TFTP Project - Team 2000000

SYSC 3303 – A

**Members:**

Supriya Gadigone

Kshamina Ghelani

Daniel Graval

Bhavik Tailor

Tanzim Zaman

Contents

[Breakdown of Responsibilities 3](#_Toc468794494)

[UCM Diagrams 6](#_Toc468794495)

[UML Class Diagram 8](#_Toc468794496)

[Timing Diagrams 9](#_Toc468794497)

[Testing Instructions & Files 30](#_Toc468794498)

[Instructions 31](#_Toc468794499)

[When running in normal mode: 31](#_Toc468794500)

[When running in test mode: 31](#_Toc468794501)

[General 31](#_Toc468794502)

[Testing Errors 32](#_Toc468794503)

[Disk Full 32](#_Toc468794504)

[Client 32](#_Toc468794505)

[Server 32](#_Toc468794506)

[File Already Exists 32](#_Toc468794507)

[Client 32](#_Toc468794508)

[Server 32](#_Toc468794511)

[File Not Found 32](#_Toc468794515)

[Client 32](#_Toc468794516)

[Server 32](#_Toc468794519)

[File Access Denied 32](#_Toc468794522)

[Illegal TFTP 33](#_Toc468794523)

[Unknown Transfer ID 33](#_Toc468794524)

[Intermediate Host – Normal, Lost, Delay, Duplicate 33](#_Toc468794527)

[Intermediate Host – Manipulating Error Packet 33](#_Toc468794528)

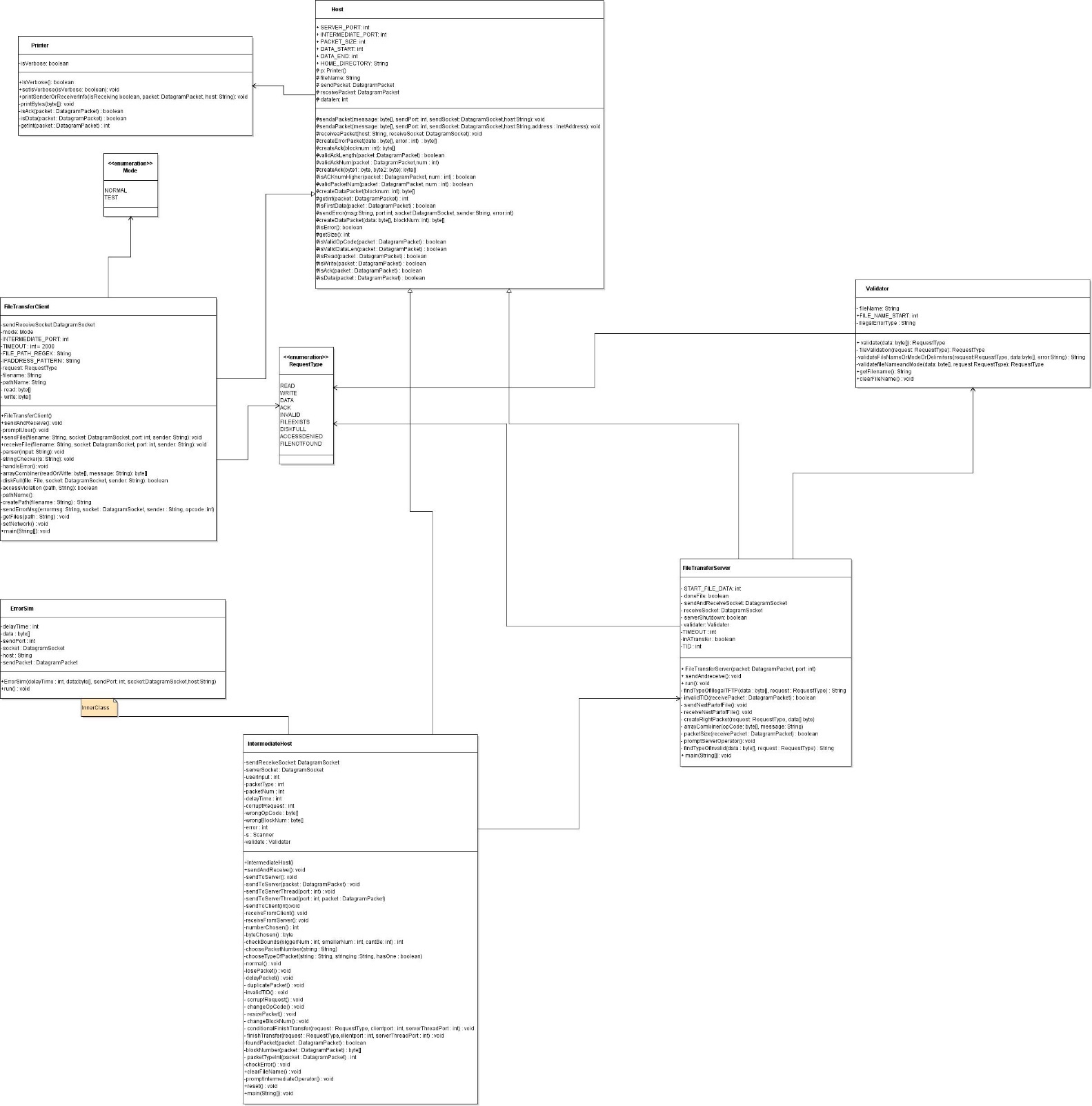
# Breakdown of Responsibilities

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| I# | Group Member | | | | |
| Supriya Gadigone | Kshamina Ghelani | Daniel Gravel | Bhavik Tailor | Tanzim Zaman |
| 1 | Made JavaDocs comments for all methods  Adjusted methods within Printer class  Helped debug FileTransferClient | UML Class Diagrams  UCM Diagrams  README.md | FileTransferClient.java  Host.java  receiveFile  sendFile  createAck  createDataPacket  Printer  Verbose/Quiet | FileTranserServer.java  Made method to receive file data from client via intermediate  Adjusted IntermediateHost to accommodate larger file transfers  Shutdown in client | FileTransferServer.java  Receiving files  Sending files  Validation and what to do with requests/data/acks  Host.java  Debugged  FileTransferClient.java  Debugged |
| 2 | Prompt for file path in client  Client timeout  UML Class Diagram  General debugging and testing (Client Side) | FileTransferServer.java  error handling, server side  server shutdown  debugging/testing  Printer.java  Worked on fixing bug in verbose mode (Server side)  timing diagrams | Testing different errors  FileTransferClient.java  receiveFile  sendFile  user prompt  server error handling  client error handling | Made methods for handling errors in client class  Adjusted UCM Diagrams  Debugging client side parser  General Debugging | Decoupled server class, moved validation to own class: Validater.java  Moved RequestType to own class  FileTransferServer.java  Debugged  Updated all aspects with regards to feedback an Iteration 2  Host.java  Debugged, added classes |
| 3 | Timing diagrams  Handled duplicate packets | UML Class Diagram  Fixed block numbers from client for ack/data  IntermediateHost.java  Debugged  Wrote prompt for different network errors  Implemented losing and delaying packets | Added ability to change directories  Added various new commands(ls, pwd)  Debugged IntermediateHost, client, and Host  Helped implement lost packets client side | Handled lost packets on server side  Handled multiple ACKs on server | Aided in implementation of duplicating, delaying, losing packets  Added changing op code feature |
| 4 | Implemented server side TID and invalid tftp  Debugged server side issues | Timing Diagrams  Added error simulations for invalid TID, corrupt request, invalid packet size, change block number  Debugged Intermediate Host and Host | Added client side error handling for TID,invalid TFTP  Helped debug host, intermediate host | Updated UML Diagrams  Updated ReadME  Helped debug server side issues | Added error simulation: alter opcode  Debugged Intermediate Host  Tested Error Sim scenarios |
| 5 | Debugged and tested on server side  Fixed illegal TFTP error on server side  Filename  Mode  Delimiter | Fixed IT4 issues in Intermediate Host  Fixed sending and receiving packets using correct length  Debugged and tested IH and Server  Put together final hard copy report | Fixed IT4 issues in client and Host  Debugged and tested  Implemented sending across a network  Diagrams  Fixed server shutdown | * Timing Diagrams * Debugged Server & Testing   Helped with invalid block number | Implemented quiet mode  Tested all cases  Fixed corrupting packet/opcode bugs  Added and aided error implementations (losing, delaying, etc)  Prompt fixed to also accept errors |

# UCM Diagrams

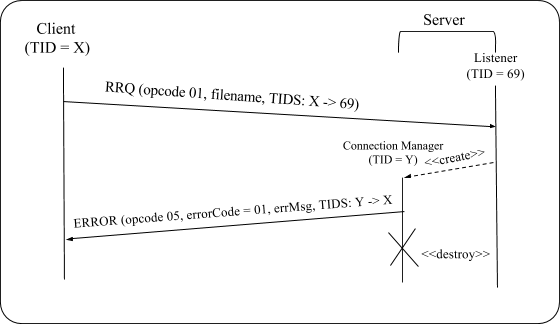


# UML Class Diagram

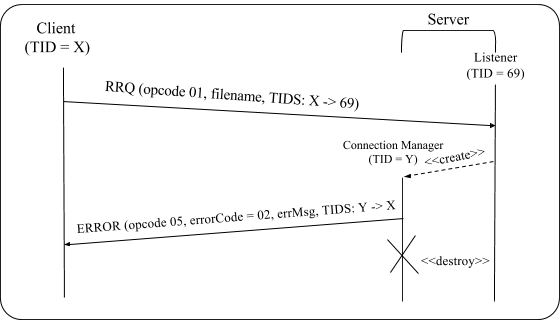


# Timing Diagrams

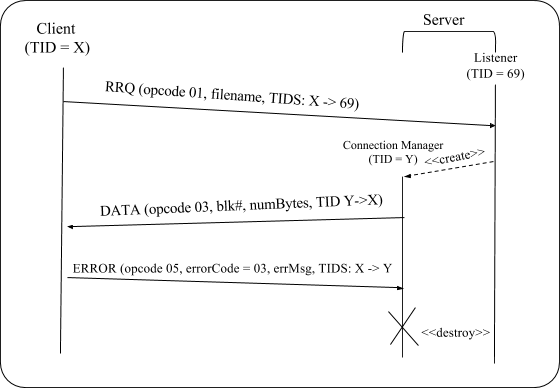
Timing Diagram, Error 1 - File Not Found



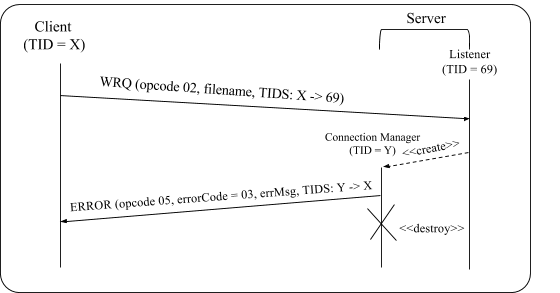
Timing Diagram, Error 2 - Access Violation



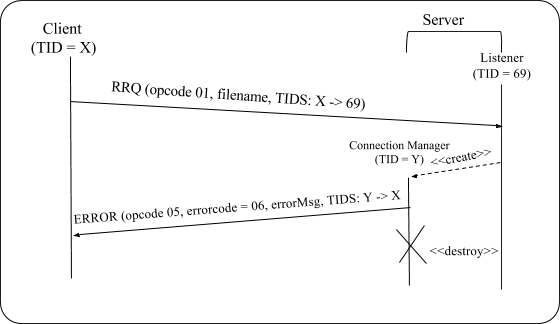
Timing Diagram, Error 3 - Disk Full, Case 1



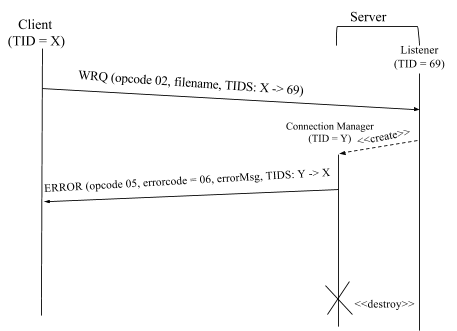
Timing Diagram, Error 3 - Disk Full, Case 2



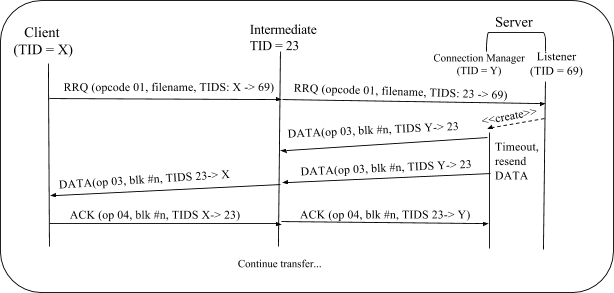
Timing Diagram, Error 6 - File Already Exists, RRQ



Timing Diagram, Error 6 - File Already Exists,WRQ



Timing Diagram, Lost Data

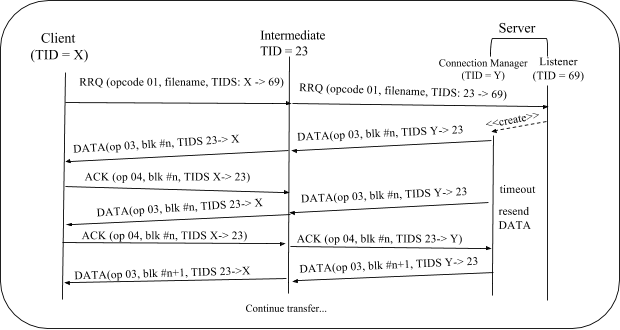


Timing Diagram, RRQ, Lose Request Packet

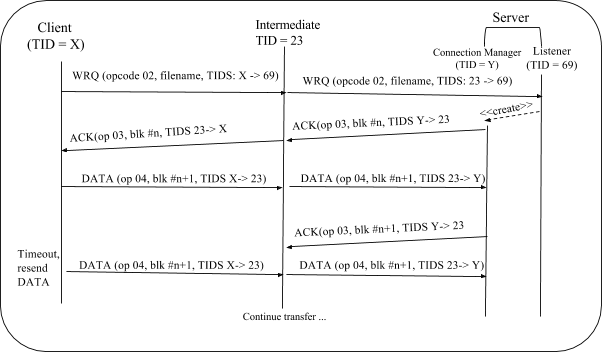


\*\* client will timeout and re prompt user for a new request (same for WRQ)

Timing Diagram, RRQ, Lose ACK Packet

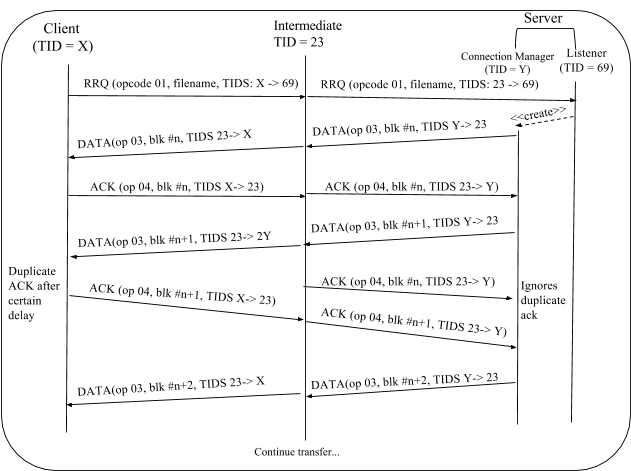


Timing Diagram, WRQ, Lose ACK Packet

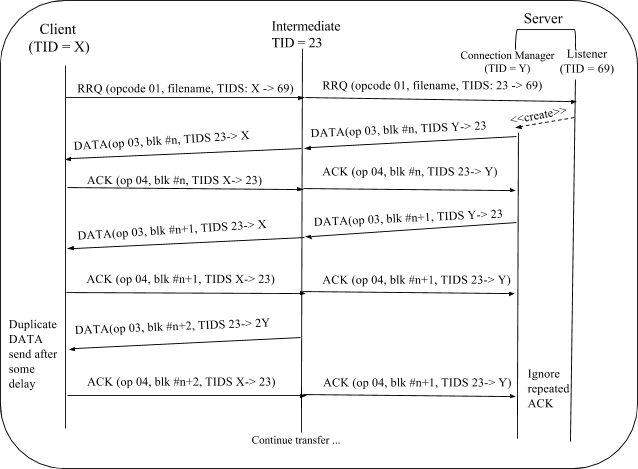


Timing Diagram, WRQ, Lose DATA Packet

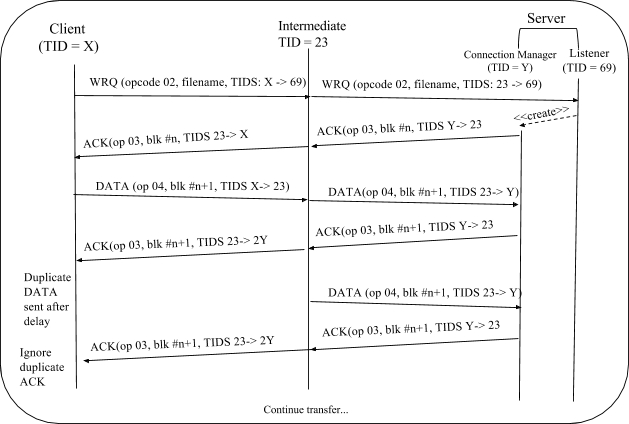


Timing Diagram, RRQ, Duplicate ACK Packet

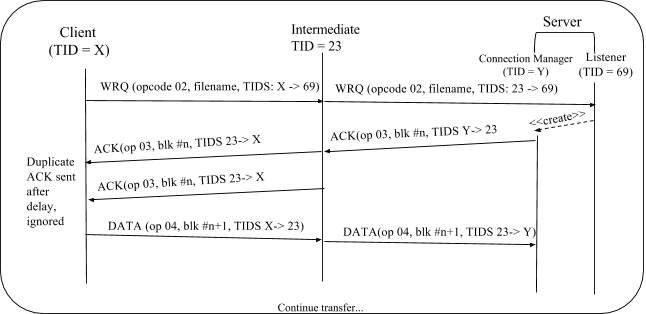
Timing Diagram, RRQ, Duplicate DATA Packet



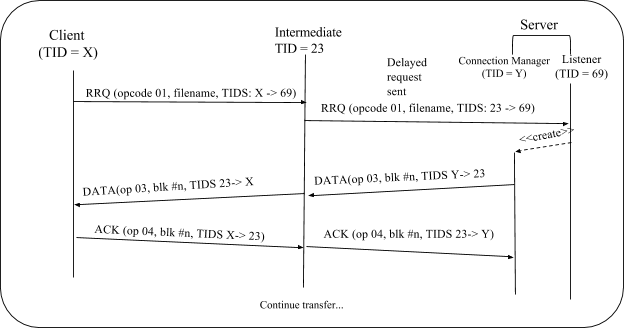
Timing Diagram, WRQ, Duplicate DATA Packet



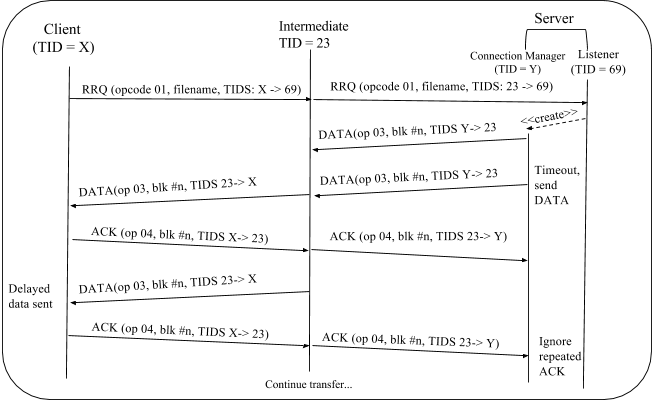
Timing Diagram, WRQ, Duplicate ACK

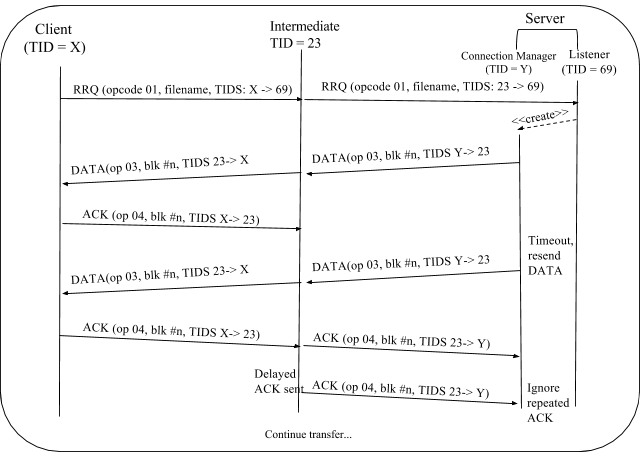


Timing Diagram, Delay Request



\*\* if client times out, client will reprompt the user for a new request (same for WRQ)

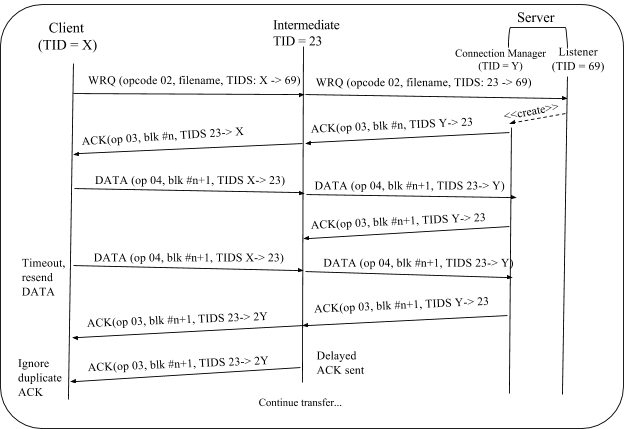
Timing Diagram, RRQ, Delay DATA  


Timing Diagram, RRQ, Delay ACK  
  


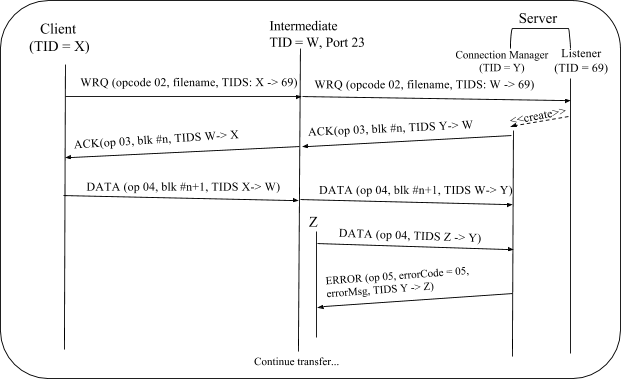
Timing Diagram, WRQ, Delay DATA



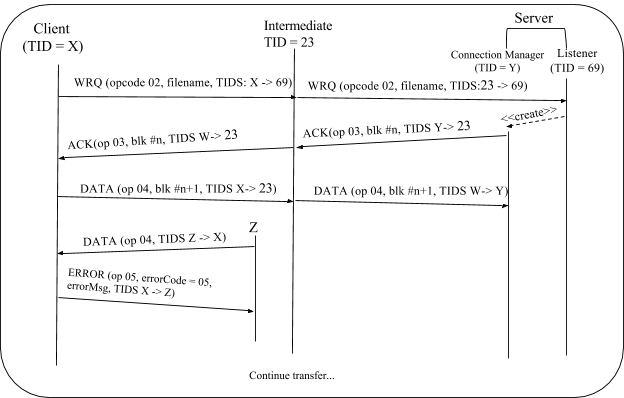
Timing Diagram, WRQ, Delay ACK

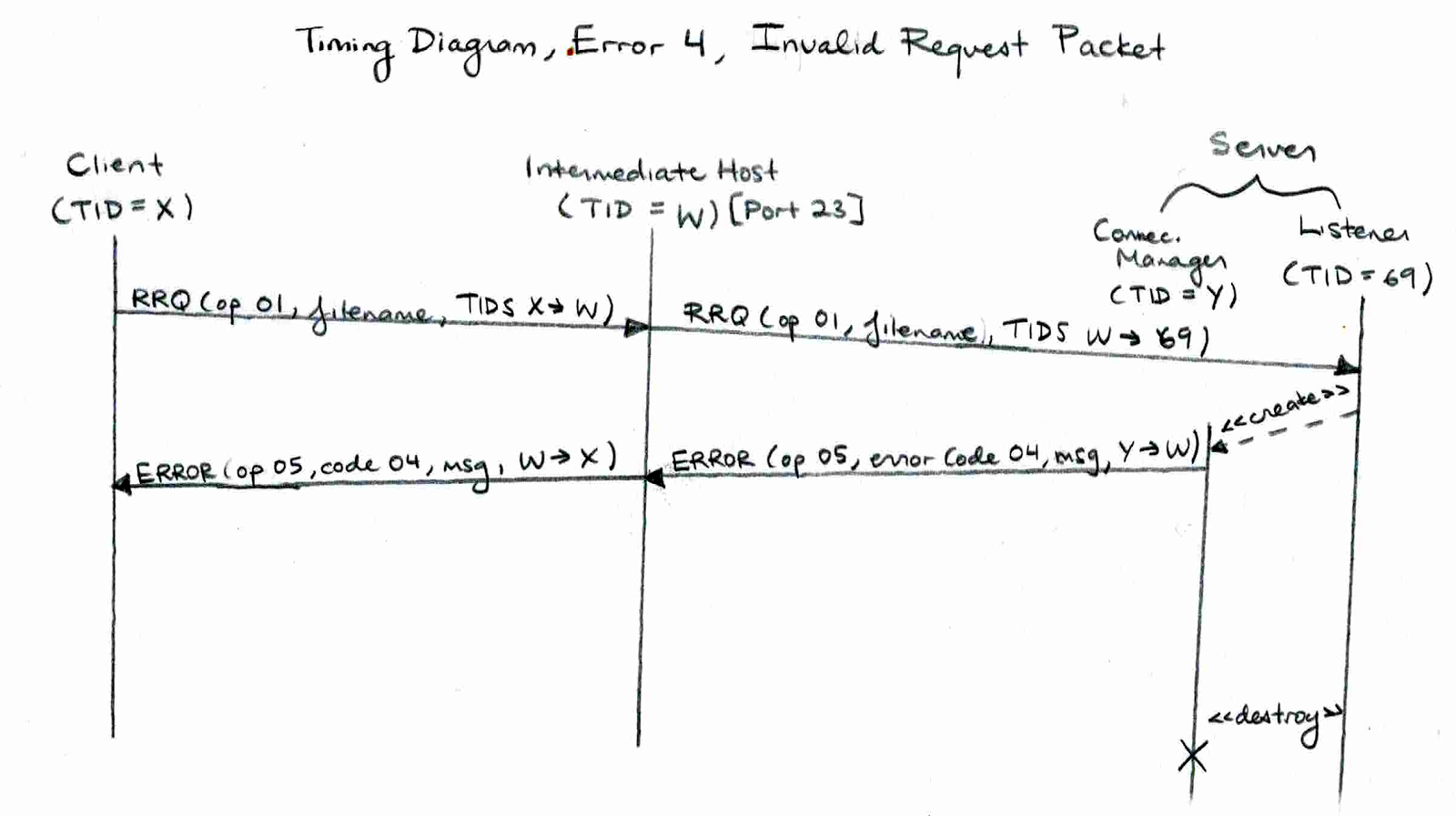


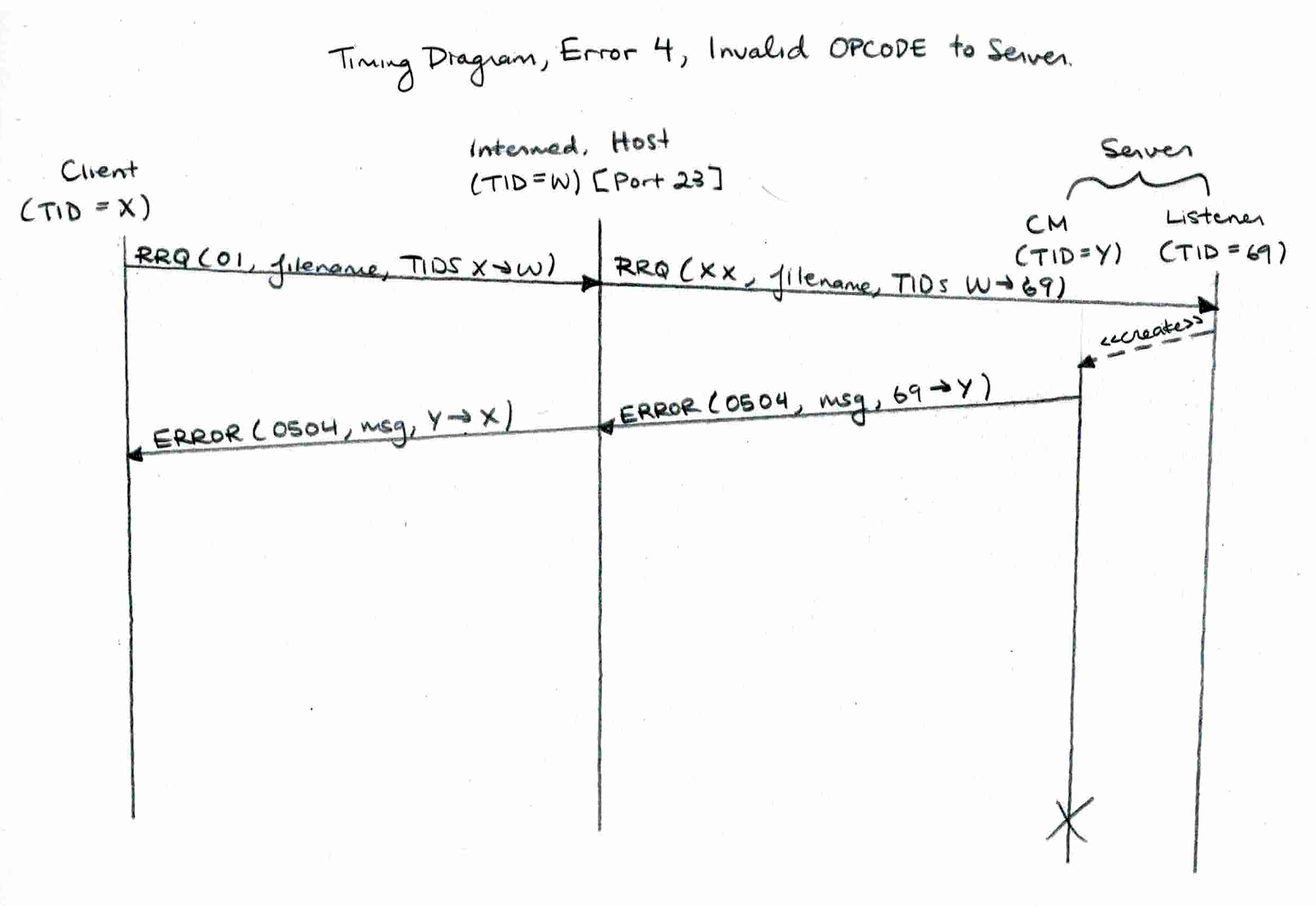
Timing Diagram, Error 5 - Invalid  TID, Sent to Client

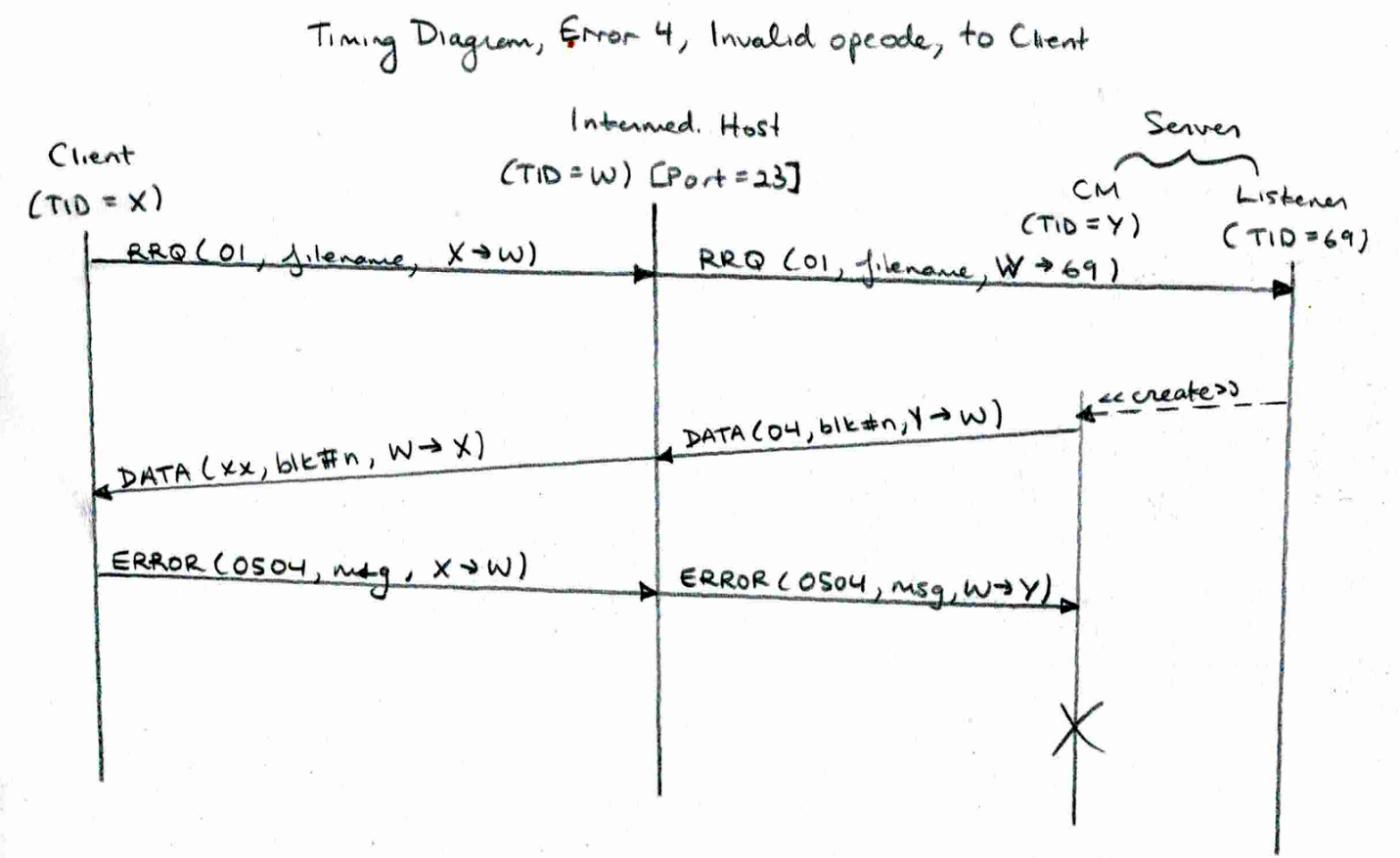


Timing Diagram, Error 5 - Invalid TID, Sent to Server



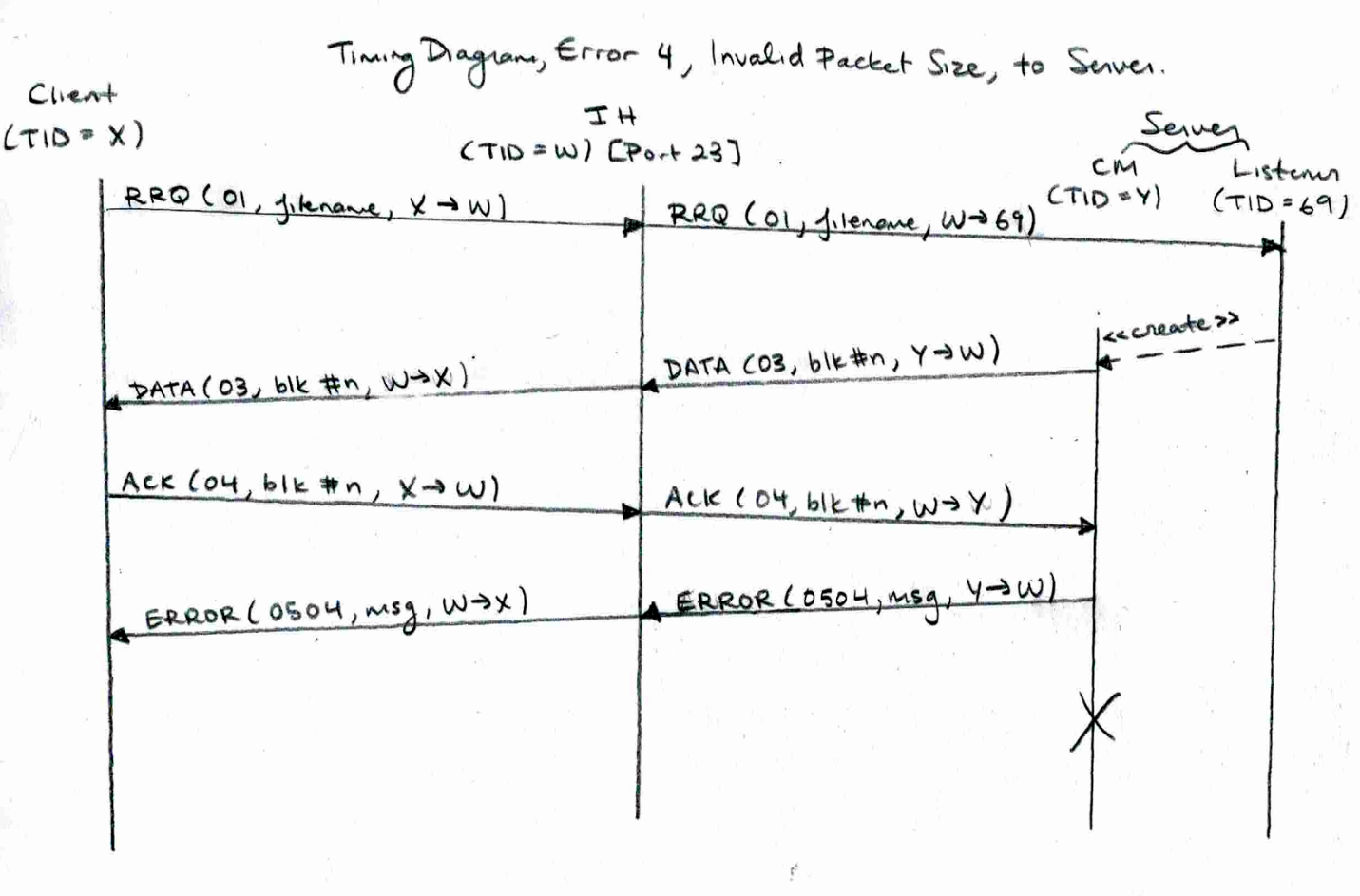




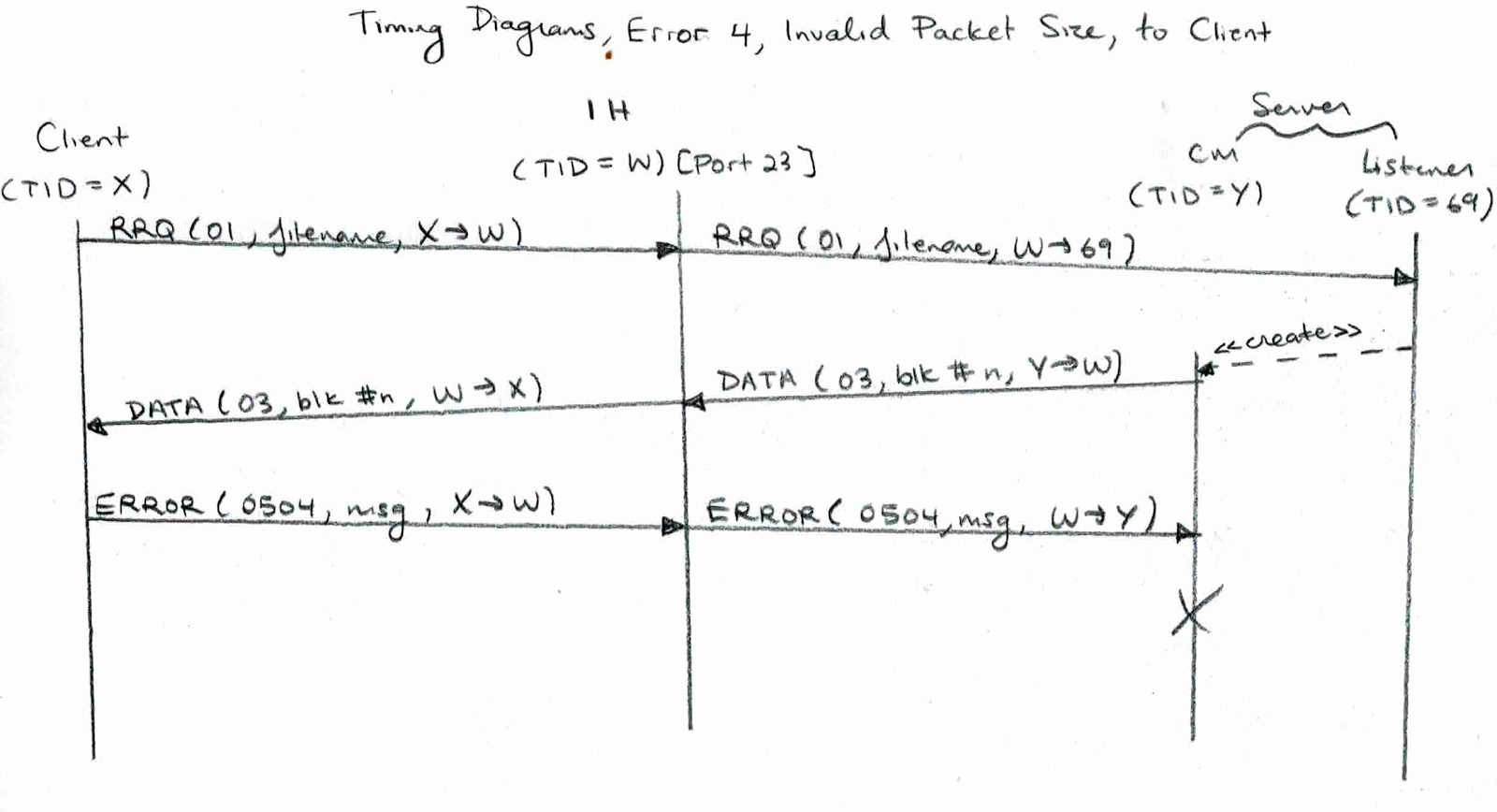


\*\* if RRQ changed to WRQ, there will be a file already exists error

\*\* WRQ to READ, file already exists error  
\*\*DATA to ACK or ACK to DATA, invalid opcode error

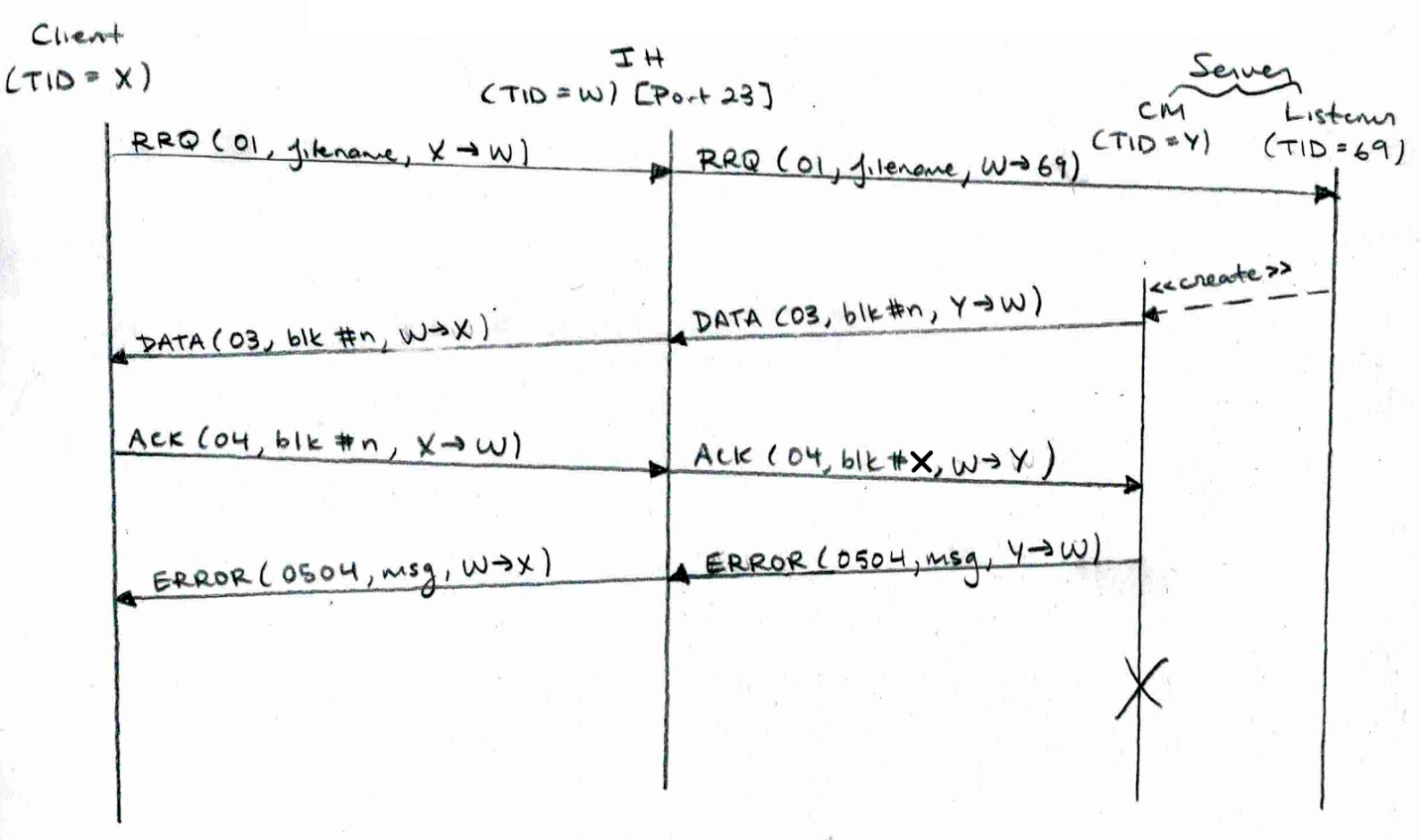


\*\* ACK made too small



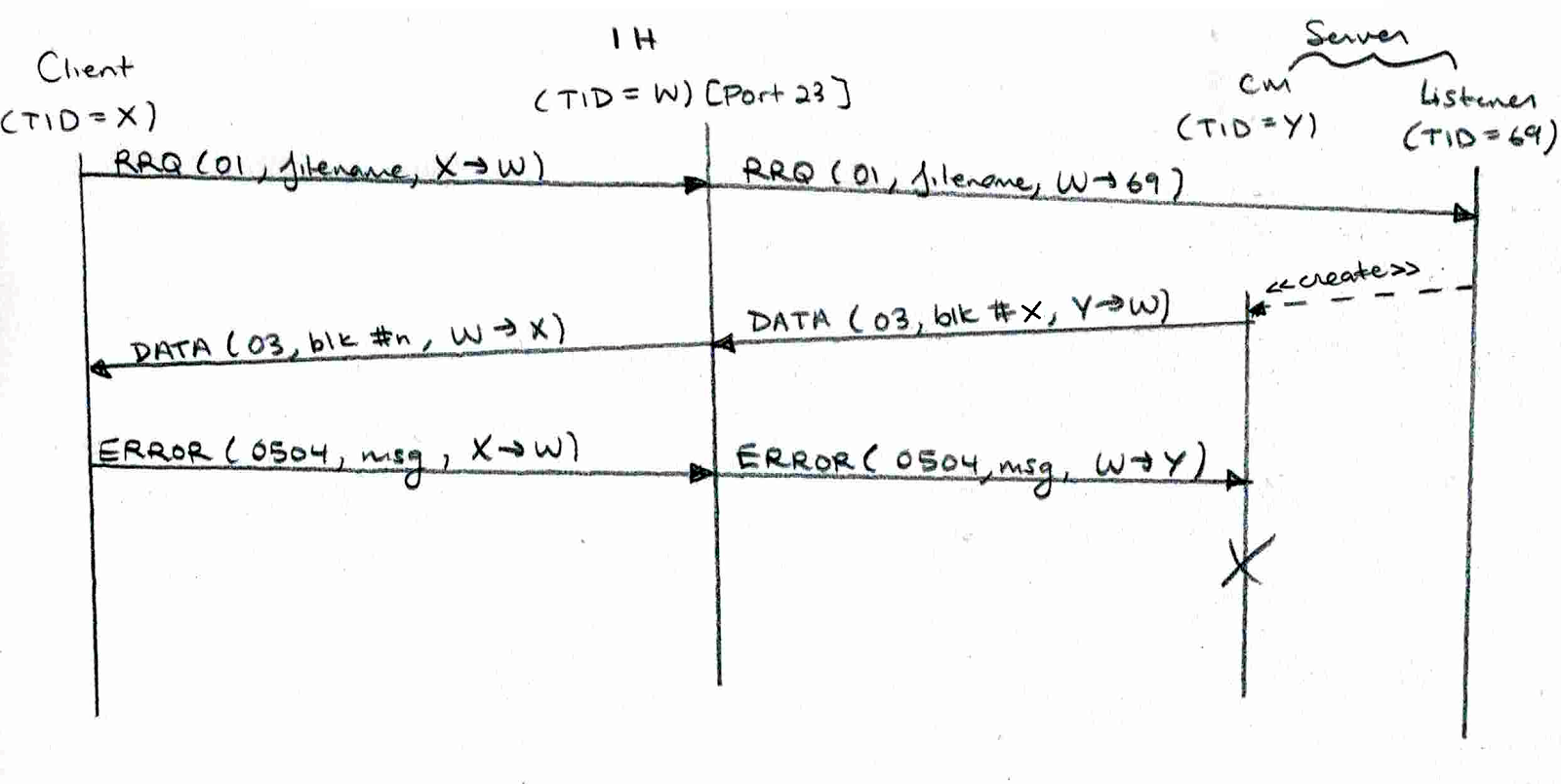
\*\* DATA packet made too large

Timing Diagram, Error 4, Invalid Block Number, to Server



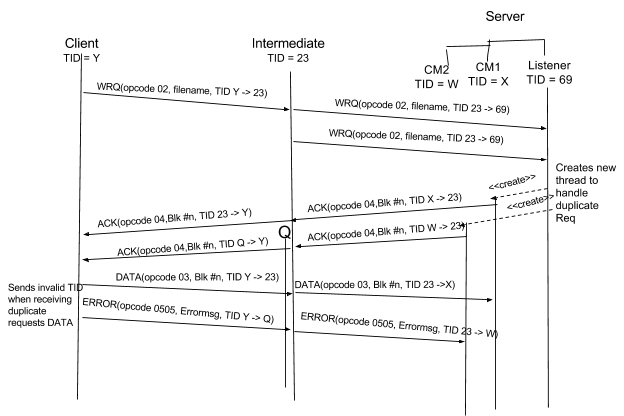
\*\* if repeated ACK, server will ignore

Timing Diagram, Error 4, Invalid Block Number, to Client



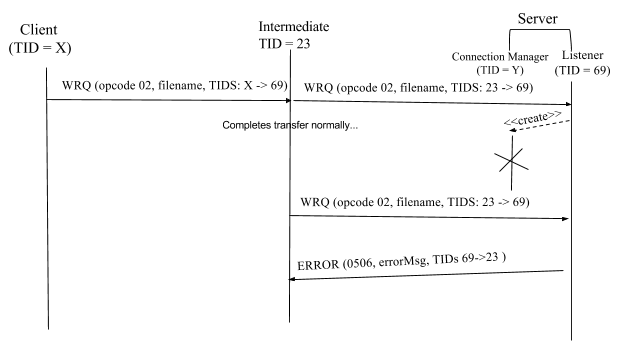
\*\* if repeated DATA, client will send ACK and server will ignore repeated ACK

Timing Diagram Duplicate WRQ



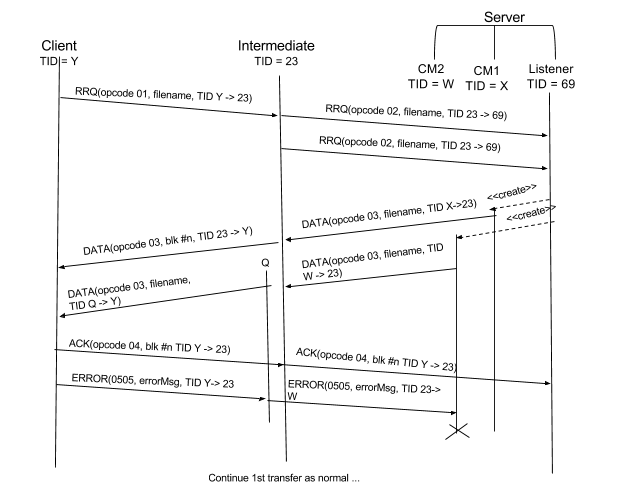
\*\*Note, intermediate uses a new port to simulate invalid tid from server

Timing Diagram Duplicate WRQ, delayTime > transfer time



\*client never receives error packet because it has already re prompted from the previous transfer

Timing Diagram Duplicate RRQ



\*\*Note, intermediate uses a new port to simulate invalid tid from server

# Testing Instructions & Files

#### FileTransferClient.java:

* The client is able to send a read/write request
* establishes connection with server and sends file in blocks of 512 bytes
* Read: read file from server, Write: write file to server
* contains a prompt method that prompts user for path, IP, Normal vs Test mode, Verbose vs Quiet mode, read vs write and file name
* extends Host.java

#### IntermediateHost.java:

* passes messages and responses between server and client in the form of packets when client is running in test mode
* performs various packet manipulations
* extends Host.java

#### FileTransferServer.java:

* Receives messages from client via IntermediateHost and responds
* If RRQ, responds with acknowledgement packet
* If WRQ, responds with data packet
* extends Host.java

#### Host.java:

* Abstract class that maintains all send and receive methods

#### Printer.java:

* Takes care of all print messages to show what is happening with client/intermediate/server

#### RequestType.java

* Class with enums of all different types of packets

#### Validater.java

* Class which contains all validation methods for packet type, validity of file, and a method that parses out filename and mode

# 

# Instructions

When running in normal mode:   
1. Run main for FileTransferServer   
2. Run main for FileTransferClient

When running in test mode:   
1. Run main for FileTransferServer   
2. Run main for IntermediateHost   
3. Run main for FileTransferClient

## General

* Will be prompted by client to enter a pathname.
* Type help when prompted to enter a command by client to see a list of commands you can use
* Read request will transfer file from Desktop to user specified path
* Write request will transfer file from user specified path to Desktop
* In the client you will be asked for a path, this is the path to where you want to save files and upload from.
* If you want to change directories, simply type in the new directory at anytime
* The available commands for the client are
  + quit - exits the client
  + pwd - present working directory
  + ls - list of files in directory
  + normal - sends packets from client directly to server
  + test - sends file to intermediate host which forwards to server
  + read - reads a file from server
  + write - writes a file to server
  + verbose - prints all data
  + !verbose - prints minimum amount of information
  + help - shows commands and sample queries

An example of a command is normal/test read/write filename.txt verbose/!verbose

# Testing Errors

## Disk Full

obtain a USB disk that is basically full

you can do this through the command line by running this command where path is the path of your USB and size is how much you want to fill your USB

Windows: fsutil file createnew <PATH> <SIZE>

Linux: mkfile <SIZE> <PATH>

Client

set the path of the client to the USB

Make a RRQ for a file in Server, and the file should stop reading as soon as the disk is full

client will send an error packet back to client saying the disk is full

Server

the path of the server directory must be changed to the USB’s path within the code

server will send an error packet back to client saying the disk is full

## File Already Exists

Client

run a read request on a file that is already if the client’s folder

the client will re-prompt the user saying that the file already exists

Server

run a write request on a file that is already in src/serverFiles

the server will send an error packet back to client saying the file already exists

client will be re-prompt

## File Not Found

Client

run a read request on a file that does not exists in src/serverFiles

the server will send an error to client saying that the file does not exist

Server

run a write request on a file that does not exists in the client folder

the client will re-prompt because the file doesn't exist

## File Access Denied

Take the file that needs to be read or written and change its access to read-only

this will cause the error to occur on both client and server side

## Illegal TFTP

An illegal TFTP operation can be performed using the intermediate host

The options in the Intermediate host that provide this error are:

Corrupt Request Packet

Change OpCode

Invalid Packet Size

Change the Block Number (if block is too large, an error will be send, if it’s a repeated number, it will be treated accordingly)

Follow all prompts accordingly to create an Illegal TFTP.

Unknown Transfer ID

An unknown transfer ID can also be performed through the intermediate host

Use Change TID prompt and follow the instructions accordingly to create the error

Intermediate Host – Normal, Lost, Delay, Duplicate

Use the intermediate host to perform these actions

Normal: passes packets through IH without changing them

Lost: will lost the specified packet

Delay: will delay the specified packet for a user inputted time

Duplicate: will duplicate a packet after a user inputted time

Follow the prompts that intermediate host gives.

Intermediate Host – Manipulating Error Packet

Use intermediate host to perform these actions

Lose an error packet

Delay an error packet

Duplicate an error packet

Send an error packet from an invalid TID

Change the opcode of an error packet

Change the code of an error packet

\*\* Note this can only be performed on a “naturally error inducing situation”, e.g. Write with file not found, file does not exist, access denied, disk full and read with disk full.