TFTP File Transfer System

Team 11:

Lau, James	101009129
Li, Yinan	101012609
Liu, Meitong	101014282
Zhong, Runhe	101016659

Carleton University

Department of Systems and Computer Engineering

SYSC 3303 - Real-Time Concurrent Systems - Summer 2018

Table of Contents

Purpose of This Project

To design and implement a file transfer system based on the TFTP specification (RFC 1350). The system consists of TFTP client(s) running on one or several computers, an error simulator, and a multithreaded TFTP server that runs on a different computer. The code is written in Java, using the Eclipse IDE. Each program (client, error simulator, and server) runs as a separate Win-32 process, and the programs communicate via DatagramSocket objects.

In “normal” mode, only the client and server programs run. In “test” mode, all three programs are used. Multiple main programs are able to ran concurrently.

Responsibilities

Iteration 1

Runhe Zhong :

- Group leader. Manage github repository
- Assigned tasks to group members
- Code review
- Drew UML and UCM diagrams
- Wrote README.txt

James Lau :

- Worked on the Client side and UI

Yinan Li :

- Worked on the Server side

Meitong Liu :

- Worked on the Error Simulator side
- Drew UCM diagrams
- Wrote README.txt

Iteration 2

Runhe Zhong :

- Group leader
- Assigned tasks to group members
- Code review
- Collaborated on Error Simulator
- Drew UML diagram
- Wrote README.txt

James Lau :
Worked on the Client side

Yinan Li :
Worked on the Server side

Meitong Liu :
Collaborated on Error Simulator
Drew UML and Timing diagrams
Wrote README.txt

Iteration 3

Runhe Zhong :
Group leader
Assigned tasks to group members
Code review
Collaborated on Error Simulator
Drew UML diagram
Wrote README.txt

James Lau :
Worked on the Client side

Yinan Li :
Worked on the Server side

Meitong Liu :
Collaborated on Error Simulator
Drew UML and Timing diagrams
Wrote README.txt

Iteration 4 & 5

Runhe Zhong :

Group leader.
Assigned tasks to group members
Code review
Collaborated on Error Simulator
Drew UML diagram
Wrote README.txt

James Lau :

Worked on the Client side

Yinan Li :

Worked on the Server side

Meitong Liu :

Collaborated on Error Simulator
Drew UML and Timing diagrams
Wrote README.txt

Setup Instruction

---

1. extract .zip file
2. open Eclipse and set workspace path
3. run Server.java and Client.java (No ErrorSimulator.java)
4. go to the console page of Client and start testing the program

Test Instruction

```
User Instruction (Error Simulator, Please select the error user want to
simulate
in the Error Simulator(UI) first then go to client console(UI) to start
select ):


--------------------------------------
Prompt User for IP Address:

---------- Please Input Destination IP Address ----------
    0. Local host (same ip)
    1. Other IP address
>>>>>>>> input quit to exit this program

If User choose 0, set IP Address to Localhost IP
If USer choose 1, prompt IP Address ---------------->>>          ----------
Please Input Destination IP Address ----------
                                                >>>>>>>> input quit to exit
this program

If user input quit, the error simulator's listener will stop

Then go to Error Main Menu
--------------------------------------

  Error Main Menu:

            ----------Error Selection----------
                 0. Normal Operation
                 1. Transmission Error
                 2. Error Codes (4 or 5)
                 >>>>>>>> input quit to exit this program

If user input quit, the error simulator's listener will stop
-------------------------------------------------------------------------
After selecting error type:

            If user select 0, error simulator start working without
simulating any error
            If user select 1, Transmission Error Menu will show
            If user select 2,Packet Selection Menu will show
            If user input quit, the error simulator's listener will stop
            If user do invalid input, the UI will show "Invalid input,
please try again." and back to prompt user for IP address.
```

After selecting 1 in error main menu (Show Transmission Error):

```
          ---------- Transmission Error ----------
              1. Lose a packet
              2. Delay a packet
              3. Duplicate a packet
              4. Back to Error main menu
          >>>>>>>> input quit to exit this program
```

(User choose an error to simulate)

After selecting transmission error:

If user select 1, go to packet selection menu
If user select 2, go to packet selection menu
If user select 3, go to packet selection menu
If user select 4, back to prompt user for IP address.
If user input quit, the error simulator's listener will stop
If user do invalid input, the UI will show "Invalid input, please try again." and show transmission error menu again.

Show the packet selection menu:

```
          ---------- Packet Selection ----------
              1. RRQ
              2. WRQ
              3. DATA
              4. ACK
              5. ERROR
              6. Back to Error Menu
          >>>>>>>> input quit to exit this program
```

(User choose an packet to simulate the error)

After selecting packet:

If user select 1 and transmission error is "Delay a packet", then go to delay selection menu.
If user select 2 and transmission error is "Delay a packet", then go to delay selection menu.
If user select 3, go to block selection menu to decide the block# of packet to simulate error,
then if transmission error is "Delay a packet", then go to delay selection menu.
If user select 4, go to block selection menu to decide the block# of packet to simulate error,
then if transmission error is "Delay a packet", then go to

```
delay selection menu.
            (User cannot select 5 temporarily)
            If user select 6, back to prompt user for IP address
            If user input quit, the error simulator's listener will stop
            If user do invalid input, the UI will show "Invalid input,
please try again." and show packet menu again.
-------------------------------------------------------------------------
----------------------------------------
Show Delay Selection Menu:          |Show Block Selection Menu:
                                    |
   ------- Delay Selection -------  |   ---- Block Selection ----------
    Please enter delay time (ms)...             |     Please enter block
number...
    Enter -1 to go back to Error Menu           |     Enter -1 to go back
to Error Menu
>>>>>>>> input quit to exit this program        |>>>>>>>> input quit to
exit this program
                                    |
(User input number to decide delay time)        |(User choose a block of
ACK/DATA packet to simulate the error)
                                    |
                                    |
-------------------------------------------------------------------------
----------------------------------------
After finish all selection, error simulator start listening and the next
loop for Error main menu start.
User can change error Selection to modify the error for simulating at any
time.

(Since the error simulator will print the packet information when request
transfer, the new error menu will be hard to find.
To change the error selection, just input 0(Normal Operation),
1(Transmission Error), 2(Error code Error) or quit and the next
menu will show)
-------------------------------------------------------------------------
----------------------------------------
After selecting 2 in error main menu (Show Error Code Menu):

      Show the packet selection menu:

            ---------- Packet Selection ----------
                1. RRQ
                2. WRQ
                3. DATA
                4. ACK
                5. ERROR
                6. Back to Error Menu
            >>>>>>>> input quit to exit this program
```

```
(User choose an packet to simulate the error)
--------------------------------------------------------------------------
----
After selecting packet:

           If user select 1 or 2 or 5, then go to ErrorCode Menu.
           If user select 3 or 4, go to block selection menu to decide
the block# of packet to simulate error,
           If user select 6, back to prompt user for IP address
           If user input quit, the error simulator's listener will stop

           If user do invalid input, the UI will show "Invalid input,
please try again." and show packet menu again.

------------>    Show Block Selection Menu: (If User choose DATA or ACK
to simulate error)
                ---------- Block Selection ----------
                Please enter block number...
                Enter -1 to go back to Error Menu
                >>>>>>>> input quit to exit this program
--------------------------------------------------------------------------
--------------------------------------------------------------------------
------------
If User choose RRQ or WRQ |If User choose DATA or ACK |If User choose
ERROR
---------- Error Code Error ----------|---------- Error Code Error
----------          |---------- Error Code Error ----------
    1. Invalid Mode           (Error Code 4) |    1. Invalid Opcode
(Error Code 4)  |    1. Invalid Opcode         (Error Code 4)
    2. Invalid Opcode         (Error Code 4) |    2. Invalid Packet Format
(Error Code 4)    |    2. Invalid Packet Format           (Error Code 4)
    3. Invalid Filename       (Error Code 4) |    3. Invalid Packet Size
(Error Code 4)  |    3. Unknown User TID               (Error Code 5)
    4. Invalid Packet Format     (Error Code 4) |    4. Unknown User TID
       (Error Code 5)  |    4. Back to Error main menu
    5. Unknown User TID       (Error Code 5) |    5. Back to Error main
menu              |>>>>>>>> input quit to exit this program
    6. Back to Error main menu               |>>>>>>>> input quit to exit
this program      |
>>>>>>>> input quit to exit this program      |
       |
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------
After selection, the program will prompt user for input to create error.
(There are some errors do not need user to Input)

Note:           ---------- Please Input a new Packet Format ----------
```

```
            (This error will modify the zero padding between Filename and
Mode in RRQ/WRQ)
            Enter -1 to go back to Error Menu
            >>>>>>>> input quit to exit this program

This will prompt user for any number to modify the zero padding between
Filename and Mode in RRQ/WRQ
-------------------------------------------------------------------------
-------------------------------------------------------------------------
--

User Instruction (Client Side):
---------------------------------------

   Main Menu:
            ------Welcome to TFTP Client System------
            Please choose your operation:
                  1. RRQ
                  2. WRQ
                  3. Set Target IP
                  4. Set Working Directory
                  5. Set Print Mode (Quiet/Verbose)
                  6. Quit
-------------------------------------------------------------------------
If select 1. RRQ
            Please select your mode:
                  1. Normal  <Client, Server>
                  2. Test  <Client, Error Simulator, Server>
                  3. Back to main menu
-------------------------------------------------------------------------
If select 2. WRQ
            Please select your mode:
                  1. Normal  <Client, Server>
                  2. Test  <Client, Error Simulator, Server>
                  3. Back to main menu
-------------------------------------------------------------------------
If select 3. Set Target IP

            Please select your destination
                  1. Local host
                  2. I have my own destination
                  3. Back to main menu
-------------------------------------------------------------------------
If select 4. Set Working Directory

            Please select your location
                  1. src\client\files\
                  2. I have my own directory
                  3. Back to main menu
```

```
------------------------------------------------------------------------
If select 5. Set Print Mode (Quiet/Verbose)
           Please enter your mode for data
                1. Verbose
                2. Quiet
                3. Back to main menu
------------------------------------------------------------------------
If user input quit, quit the program.
------------------------------------------------------------------------
If users choose Normal mode, program start loop until user type quit
or error is triggered.

------------------------------------------------------------------------
Test Instructions:

           Test File: 0b.txt, 10block.txt, 132kb.txt, 35mb.txt, 512.txt,
65535blocks.txt

           0b.txt: test file transfer with 0 byte. Should be able to
create a local file with size 0.
           10block.txt, 132kb.txt, 512.txt: Normal file transfer. Use to
check block number.
           65535blocks.txt: Use to check extreme cases. When doing a file
transfer, it should stop right at the last
           block, 65535. Display the block number and quit the file
transfer.

           (Timeout 3000ms Up tp 4 times, then disconnect)
               (Error Simulator thread closes if no packet is received in
15 seconds)

Transmission Error:
           Lost a packet:

                RRQ: packet lost no resend, reprompt request from user
after time out

                WRQ: packet lost no resend, reprompt request from user
after time out

                ACK: ACK lost no resend. Wait for DATA packet, disconnect
after timeout

                DATA: resend DATA packet, Wait for ACK packet, disconnect
after timeout

           Delay a packet:

                RRQ: reprompt request from user after time out
```

WRQ: reprompt request from user after time out

ACK (blk# = n): resend DATA packet (blk# = n), send new ACK packet (blk# = n),
receive old ACK packet(blk# = n), send DATA packet (blk# = n+1), receive
new ACK packet (blk# = n), wait for ACK packet (blk# = n+1)

DATA (blk# = n): resend a new DATA packet (blk# = n), receive new DATA packet (blk# = n),
receive old DATA packet (blk# = n), send ACK packet(blk# = n), wait for DATA packet (blk# = n+1)

Duplicate a packet:

RRQ: send DATA packet(blk# = 1),send ACK packet(blk# = 1)

WRQ: send ACK packet(blk# = 0),send DATA packet(blk# = 1)

ACK (blk# = n-1): Ignore duplicate ACK packet (blk# = n-1), send DATA packet (blk# = n-1)

DATA(blk# = n): Ignore duplicate DATA packet, send ACK packet(blk# = n)

Error Code:

To test error code 4, 5, using error simulator to select error, and client choose test mode.

If user want to test error code 1, 2, 3, 6

Error code 1:

RRQ: a. Input a filename does not exist in /src/server/files or the directory user input
b. Input a filename exist in /src/server/files or the directory user input, then using error simulator to change the filename in RRQ packet to a new filename not exist in Server.

WRQ: Input a filename does not exist in /src/client/files or the directory user input

Error code 2:

RRQ: close access permission for the file in /src/server/files or the directory user input

WRQ: close access permission for the file in /src/client/files or the directory user input

Error code 3:

Use a full USB drive to test

Error code 6:

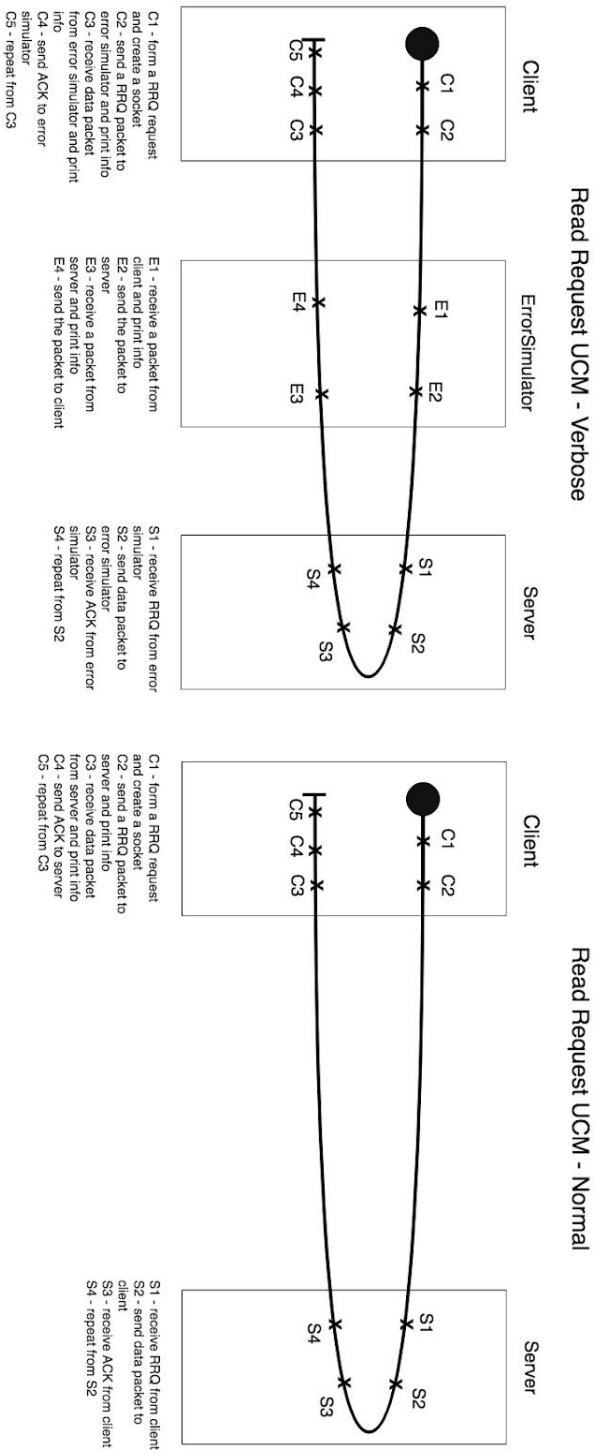
RRQ: Input a filename already exist in /src/client/files or the directory user input

WRQ: a. Input a filename already exist in /src/server/files or the directory user input

b. Input a filename not exist in /src/server/files or the directory user input, then use error simulator to change the filename in WRQ packet to a filename already exist in Server.

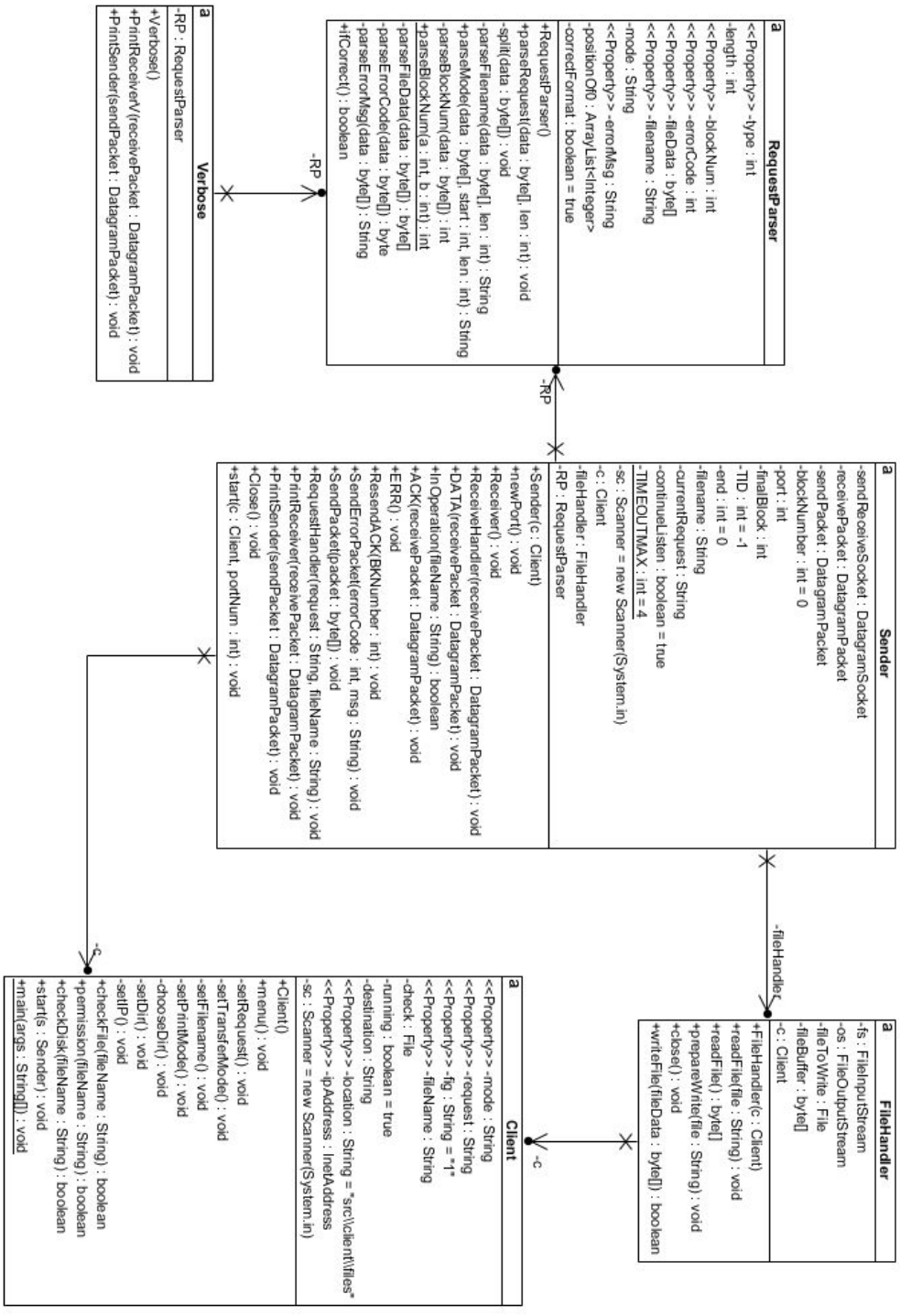
Diagrams

UCMs from iteration 1

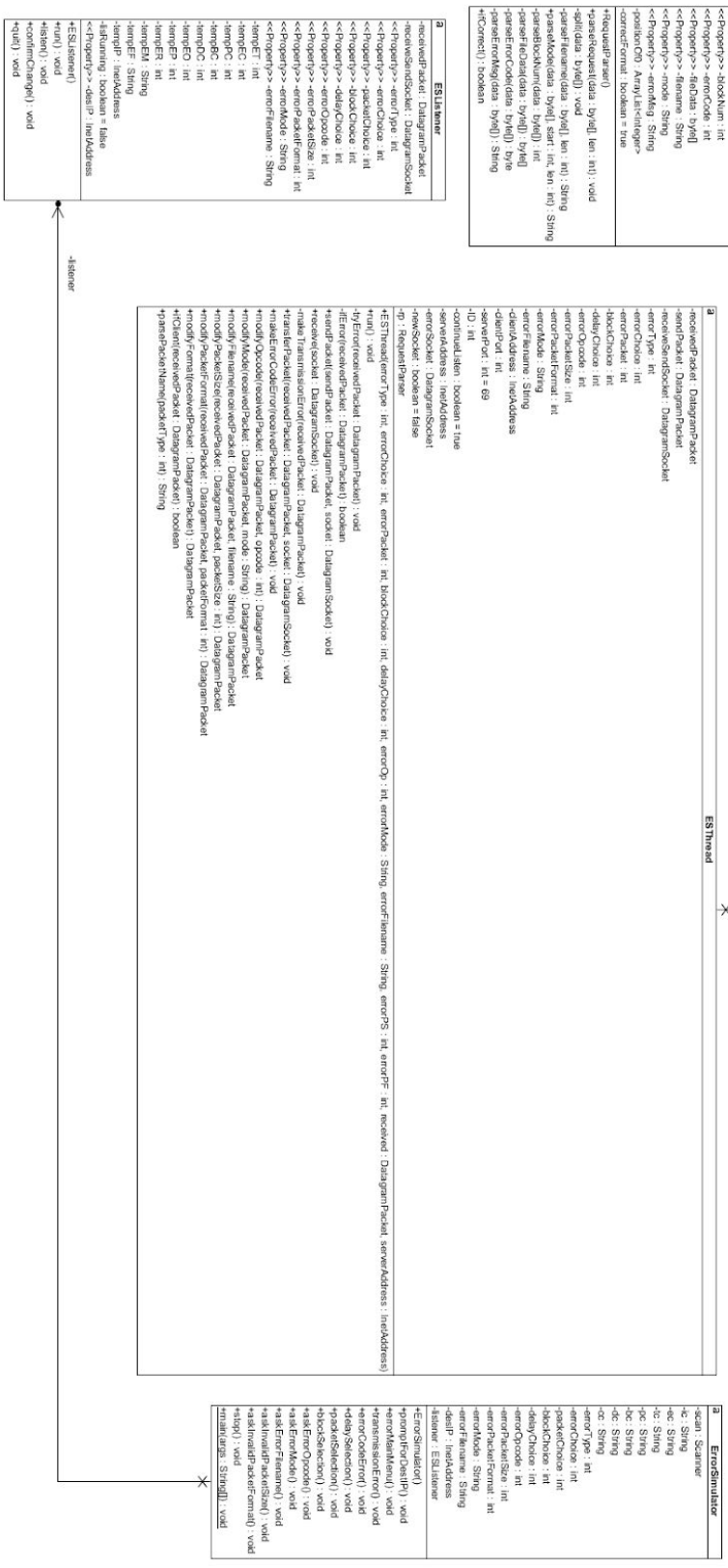


Write Request UCM - Verbose
Client
ErrorSimulator
Server
C1 C2
C5 C4 C3
E1
E2
E4
E3
S1
S2
S4
S3
C1 - form a WRQ request and create a socket
C2 - send a WRQ packet to error simulator and print info
C3 - receive ACK from error simulator and print info
C4 - send data packet to error simulator
C5 - repeat from C3
E1 - receive a packet from client and print info
E2 - send the packet to server
E3 - receive a packet from server and print info
E4 - send the packet to client
S1 - receive WRQ from error simulator and create the file
S2 - send ACK to error simulator
S3 - receive data from error simulator and write to the file
S4 - repeat from S2
Write Request UCM - Normal
Client
Server
C1 C2
C5 C4 C3
S1
S2
S4
S3
C1 - form a WRQ request and create a socket
C2 - send a WRQ packet to server and print info
C3 - receive ACK from server and print info
C4 - send data packet to server
C5 - repeat from C3
S1 - receive WRQ from client and create the file
S2 - send ACK to client
S3 - receive data from client and write to the file
S4 - repeat from S2

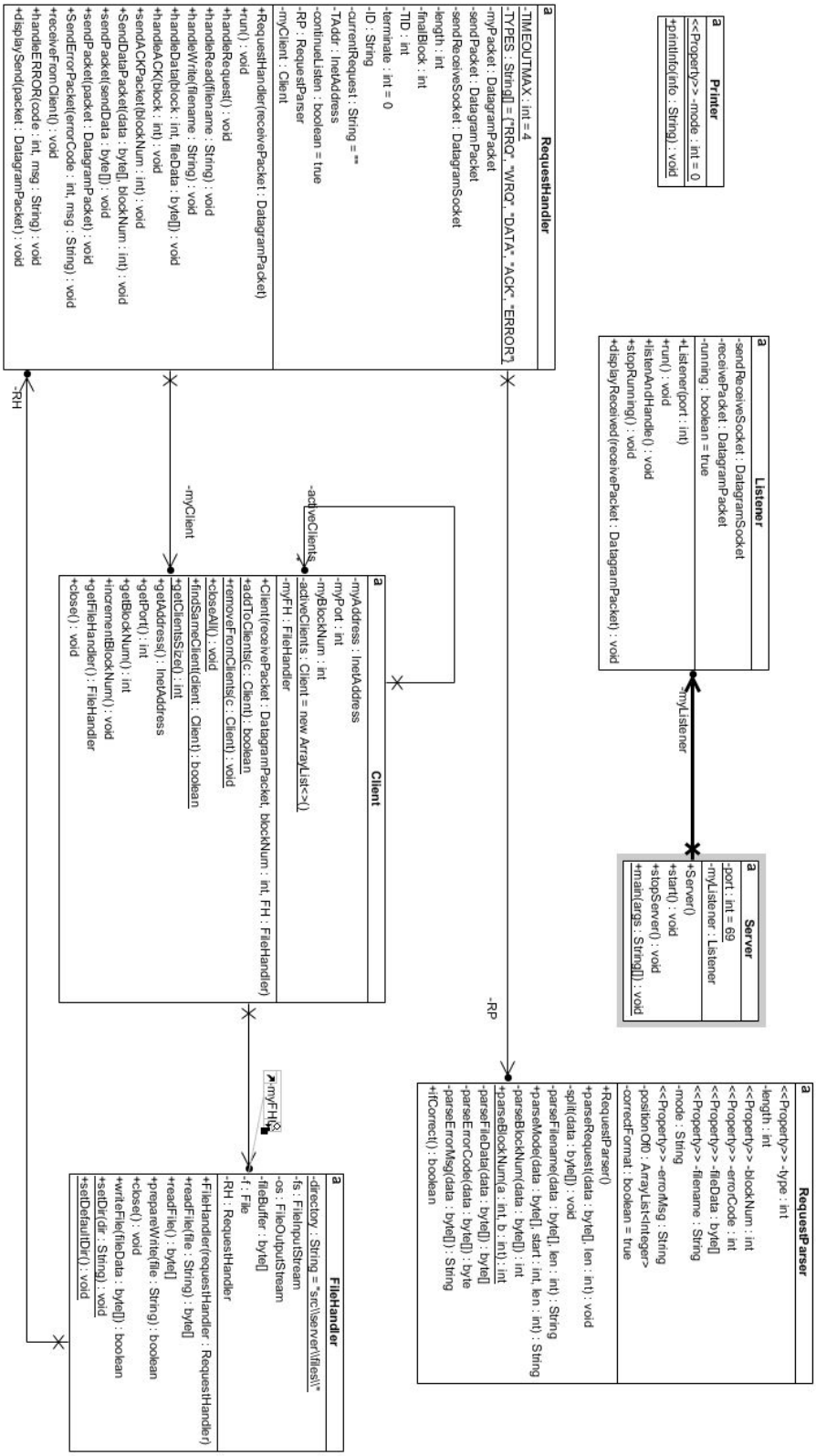
UML diagrams



Client UML

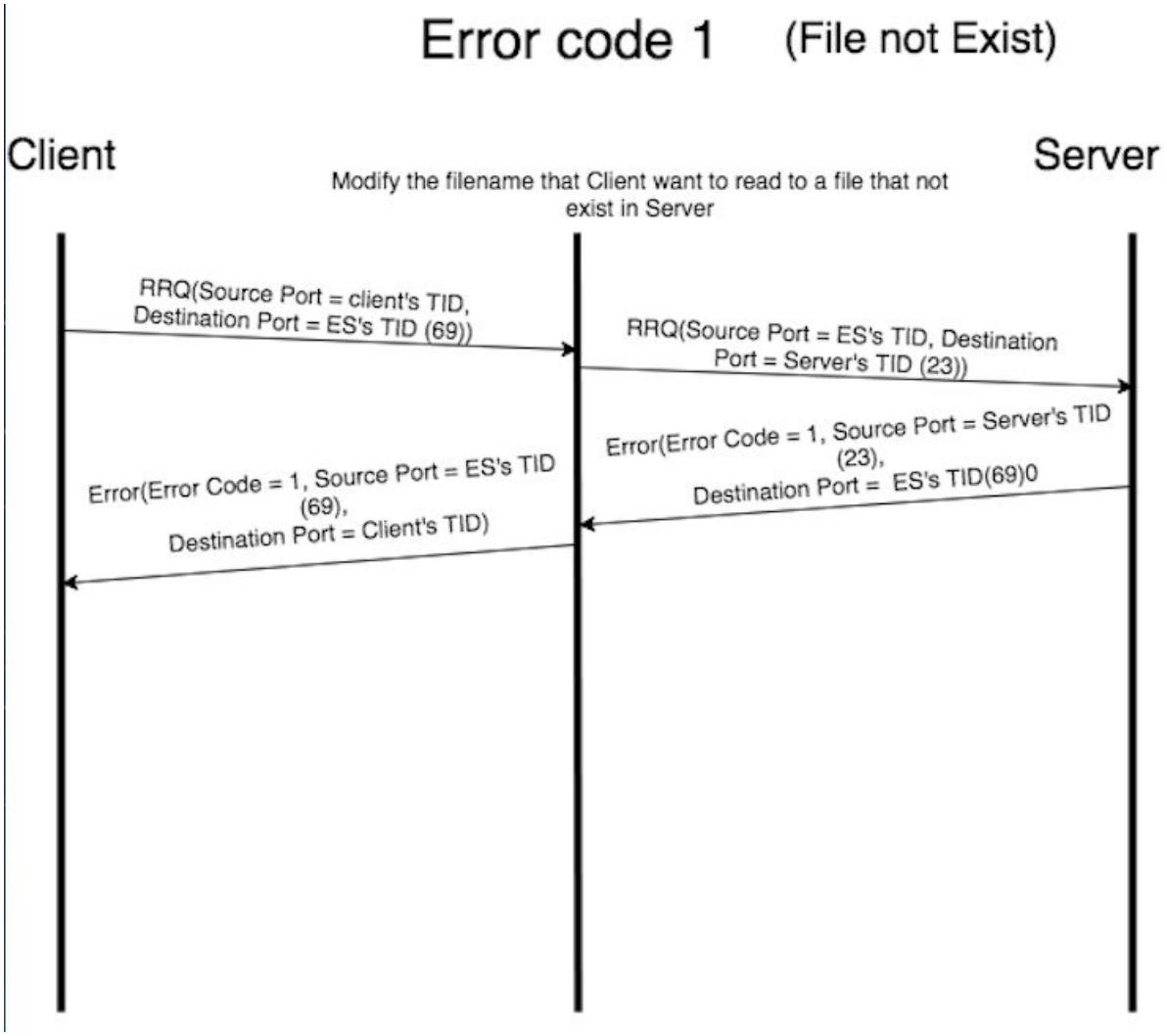


Error Simulator UML

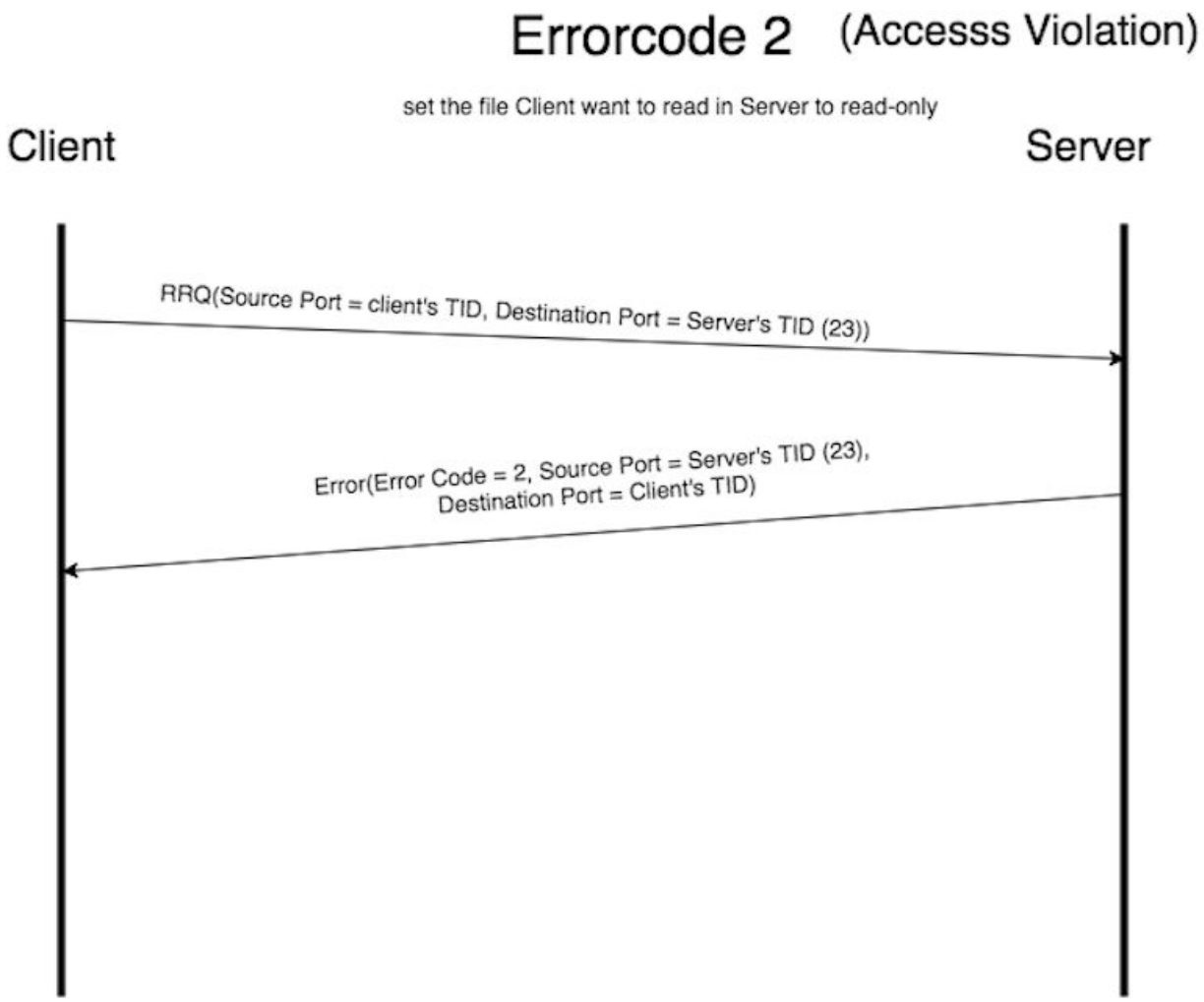

Printer
<<Property>> -mode : int = 0
+printInfo(info : String) : void
Listener
-sendReceiveSocket : DatagramSocket
-receivePacket : DatagramPacket
-running : boolean = true
+Listener(port : int)
+run() : void
+listenAndHandle() : void
+stopRunning() : void
+displayReceived(receivePacket : DatagramPacket) : void
-myListener
Server
-port : int = 69
-myListener : Listener
+Server()
+start() : void
+stopServer() : void
+main(args : String[]) : void
RequestHandler
-TIMEOUTMAX : int = 4
-TYPES : String[] = {"RRQ", "WRQ", "DATA", "ACK", "ERROR"}
-myPacket : DatagramPacket
-sendPacket : DatagramPacket
-sendReceiveSocket : DatagramSocket
-length : int
-finalBlock : int
-TID : int
-terminate : int = 0
-ID : String
-currentRequest : String = ""
-TAddr : InetAddress
-continueListen : boolean = true
-RP : RequestParser
-myClient : Client
+RequestHandler(receivePacket : DatagramPacket)
+run() : void
+handleRequest() : void
+handleRead(filename : String) : void
+handleWrite(filename : String) : void
+handleData(block : int, fileData : byte[]) : void
+handleACK(block : int) : void
+sendACKPacket(blockNum : int) : void
+SendDataPacket(data : byte[], blockNum : int) : void
+sendPacket(sendData : byte[]) : void
+sendPacket(packet : DatagramPacket) : void
+SendErrorPacket(errorCode : int, msg : String) : void
+receiveFromClient() : void
+handleERROR(code : int, msg : String) : void
+displaySend(packet : DatagramPacket) : void
-RH
-RP
RequestParser
<<Property>> -type : int
-length : int
<<Property>> -blockNum : int
<<Property>> -errorCode : int
<<Property>> -fileData : byte[]
<<Property>> -filename : String
-mode : String
<<Property>> -errorMsg : String
-positionOf0 : ArrayList<Integer>
-correctFormat : boolean = true
+RequestParser()
+parseRequest(data : byte[], len : int) : void
-split(data : byte[]) : void
-parseFilename(data : byte[], len : int) : String
+parseMode(data : byte[], start : int, len : int) : String
-parseBlockNum(data : byte[]) : int
+parseBlockNum(a : int, b : int) : int
-parseFileData(data : byte[]) : byte[]
-parseErrorCode(data : byte[]) : byte
-parseErrorMsg(data : byte[]) : byte
+ifCorrect() : boolean
Client
-myAddress : InetAddress
-myPort : int
-myBlockNum : int
-activeClients : Client = new ArrayList<>()
-myFH : FileHandler
+Client(receivePacket : DatagramPacket, blockNum : int, FH : FileHandler)
+addToClients(c : Client) : boolean
+removeFromClients(c : Client) : void
+closeAll() : void
+findSameClient(client : Client) : boolean
+getClientsSize() : int
+getAddress() : InetAddress
+getPort() : int
+getBlockNum() : int
+incrementBlockNum() : void
+getFileHandler() : FileHandler
+close() : void
-activeClients
-myClient
myFH
FileHandler
-directory : String = "src\\server\\files\\"
-fs : FileInputStream
-os : FileOutputStream
-fileBuffer : byte[]
-f : File
-RH : RequestHandler
+FileHandler(requestHandler : RequestHandler)
+readFile(file : String) : byte[]
+readFile() : byte[]
+prepareWrite(file : String) : boolean
+close() : void
+writeFile(fileData : byte[]) : boolean
+setDir(dir : String) : void
+setDefaultDir() : void

Server UML

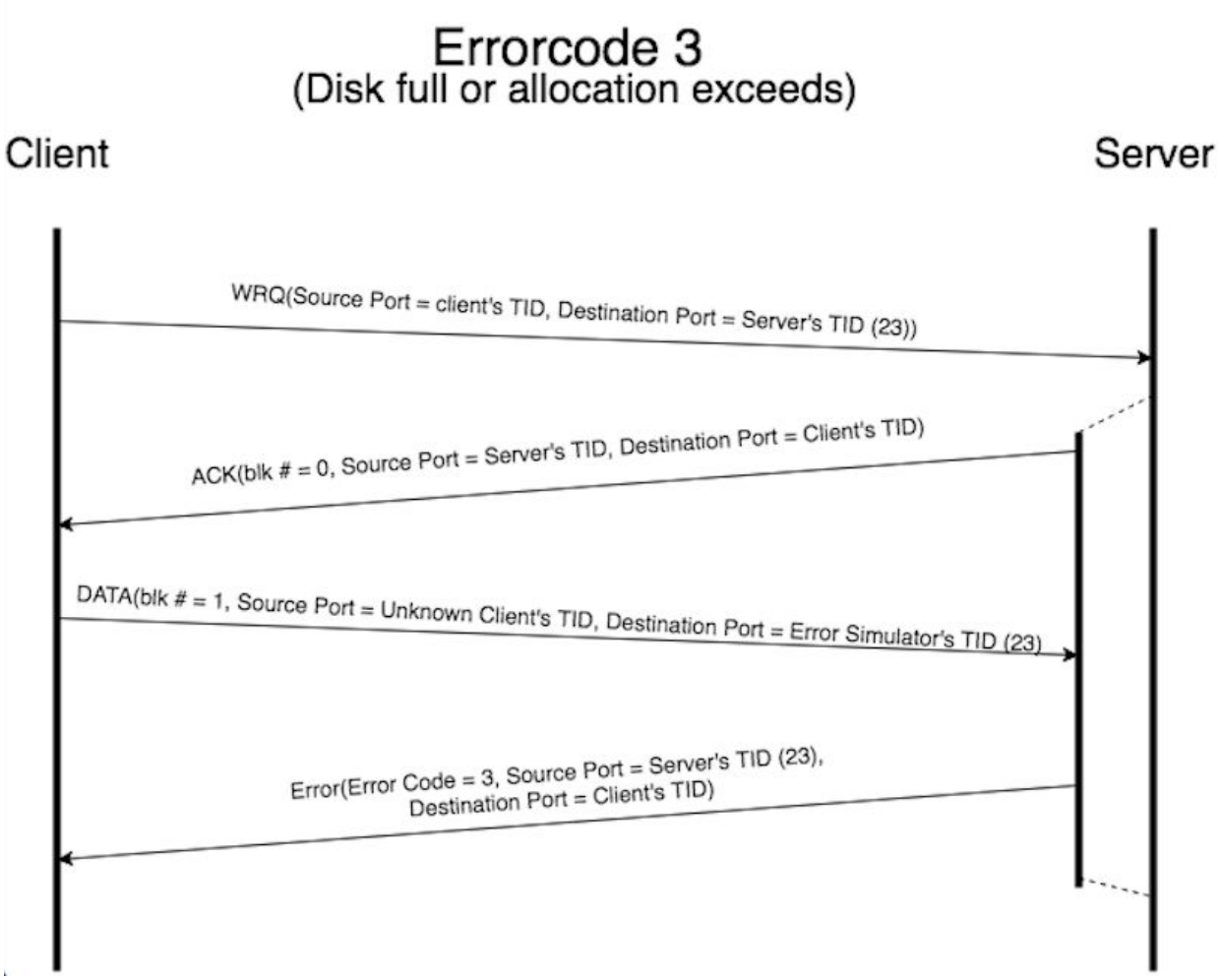
Timing diagrams



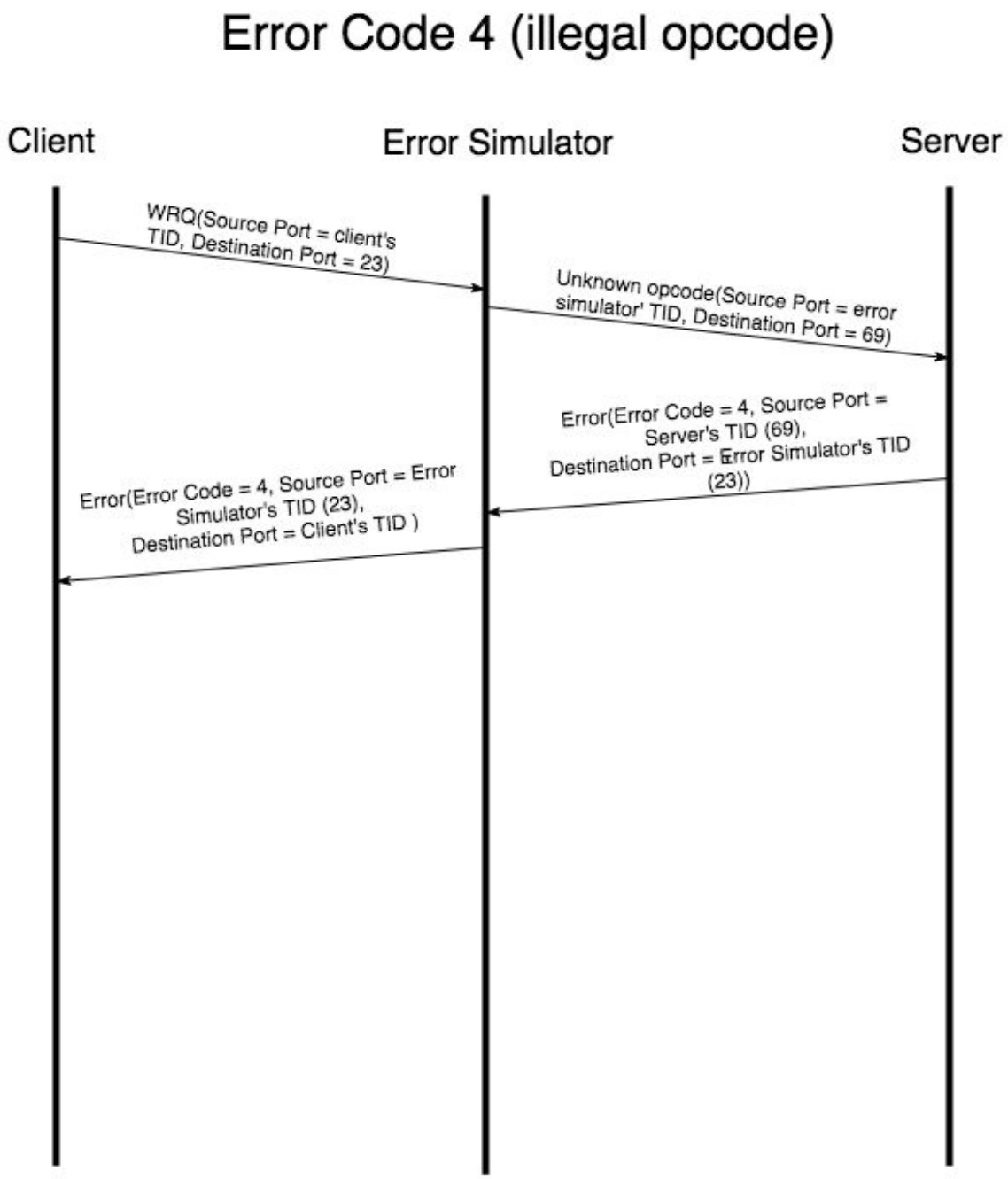
Timing diagram (Error Code 1)

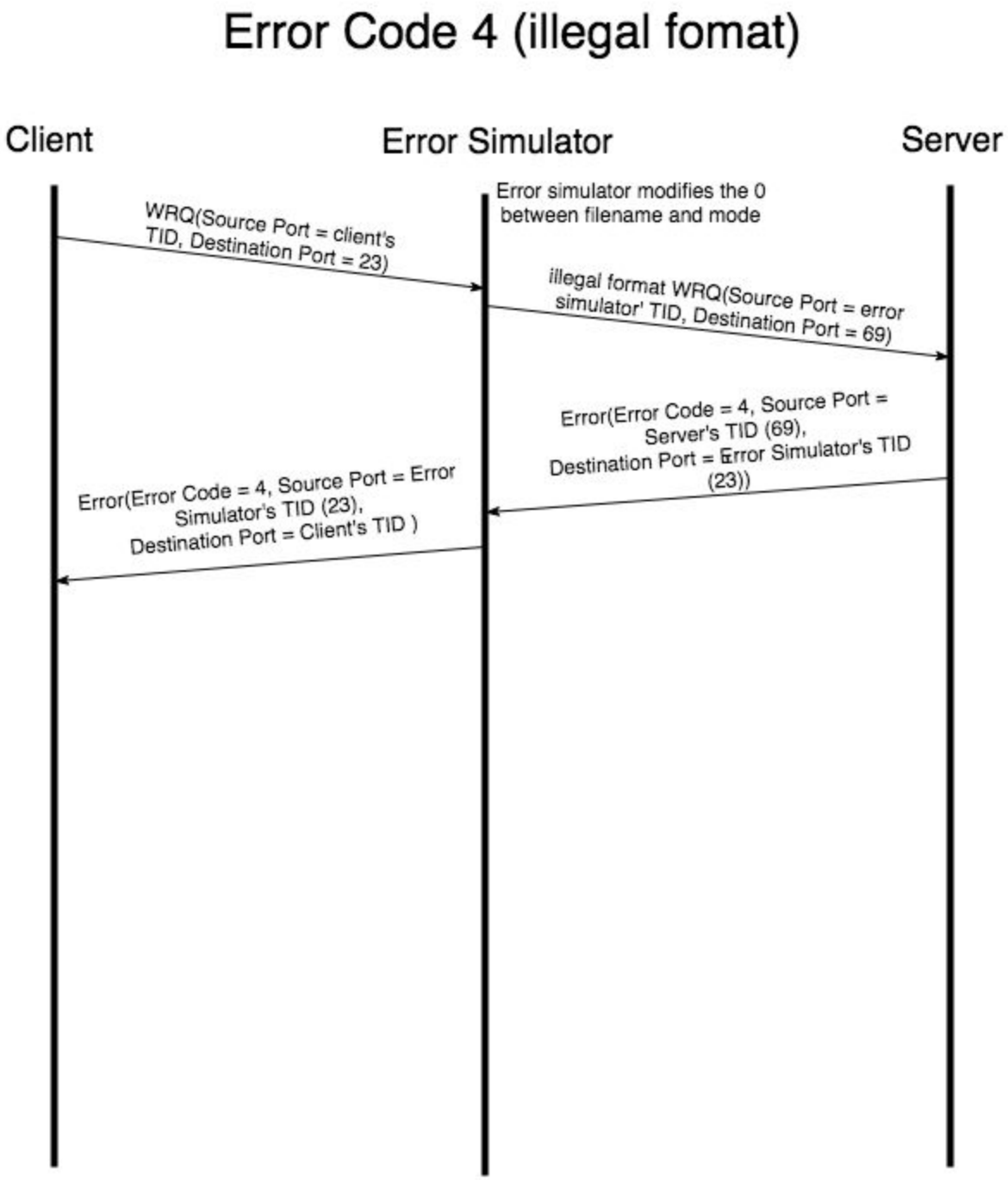


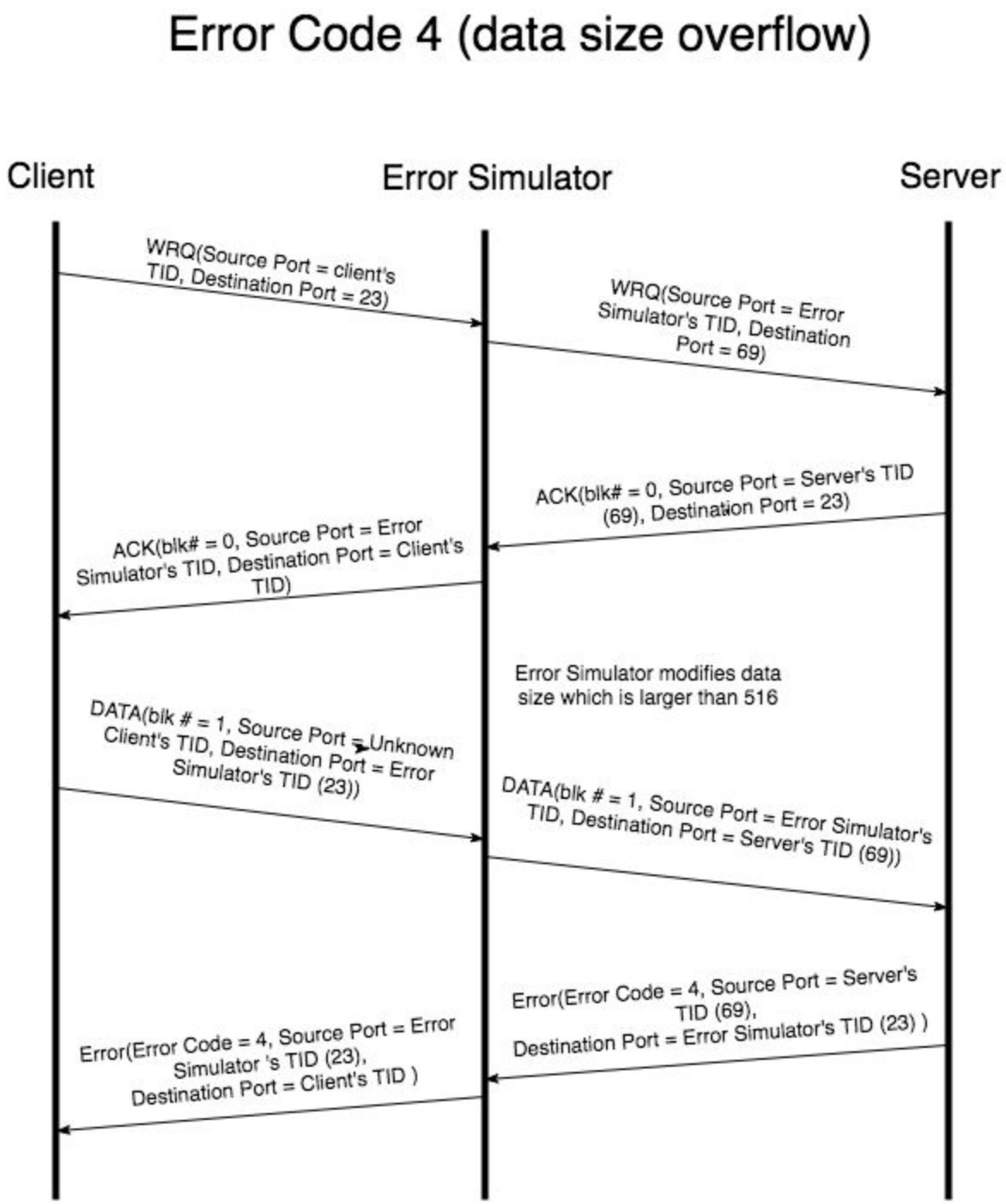
Timing diagram (Error Code 2)



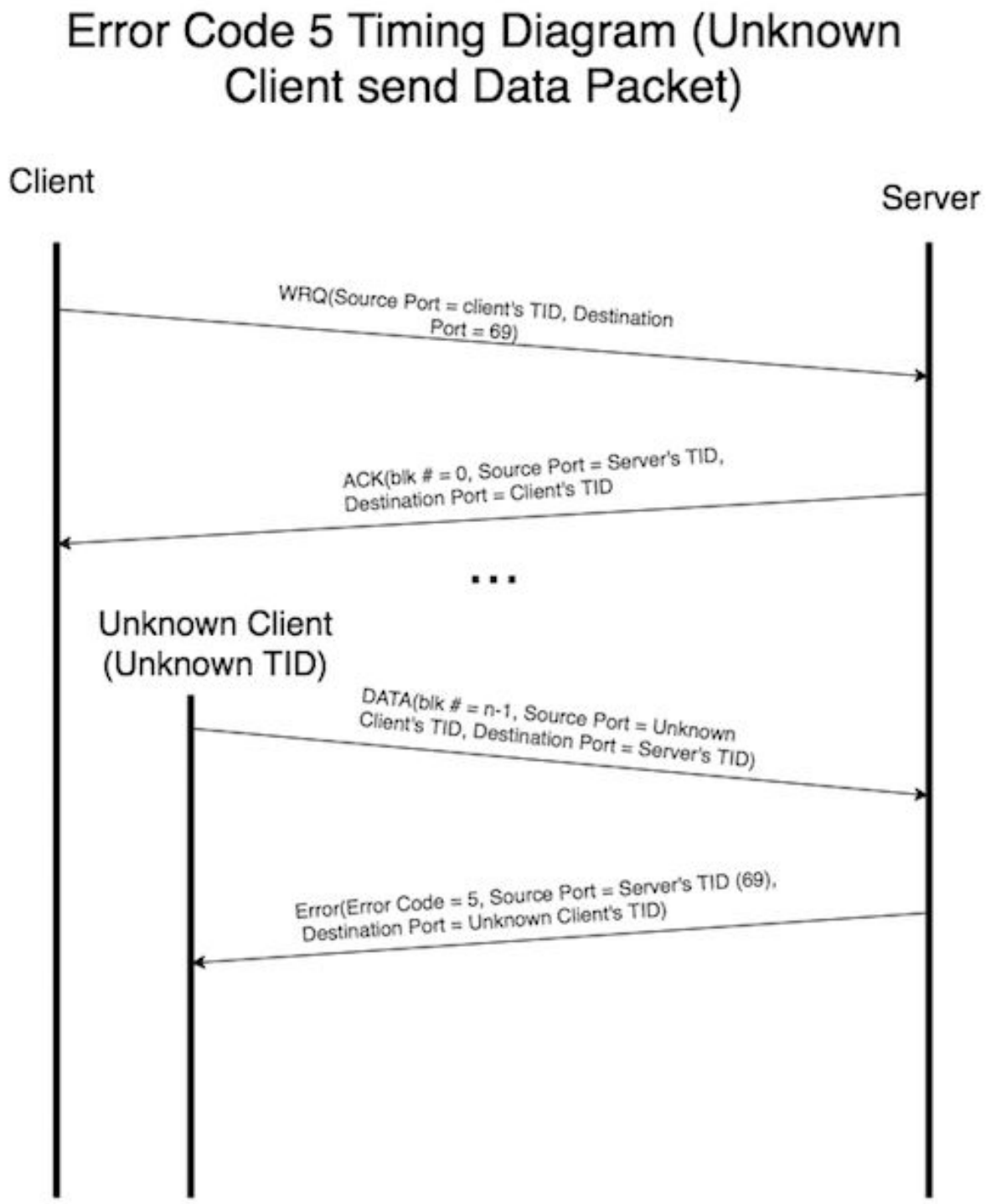
Timing diagram (Error Code 3)

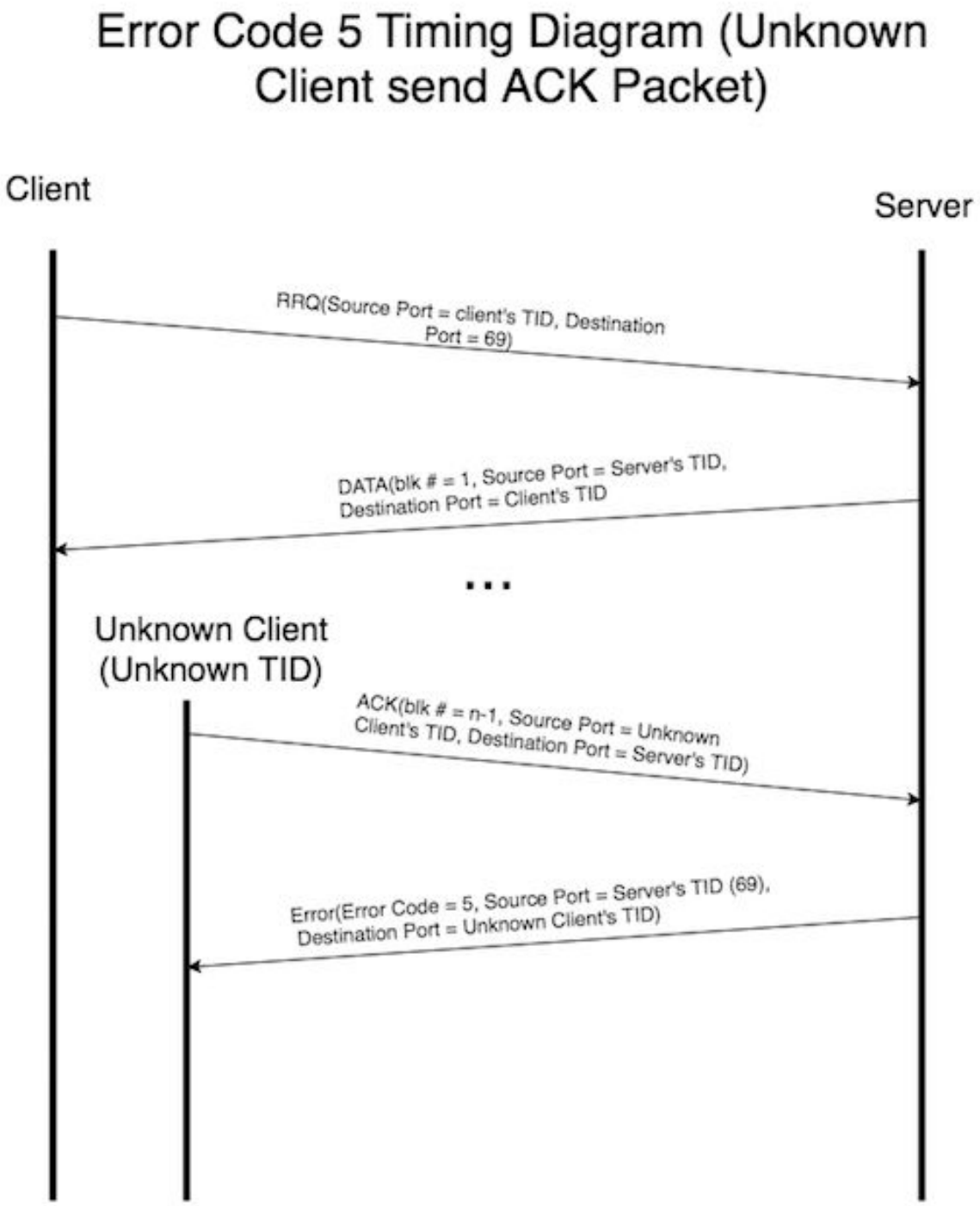

Error Code 4 (illegal opcode)
Client
Error Simulator
Server
WRQ(Source Port = client's TID, Destination Port = 23)
Unknown opcode(Source Port = error simulator' TID, Destination Port = 69)
Error(Error Code = 4, Source Port = Server's TID (69), Destination Port = Error Simulator's TID (23))
Error(Error Code = 4, Source Port = Error Simulator's TID (23), Destination Port = Client's TID)


Error Code 4 (illegal fomat)
Client
Error Simulator
Server
WRQ(Source Port = client's TID, Destination Port = 23)
Error simulator modifies the 0 between filename and mode
illegal format WRQ(Source Port = error simulator' TID, Destination Port = 69)
Error(Error Code = 4, Source Port = Server's TID (69), Destination Port = Error Simulator's TID (23))
Error(Error Code = 4, Source Port = Error Simulator's TID (23), Destination Port = Client's TID)

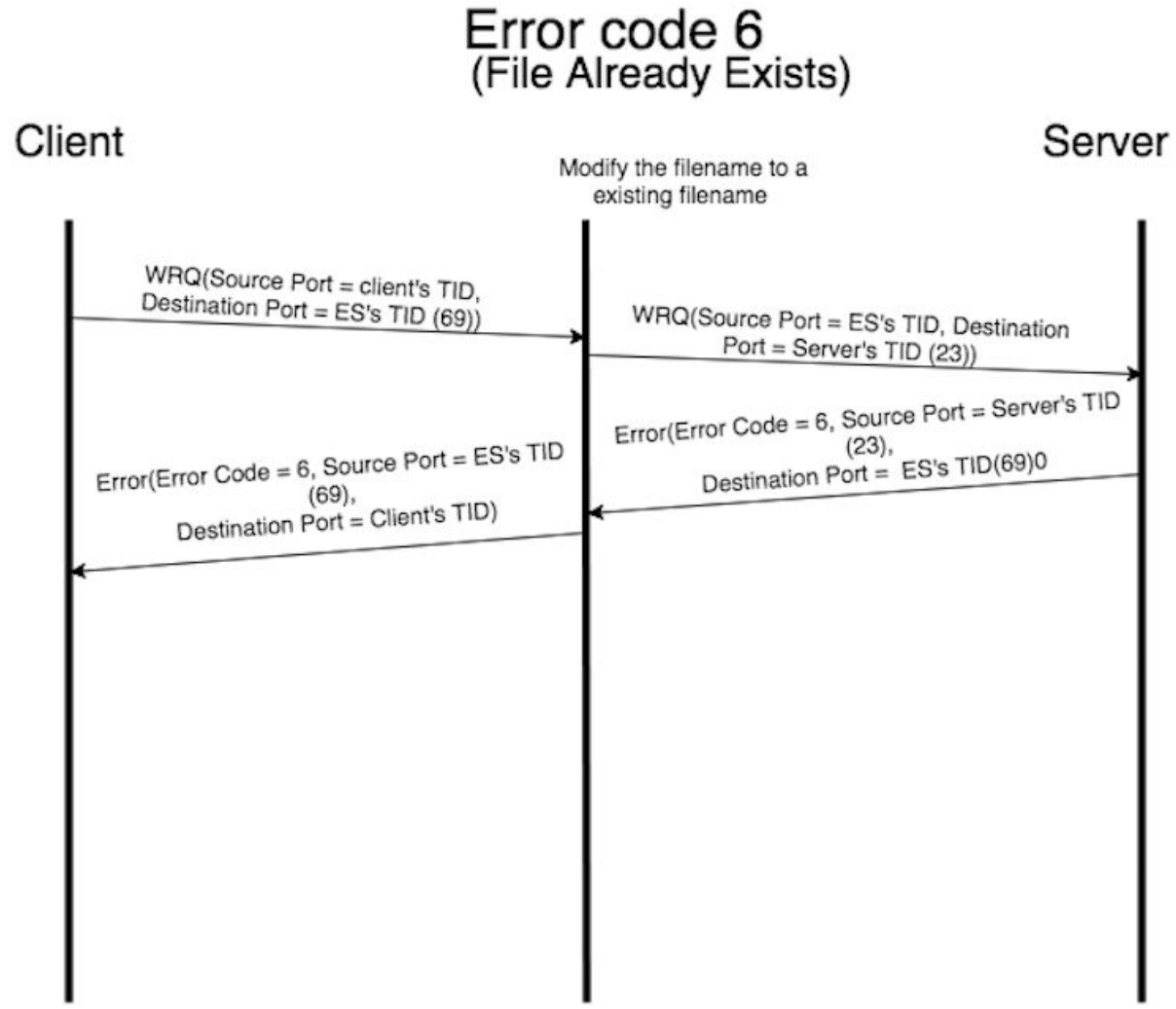


Timing diagram (Error Code 4)


Error Code 5 Timing Diagram (Unknown Client send Data Packet)
Client
Server
WRQ(Source Port = client's TID, Destination Port = 69)
ACK(blk # = 0, Source Port = Server's TID, Destination Port = Client's TID
Unknown Client (Unknown TID)
DATA(blk # = n-1, Source Port = Unknown Client's TID, Destination Port = Server's TID)
Error(Error Code = 5, Source Port = Server's TID (69), Destination Port = Unknown Client's TID)



Timing diagram (Error Code 5)



Timing diagram (Error Code 6)