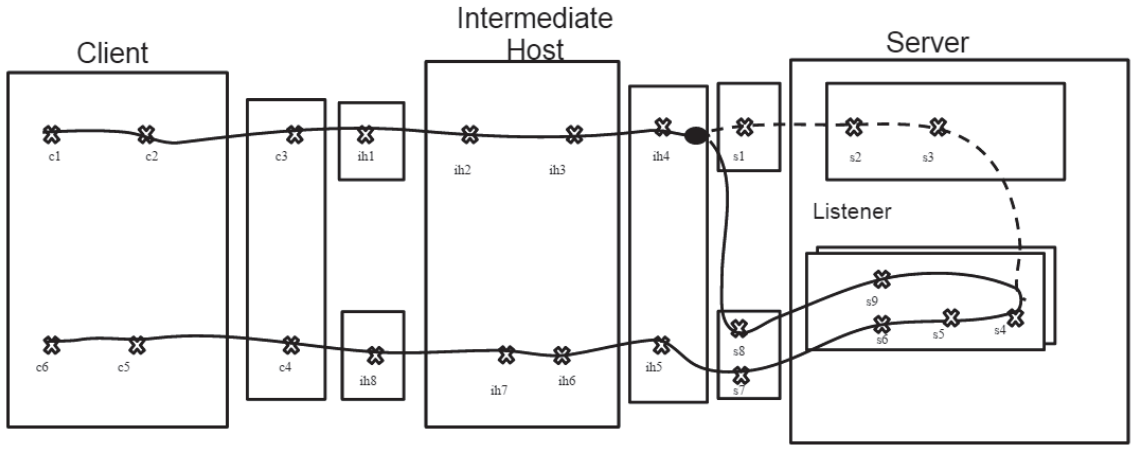


c1: form message
c2: create datagram
c3: send datagram
c4: wait and receive response
c5: extract message
c6: output message

ih1: wait and receive packet
ih2: extract message
ih3: form packet
ih4: send packet
ih5: wait and receive response
ih6: extract message
ih7: form packet
ih8: send packet

s1: receive datagram
s2: extract message
s3: verify message (quit if error)
s4: form response and data
s5: form datagram
s6: send datagram
s7: receive datagram
s8: extract message

UCM Write Request:



c1: form message
c2: create datagram
c3: send datagram
c4: wait and receive response
c5: extract message
c6: output message

ih1: wait and receive packet
ih2: extract message
ih3: form packet
ih4: send packet
ih5: wait and receive response
ih6: extract message
ih7: form packet
ih8: send packet

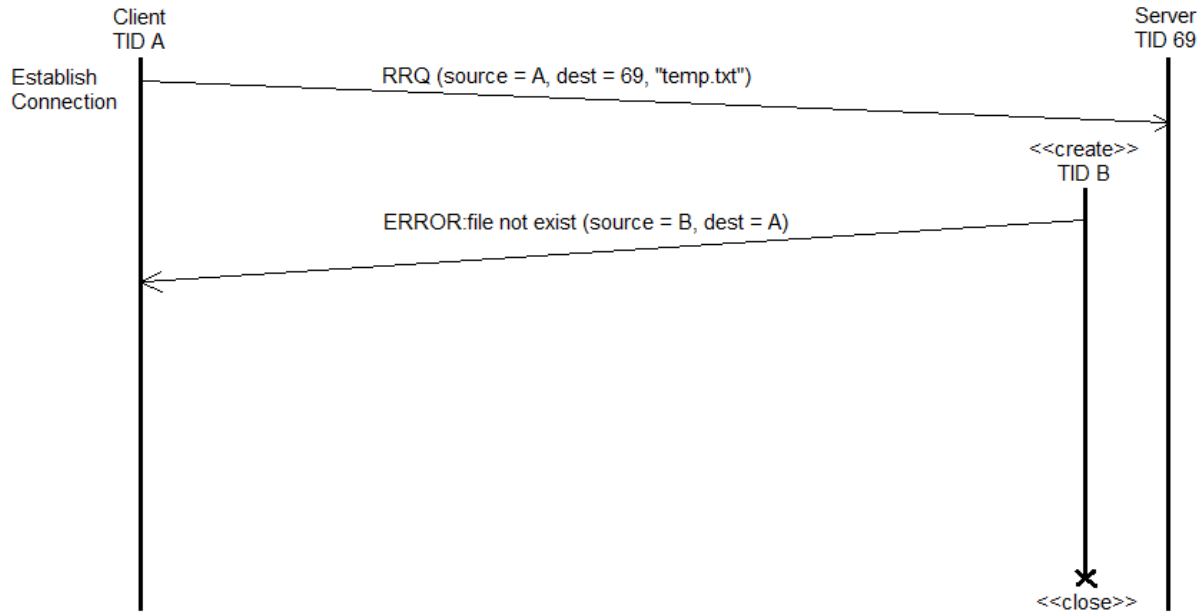
s1: receive datagram
s2: extract message
s3: create socket
s4: verify message (quit if error)
s5: form response and Ack
s6: form datagram
s7: send datagram
s8: receive datagram
s9: extract message

Timing diagrams for iteration #2

[Error Code 1]

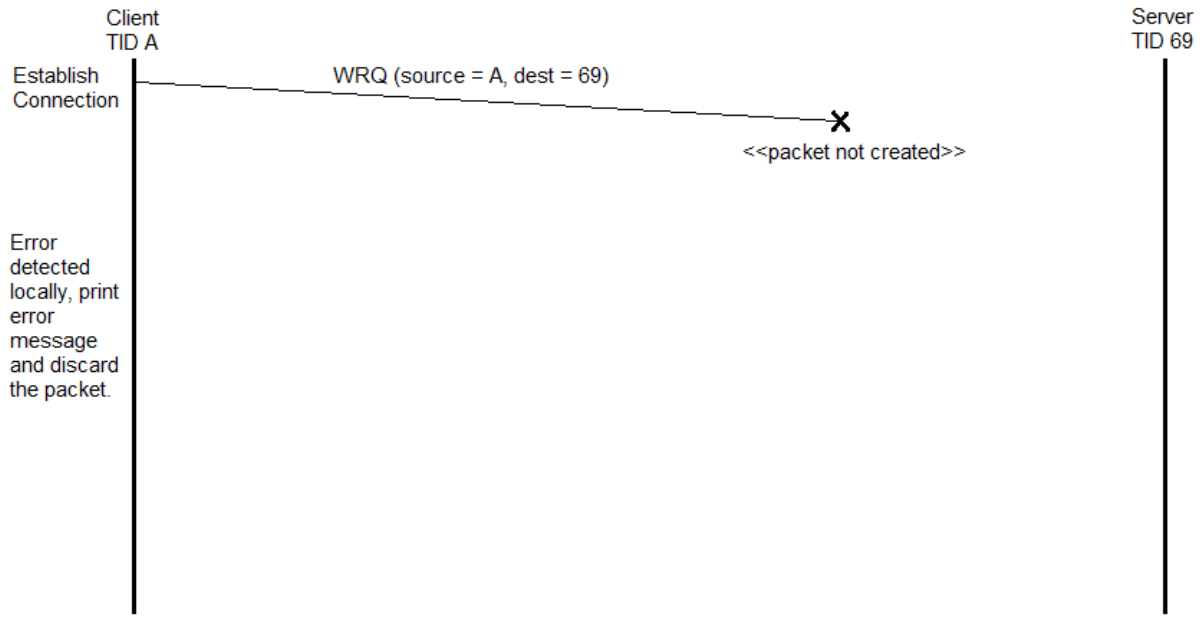
Scenario 1 - RRQ File not found on server:

1. Client sends RRQ.
2. Server detects that file does not exist.
3. Server forms error packet and sends back to client.
4. Server closes its socket with client.
5. Client displays error message.



Scenario 2 - WRQ File not found on Client:

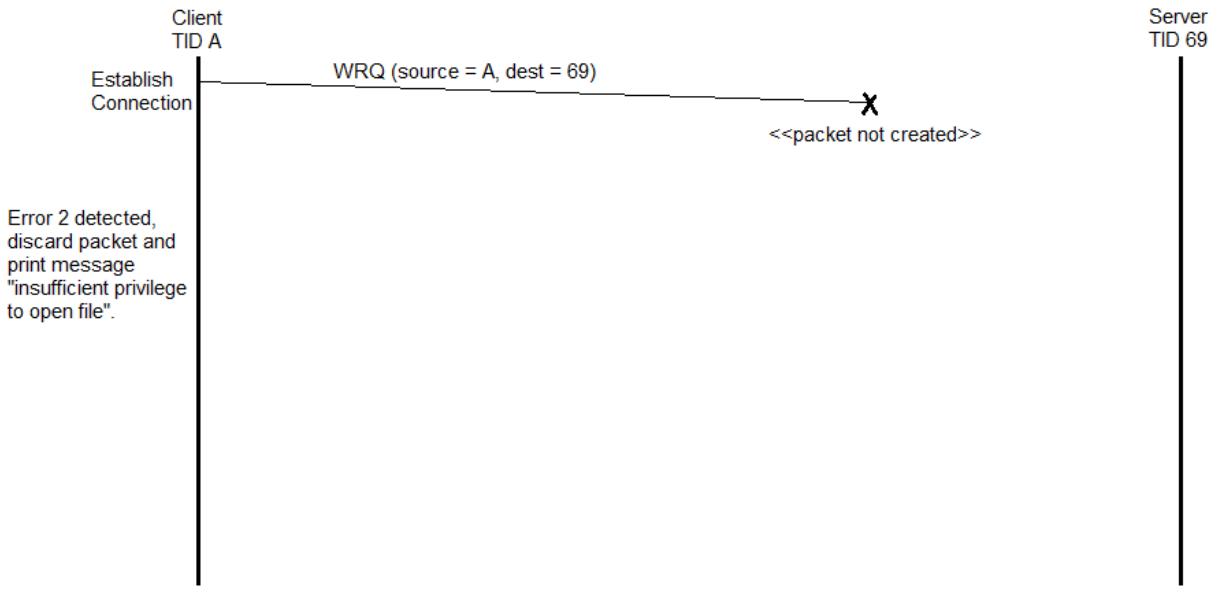
1. User initiates WRQ.
2. Client detects file does not exist on client side.
3. Packet not created, error message displayed.



[Error Code 2]

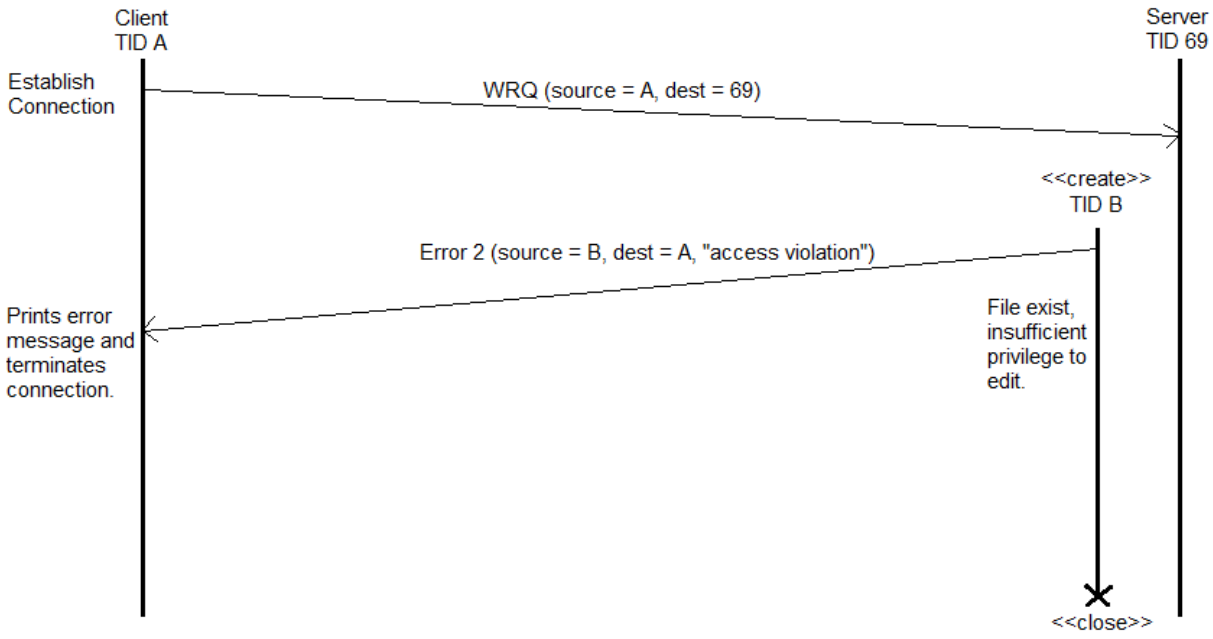
Scenario 1 - WRQ Access violation on client:

1. User initiates WRQ.
2. Client detects that it has insufficient privileges to open file to be sent.
3. Client displays error message



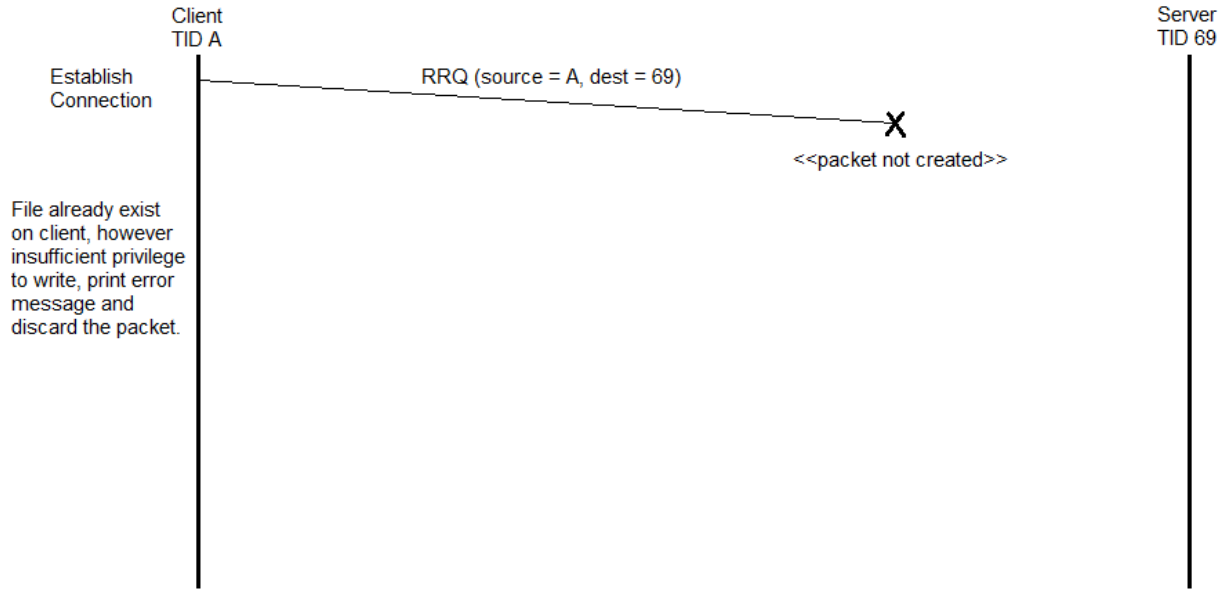
Scenario 2 - WRQ Access violation on server:

1. Client sends WRQ to server.
2. Server detects that file already exists and that client has insufficient privileges to write to file.
3. Server sends ERROR packet to client.
4. Server closes its connection with the client.
5. Client displays error message.



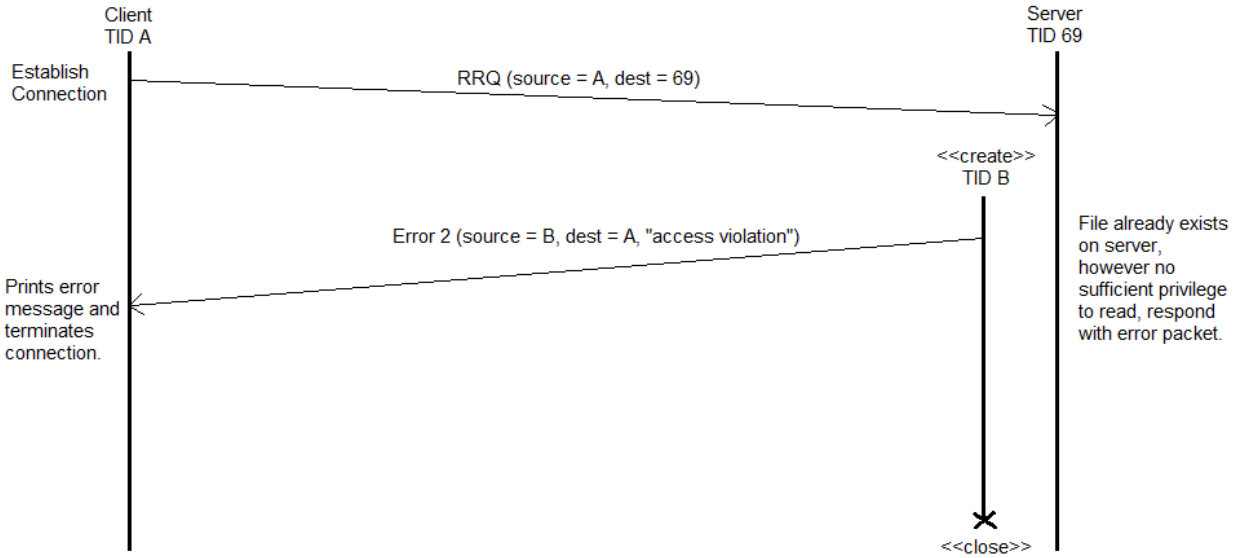
Scenario 3 - RRQ Access violation on client:

1. User initiates a RRQ.
2. Client detects that file already exists.
3. Client detects insufficient privileges to write to file.
4. Client displays error message.



Scenario 4 - RRQ Access violation on client:

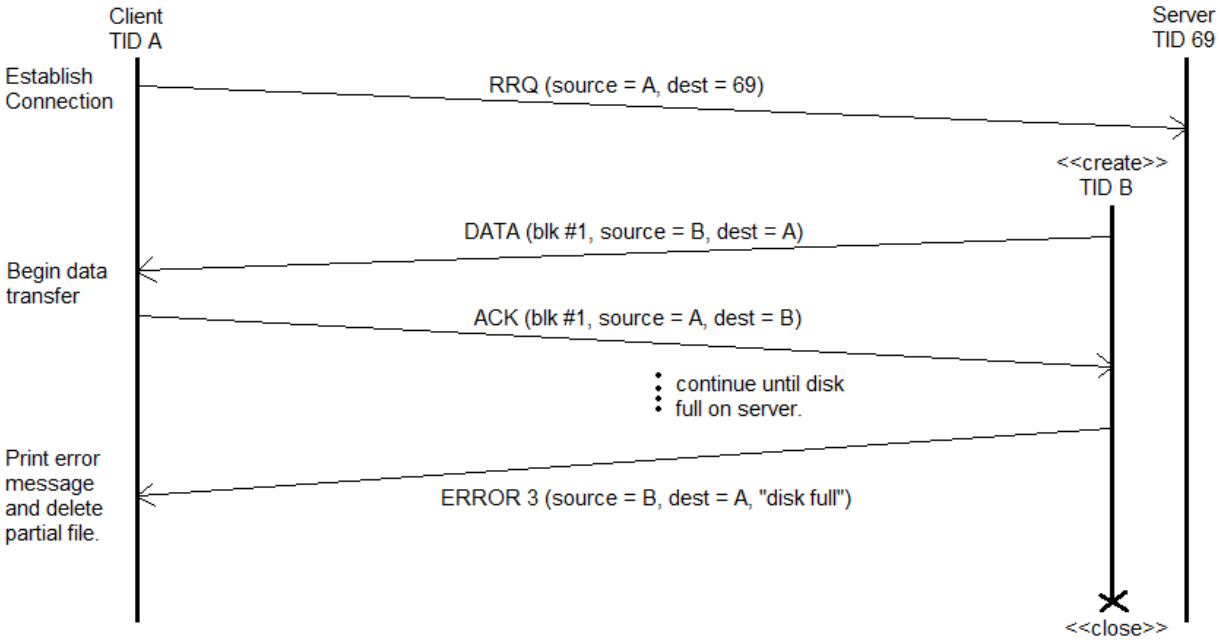
1. User initiates a RRQ
2. Client sends RRQ to Server
3. Server detects that file exists but has insufficient privileges to open.
4. Server sends access violation error packet to client
5. Client displays error message.



[Error Code 3]

Scenario 1 - Disk full on client:

1. User initiates RRQ
2. Client sends RRQ to Server
3. Server sends first data packet
4. Client acknowledges
5. Continues until client detects disk is full
6. Client sends Disk Full error packet
7. Server closes socket and closes file
8. Client deletes incomplete data and displays message



Scenario 2 - Disk full on server:

1. User initiates WRQ
2. Client sends WRQ to Server
3. Server responds with ACK
4. Client sends data packet
5. Continues until Server detects disk full
6. Server sends Disk Full error packet
7. Client displays error message
8. Server closes socket and deletes incomplete data
9. Client closes file