SYSC 3303 Project

TFTP file transfer system

Lab #2 Team #7

Team member:

Yunkai Wang(100968473)

Qingyi Yin(100968696)

Table of Contents

- Task Breakdowns & Responsibilities……2
- Detail Set Up & Test Instructions……..2
- Diagrams…...……..8
    - UCM……8
    - UML……...10
    - Timing diagrams……11

Task Breakdowns & Responsibilities

Yunkai Wang: Most of coding part and changed error simulator & request handler & client to handler duplicate/delay/lose packet cases, improve the usability of the program

Qingyi Yin: TFTPAckPacket class and all diagrams/documents

Detail Set Up & Test Instructions

Create test files

Step 1: Go to the project directory.

Step 2: Create test files in 'client_files' and 'server_files' folder (create these two folders if they do not already exist)

Step 3: To create test files conveniently, in client's or server's terminal, type 'touch <filename> <size>" to create a file with random bytes with the given size

Set up environment

Step 1: After importing the project, right click on the project -> click properties -> choose 'Java build path' -> select 'libraries' tab -> click 'add Library' -> choose 'JRE system library' and click next -> choose 'Alternate JRE' and click 'finish' -> click 'apply and close'.

Step 2: right click on the project -> click properties -> choose 'Java Compiler' -> uncheck 'Enable project specific settings' option on the top -> click 'apply and close'. (click yes if the rebuild request appears)

Run the program

Step 1: Go to TFTPClient then click run.

Step 2: Go to TFTPServer then click run.

Step 3: Go to TFTPErrorSimulator then click run.

Instructions of using our system (terminal input)

TFTPClient: (important commands are labeled in red)

- menu: To show the menu, type "menu" then press enter
- exit: To stop the client and exit, type "exit" press enter
- mode: To see the current mode, type "mode" then press enter

- switch: To switch mode(verbose, quite, normal, or test), type "switch" then press enter

- reset: To switch running mode(test or normal), type "reset" then press enter

- read: To read a file from server, type "read <filename>" then press enter

- write: To write a file to server, type "write <filename>" then press enter

- la/ls: To see all the files under current directory, type "la" then press enter

- rm: To remove a file under current directory, type "rm <filename>" then press enter

- pwd/dir: To see current directory path, type "pwd" or "dir" then press enter

- cd: To check the directory path, type "cd <path>" then press enter

- ip: To check the server address that the client is currently sending request to, type "ip" then press enter

- connect: To send the RRQ/WRQ to the server that runs on different host, type 'connect <ip_address>" then press enter

- touch: if you want to create a new file in the current directory, for convenience, you can simply type "touch <filename>" or "touch <filename> <size>" to create a new file with 0 byte or with the given size

TFTPServer:

- menu: To show the menu, type "menu" then press enter

- exit: To stop the client and exit, type "exit" press enter

- mode: To see the current mode, type "mode" then press enter

- switch: To switch mode (verbose, quite, normal, or test), type "switch" then press enter

- count: To see current number of running threads, type "count" then press enter

- la/ls: To see all the files under current directory, type "la" then press enter

- rm: To remove a file under current directory, type "rm <filename>" then press enter

- pwd/dir: To see current directory path, type "pwd" or "dir" then press enter

- cd: To check the directory path, type "cd <path>" then press enter

- touch: if you want to create a new file in the current directory, for convenience, you can simply type "touch <filename>" or "touch <filename> <size>" to create a new file with 0 byte or with the given size

TFTPErrorSimulator:

- menu: To show the menu, type "menu" then press enter

- exit: To stop the client and exit, type "exit" press enter

- normal: To perform a TFTP file transfer without any error, type "normal" press enter

- lose: To perform a TFTP file transfer with packet lose, type "lose" press enter

- delay: To perform a TFTP file transfer with packet delay with the a specified delay time, type "delay <millisecond>" press enter

- duplicate: To perform a TFTP file transfer with duplicate packet, type "duplicate" press enter

- ip: To check the server address that the error simulator is currently sending request to, type “ip” then press enter

- connect: To forward the request to the server that runs on different host, type ‘connect <ip_address>” then press enter

If you entered “corrupt” in previous choice, the program will ask you for type of corruption to perform on the packet, where you are given the following choices to choose:

1. Change opcode in packet
2. Remove ‘0’ byte after file name
3. Remove ‘0’ byte after mode
4. Remove file name
5. Change mode in packet
6. Add extra byte to the packet
7. Shrink the packet
8. Change block number

If you entered "lose", "delay" or "duplicate" in previous choice, the program will ask you to input again, where you will be choosing the type of packet and block number to simulate the error:

- request: To simulate the error on request packet, type "request" and press enter
- data: To simulate the error on data packet, type "data <blkNum>" press enter, the block number is the specified data packet which you want to simulate the error

- ack: To simulate the error on ack packet, type "ack <blkNum> press enter, the block number is the specified data packet which you want to simulate the error

Testing Steps

1. Follow the instructions in “setup environment” section to configure all the applications and start all the applications
2. If you want to test normal mode
    a. In terminal, type read <filename> to test read request (make sure you have <filename> in your server_files folder)
    b. In terminal, type write <filename> to test write request (make sure you have <filename> in your client_files folder)
    c. Type 'mode' to switch the mode
    d. To test file transfer on different computer, in the client’s terminal, type ‘connect <server_ip>” so that all requests will be send to the given ip address
3. If you want to test test mode
    a. In client’s terminal, type reset (this will set the running mode to test)
    b. To send a RRQ or WRQ in test mode, make sure that the client will send the requests to the error simulator, and the error simulator should be running on the same computer as the client, and the error simulator will send the request to the ip address where the server is running on
    c. In error simulator’s terminal, follow step 4 to select the type of errors you want to simulate on the nest transfer, or type ‘normal’ to test a file transfer without simulating error
    d. In client’s terminal, type ‘read <filename>’ or ‘write <filename>’ to send the RRQ or WRQ
4. If you want to simulate error (make sure you are in test mode first)
    a. In error simulator's terminal, choose the type of error based on the given menu
    b. If you want to test lost packet, type "lose" and press enter, then select the packet type and press enter
    c. If you want to test delay packet, type "delay" and press enter, then select the

packet type and press enter

d. If you want to test duplicate packet, type "duplicate" and press enter, then select the packet type and press enter
e. If you want to test corrupt packet, type ‘corrupt’ and press enter, then select the corruption type, the packet type and press enter
f. If you want to test unknown tid error, type ‘tid’ and press enter, then select the packet type and press enter

5. In client’s or server’s terminal, type ‘mode’ to switch the printing mode
6. In all three programs’ terminal, type ‘exit’ to stop the application

Important instruction about TFTPErrorSimulator program

1. As I simplified error simulator’s logic, it will only receive a packet and forward the packet, so the error simulator will have no idea about when the TFTP file transfer has ended (since it doesn’t check if the data packet is the last packet), so instead, we set the error simulator’s socket’s timeout to be 10 seconds. If no packet is received within 10 seconds, the error simulator will assume that the file transfer has ended and return to the ask for input part. (The time out time is set to be 10 seconds since any side may retransmit the packet for up to 5 times, and wait for request for 2 seconds, so if no packet is received within 10 seconds, we are safe to assume that the connection has ended) So make sure to wait for 10 seconds after you have used error simulator for one file transfer.
2. The error simulator is single thread program most of the time, however, in only one case the program will become multi-thread which is the case when the user choose to delay a packet, in that case, a new thread will be created and it will send the packet after the specified time, in any other cases, the error simulator will be single thread, so make sure to choose the type of error you want to simulate before using the error simulator in TFTP file transfer. After you have chosen the type of error to simulate (or type normal for a normal file transfer), the error simulator will start to wait for new request.
3. Please note that when you choose to corrupt a packet, you must choose valid packet type for corruption. This means that for instance, if you choose to simulate the “remove file name” corruption, this error must be simulated on request packet, but the program

doesn't perform any checking. So, if you choose the "remove file name" corruption and a different packet type to simulate the error, the error will not be simulated as data packet or ack packet doesn't has any file name contained.

Explaining the names of files:

- TFTPPacket.java - abstract class for all TFTP packets which defined several common functions
- TFTPAckPacket.java - class for TFTPAckPacket
- TFTPDataPacket.java - class for TFTPDataPacket
- TFTPErrorPacket.java - class for TFTPErrorPacket
- TFTPRequestPacket.java - class for TFTP RRQ request for TFTP WRQ request
- TFTPErrorType.java - enum class for all possible types of TFTP errors
- TFTPErrorException.java - exception that is thrown when a TFTPErrorPacket is received, or a TFTPErrorPacket is sent, used for terminating the connection
- TFTPClient.java - TFTP client class, can send RRQ or WRQ to the TFTP server
- TFTPServer.java - TFTP server class, it will initialize a TFTP request listener thread which will listen for new request
- TFTPRequestListener.java - TFTP request listener class, it's a sub-class of thread, and it will listen to WRQ or RRQ on port 69
- TFTPRequestHandler.java - TFTP request handler class, it's a sub-class of thread, and it will handle the WRQ or RRQ received by TFTP request listener thread
- TFTPErrorSimulator.java - Used for simulating error
- Mode.java - enum class for current printing mode (quite or verbose)
- Type.java - enum class for all different types of TFTP requests (WRQ, RRQ, DATA, ACK, ERROR)
- ThreadLog.java - Helper class for all different threads to print information
- TFTPHost.java - Class that abstracts the common functions between client and server

Diagrams

- Part 1 UCM Diagrams

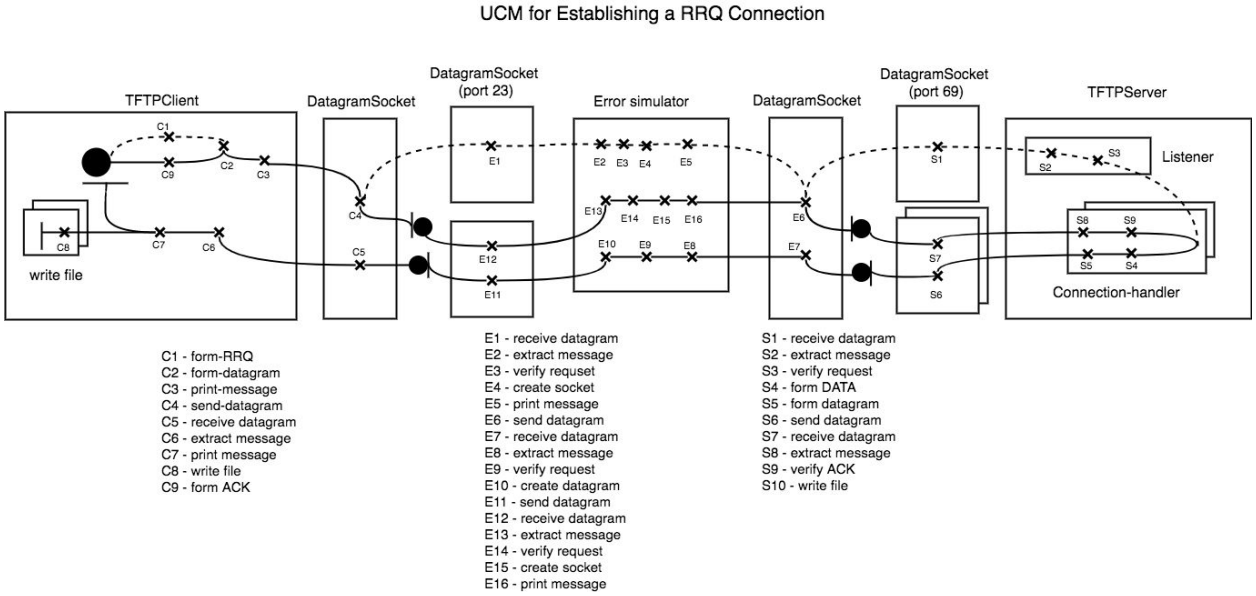

Figure 1.1: UCM for Establishing a Read Request Connection

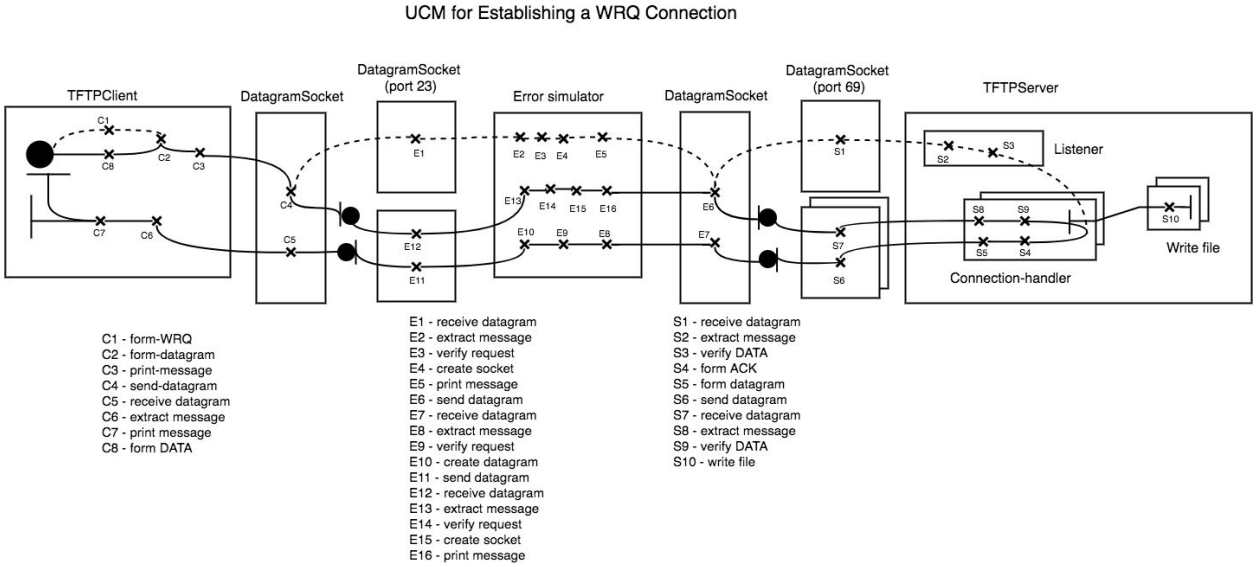

Figure 1.2: UCM for Establishing a Write Request Connection

- Part 2 UML Diagram

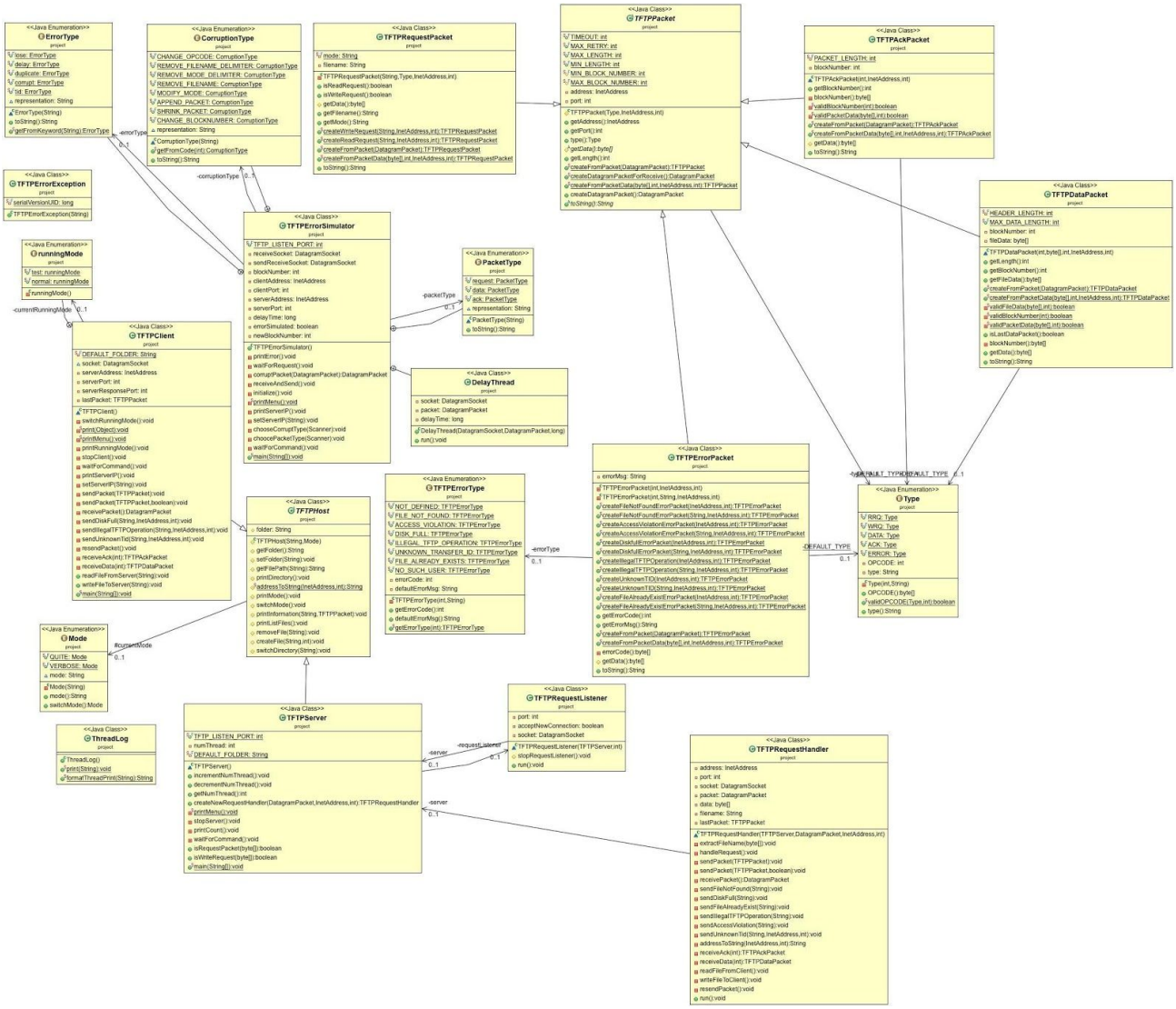

Figure 2.1: UML diagram for the final iteration

- Part 3 Timing Diagrams

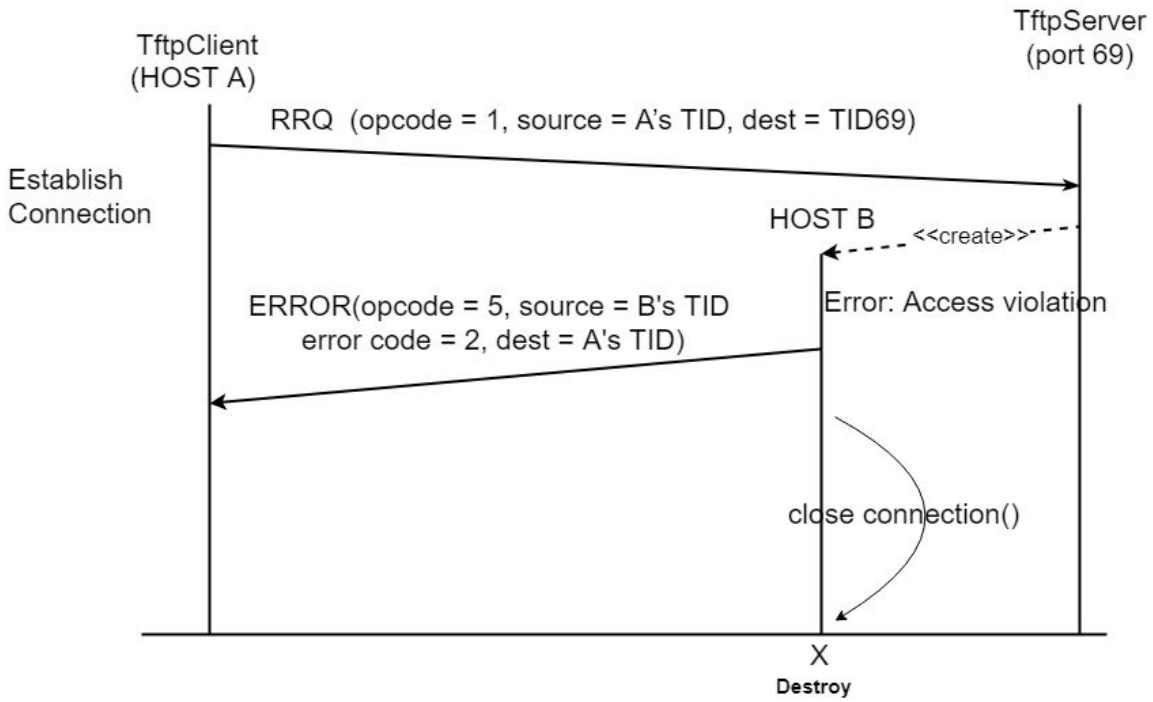

Figure 3.1: Access Violation During Read Request

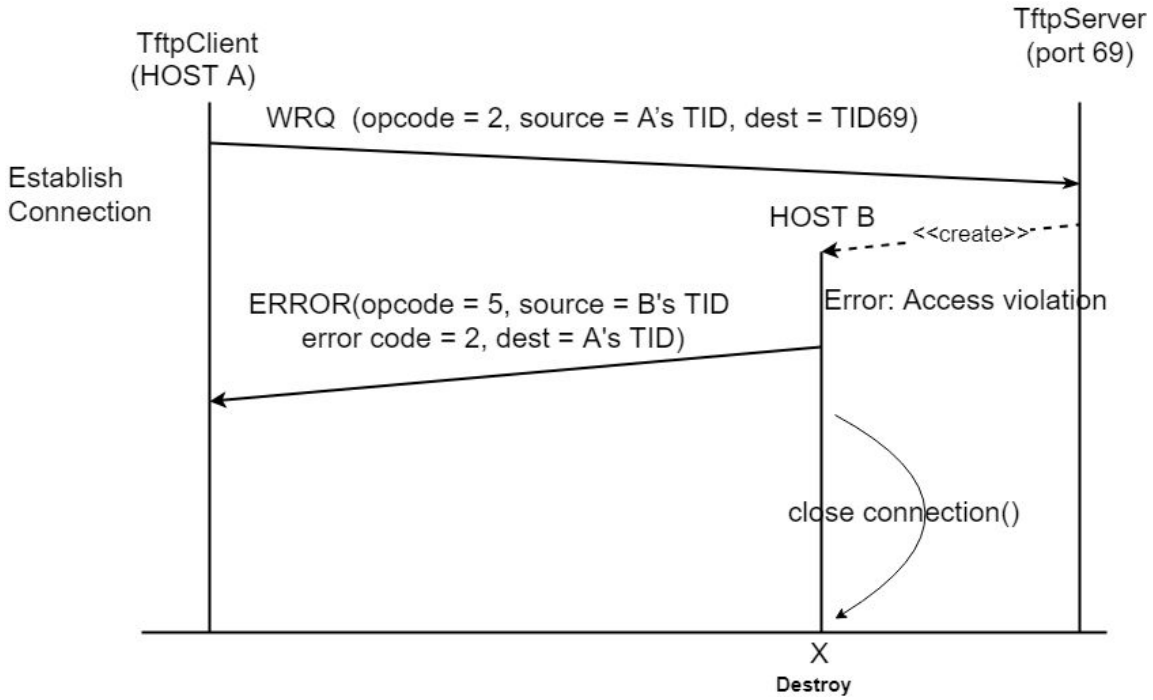

Figure 3.2: Access Violation During Write Request

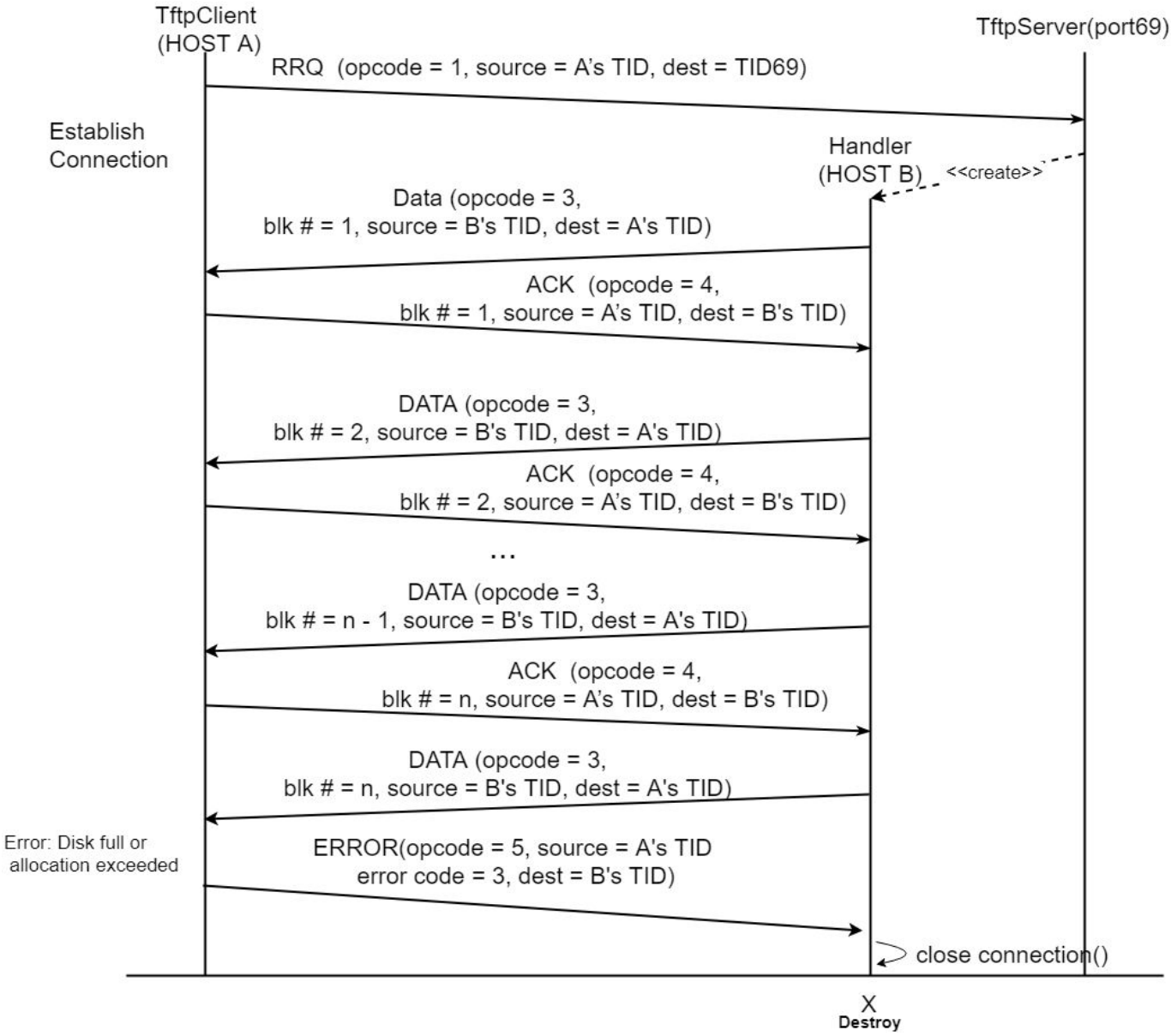

Figure 3.3: Disk Full During Read Request

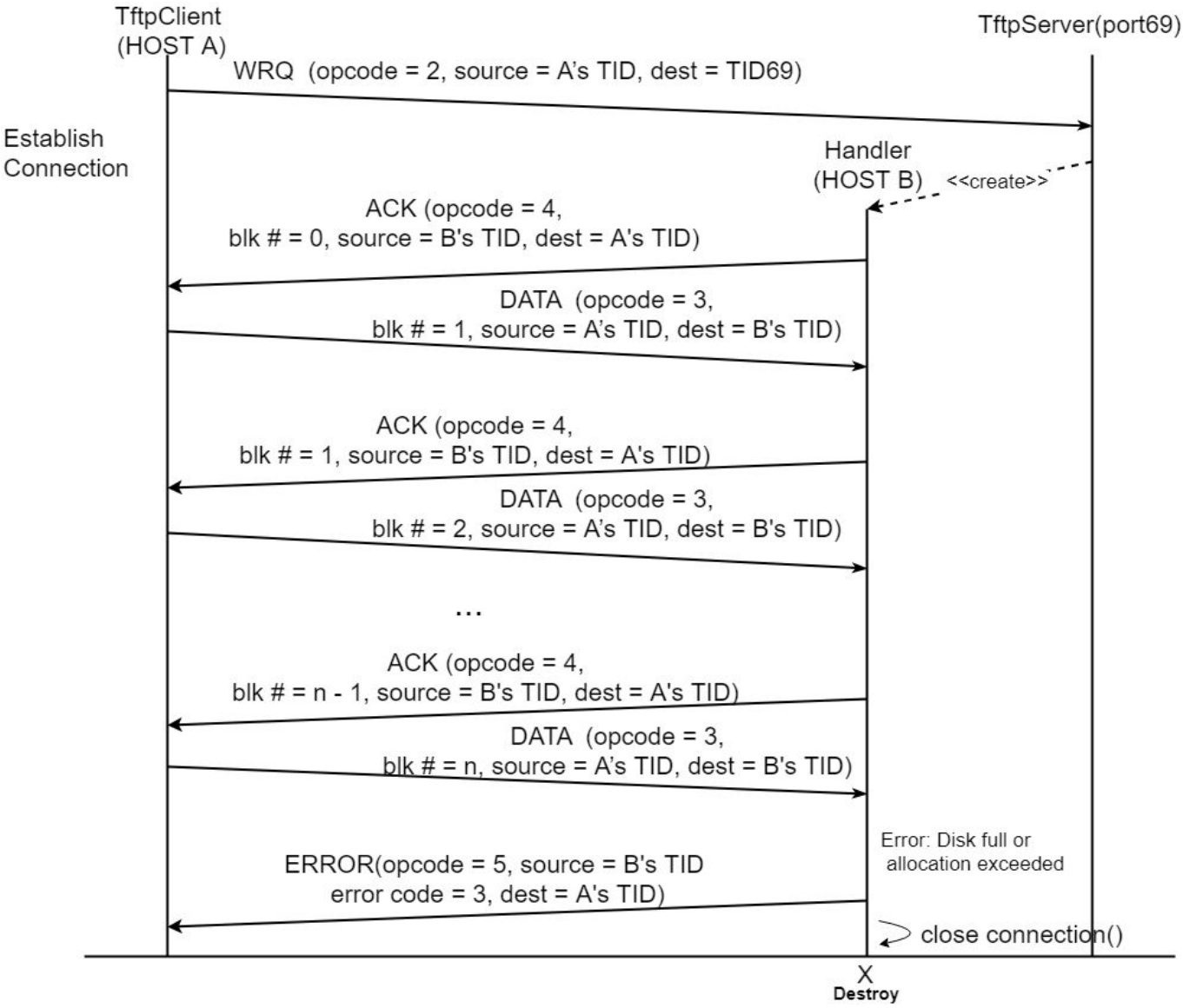

Figure 3.4: Disk Full During Write Request

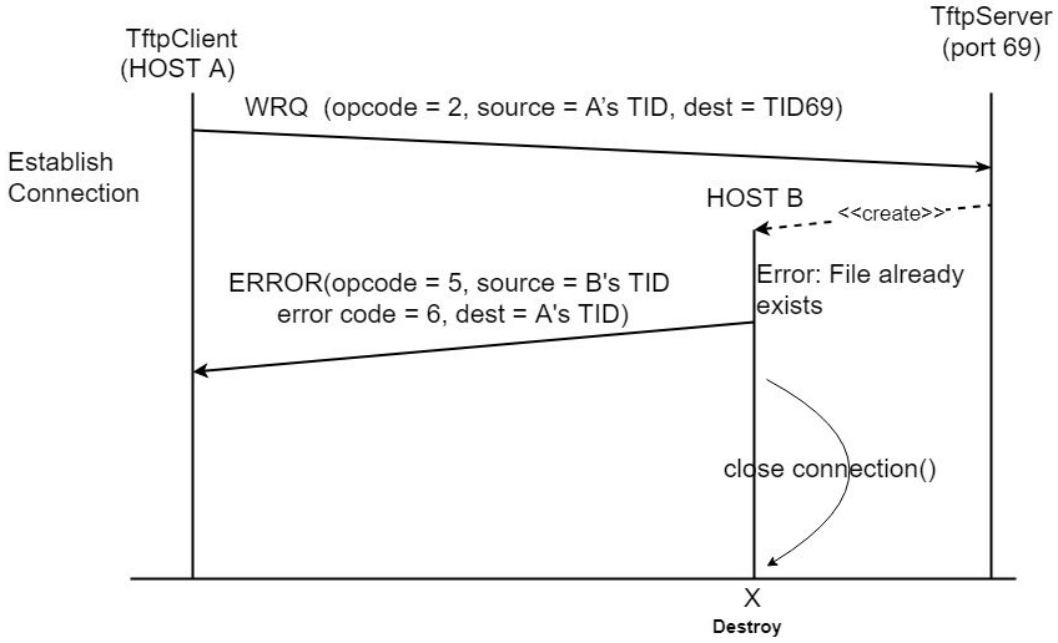

Figure 3.5: File already exist for Write Request

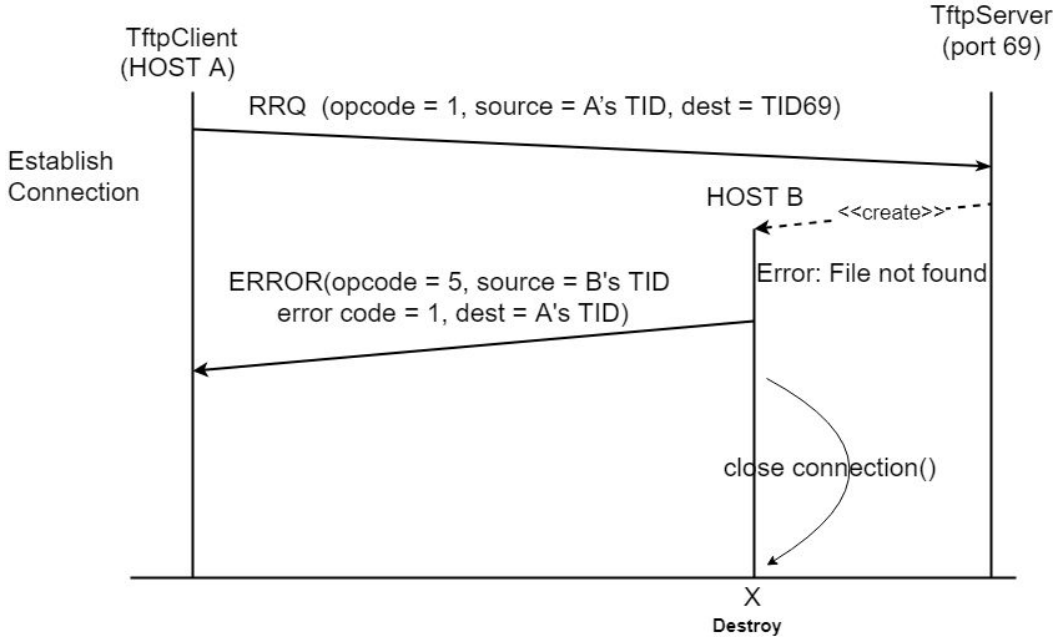

Figure 3.6: File not found for Read Request

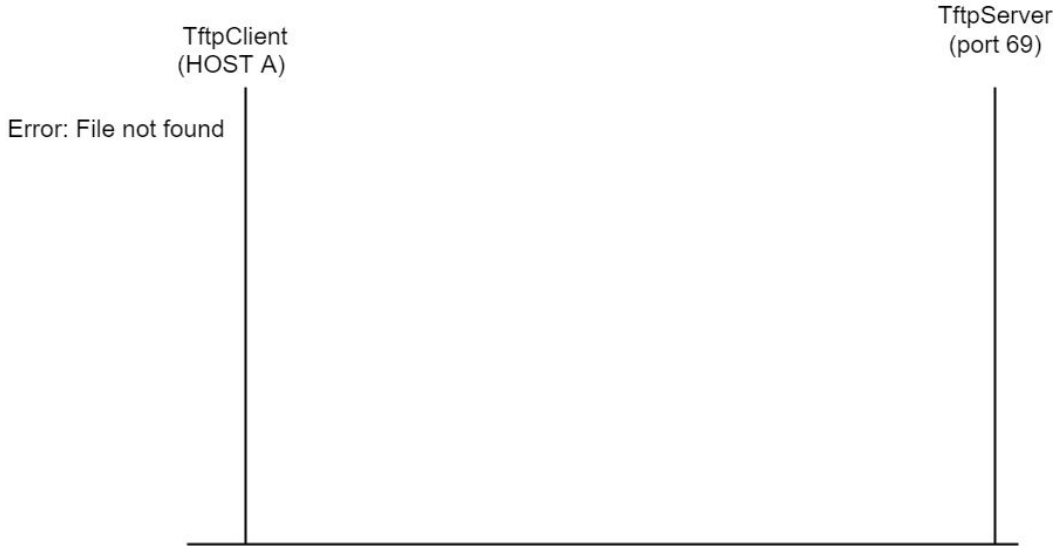

Figure 3.7: File not found for Write Request

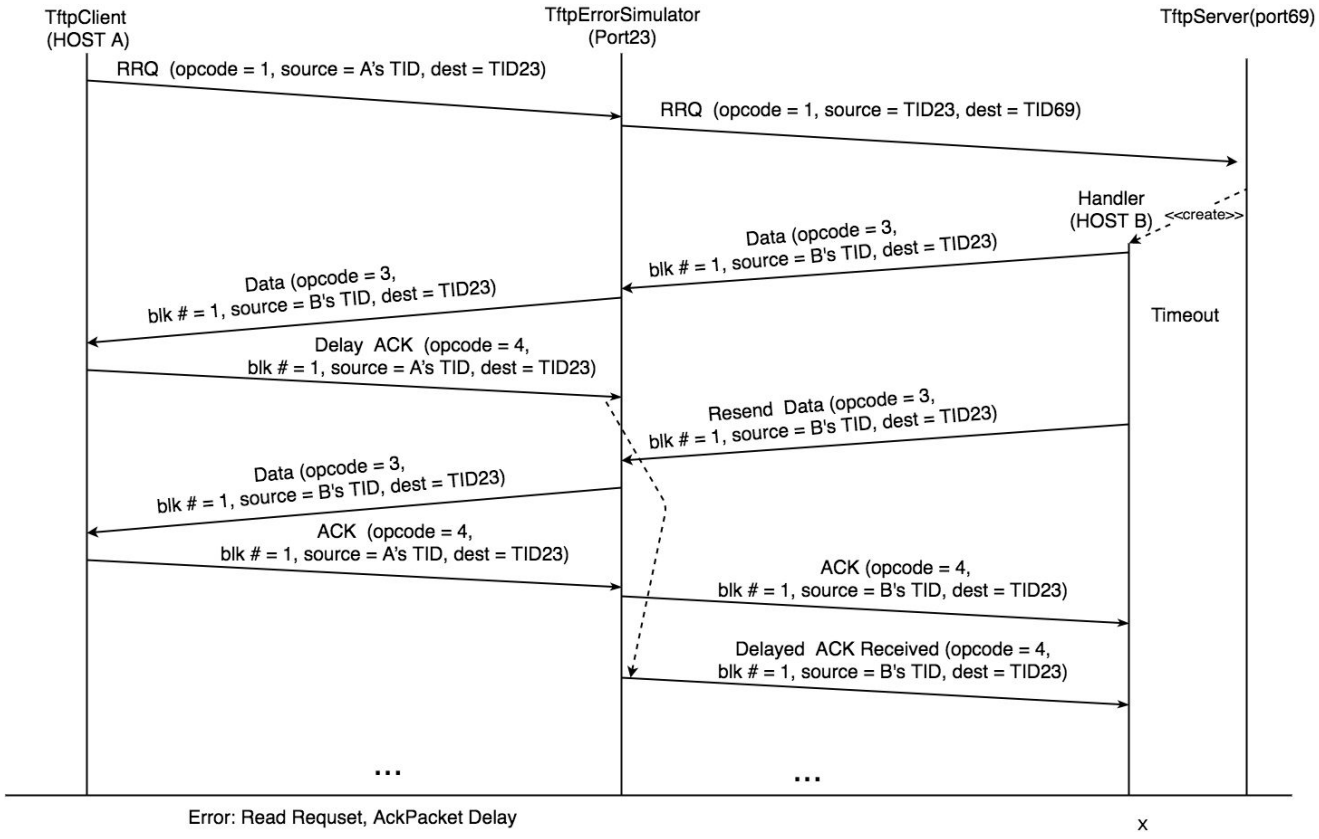

Figure 3.8: Delay AckPacket during Read Request

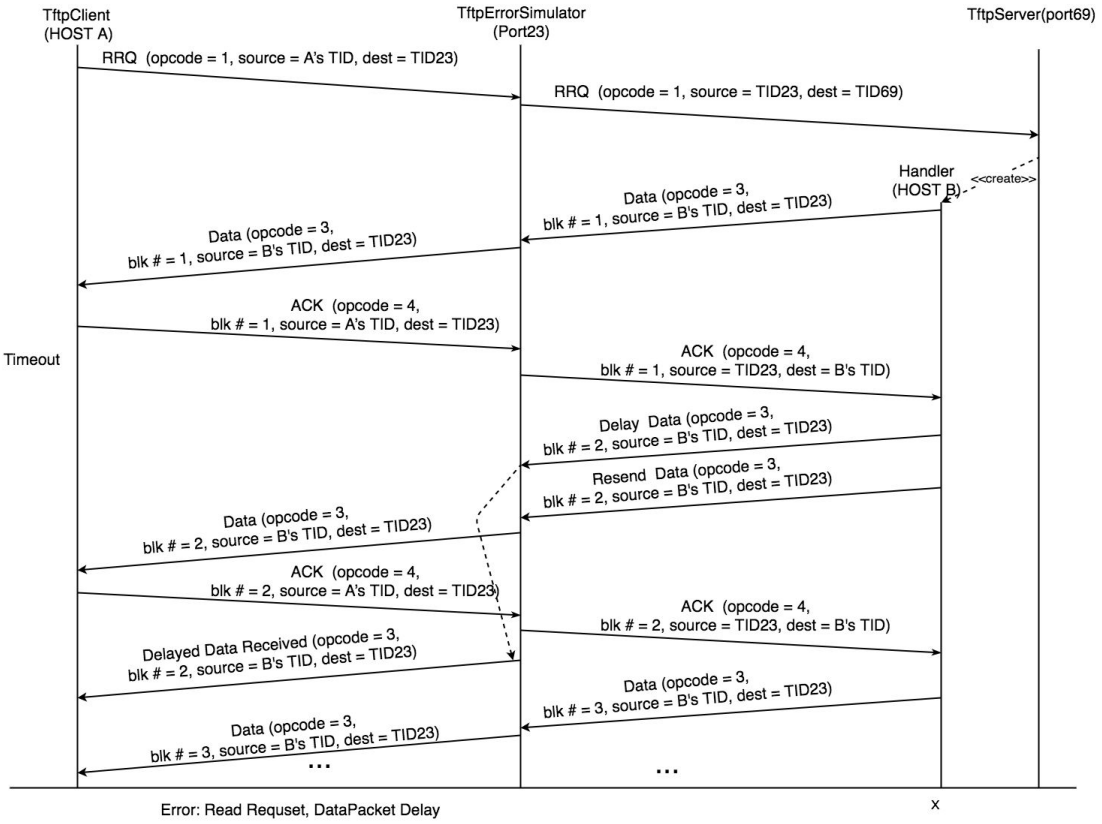

Figure 3.8: Delay DataPacket during Read Request

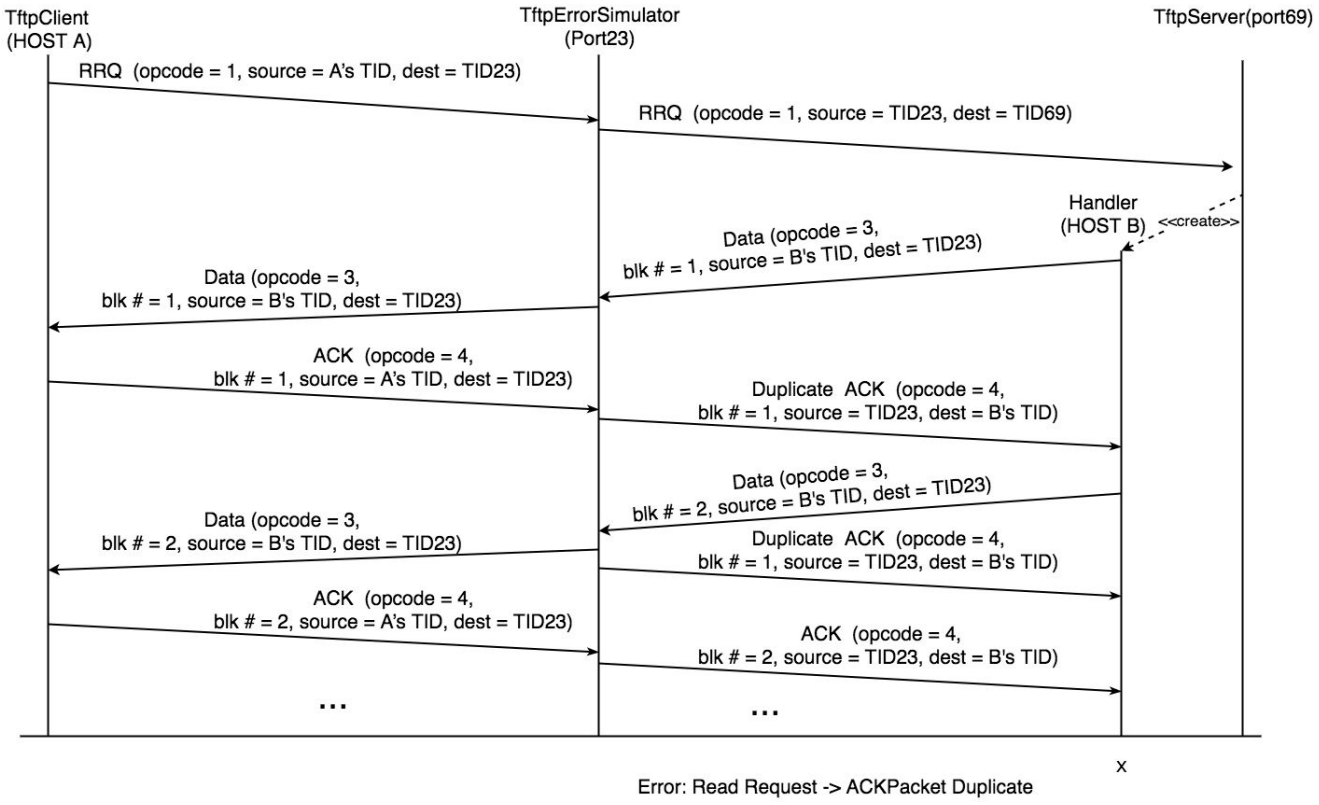

Figure 3.8: Duplicate AckPacket During Read Request

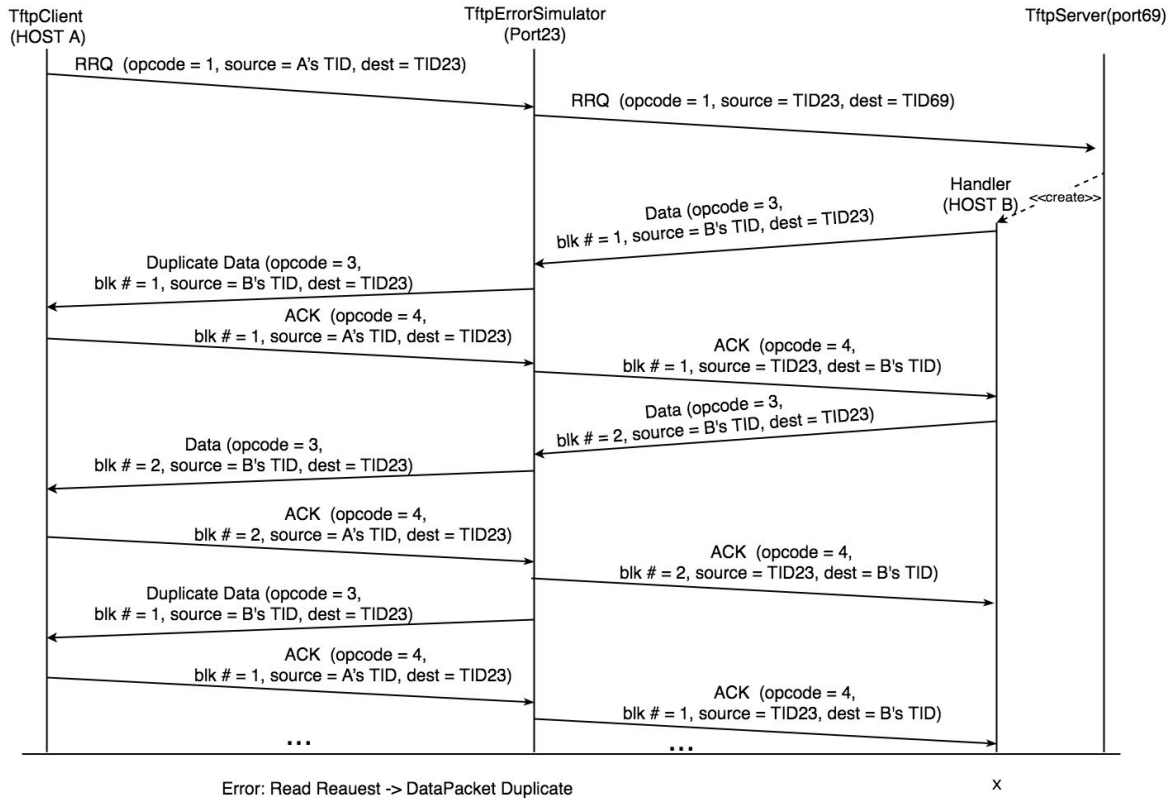

Figure 3.9: Duplicate DataPacket During Read Request

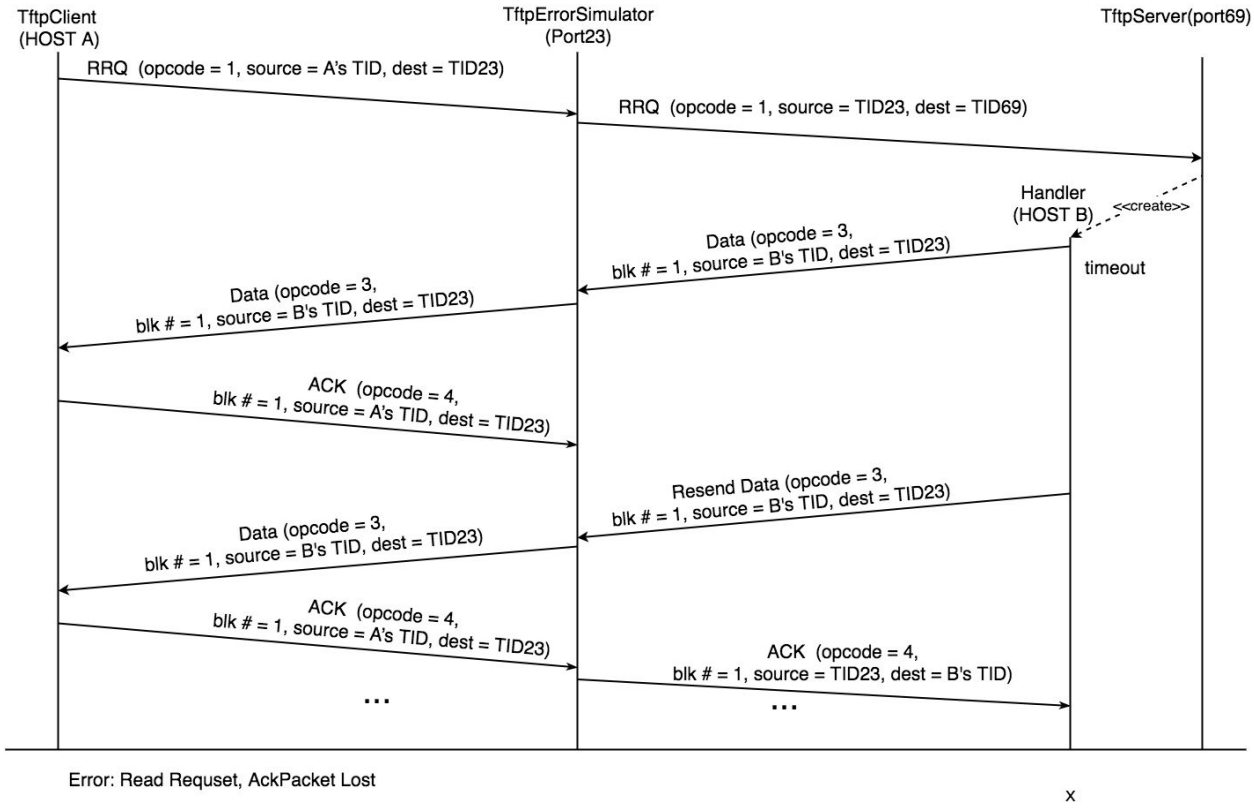

Figure 3.10: AckPacket Lost During Read Request

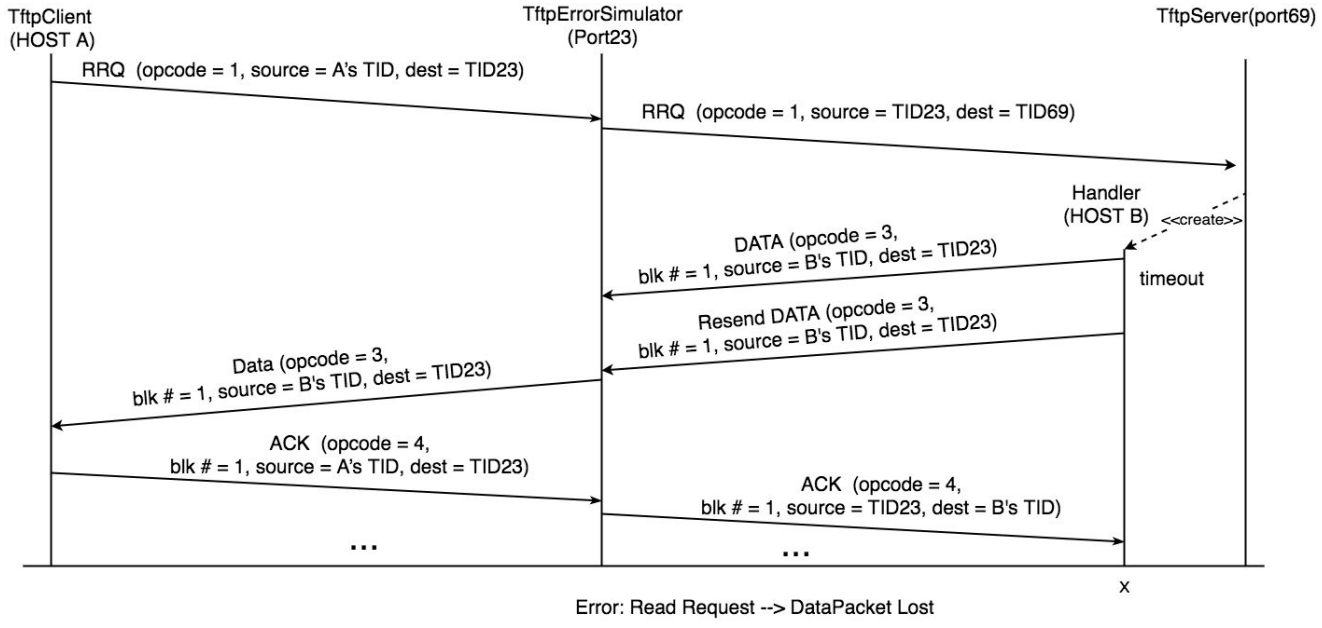

Figure 3.11: DataPacket Lost During Read Request

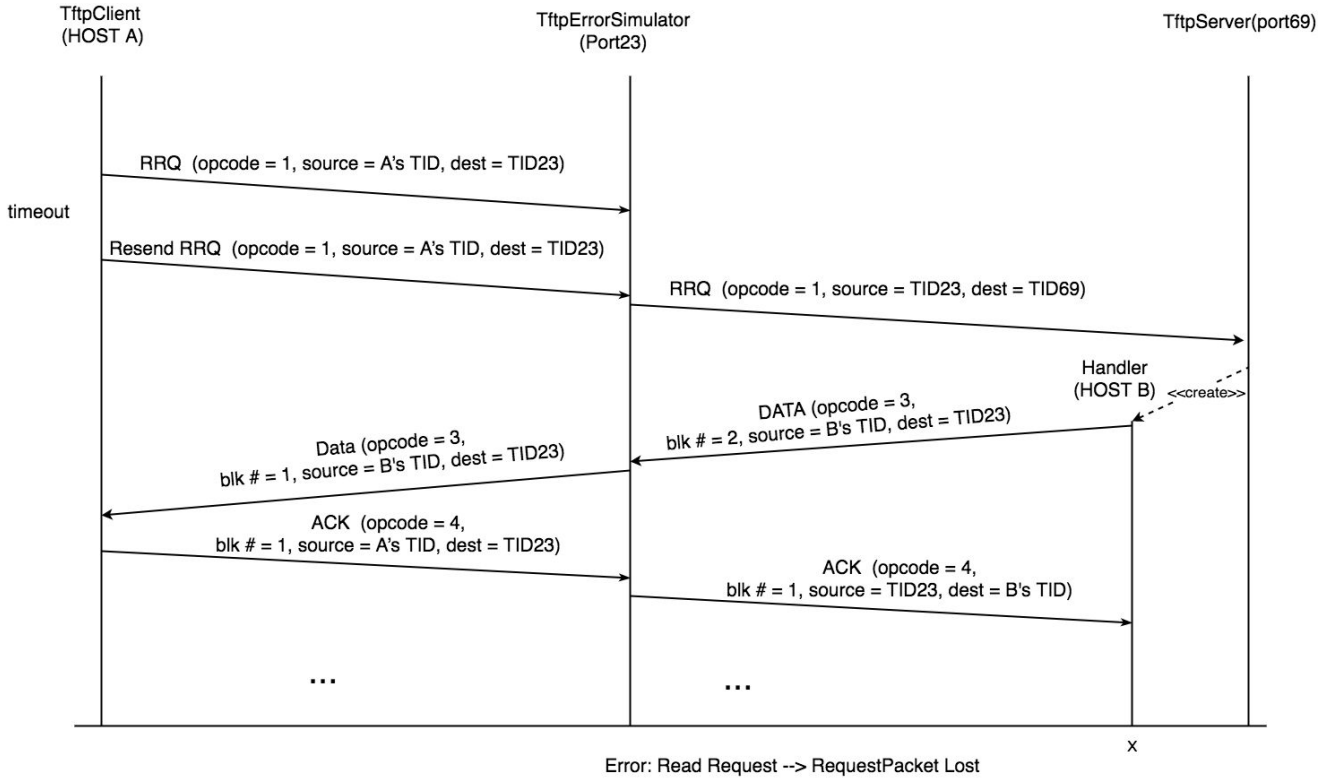

Figure 3.12: RequestPacket Lost During Read Request

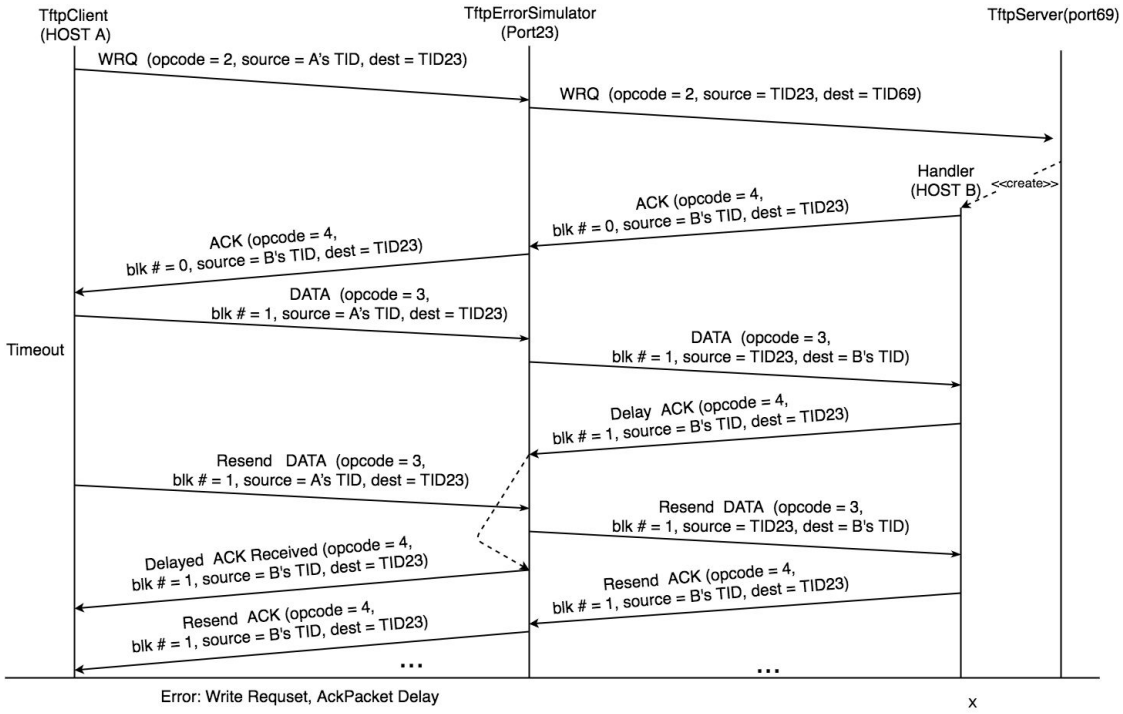

Figure 3.13: Ackpacket Delay During Write Request

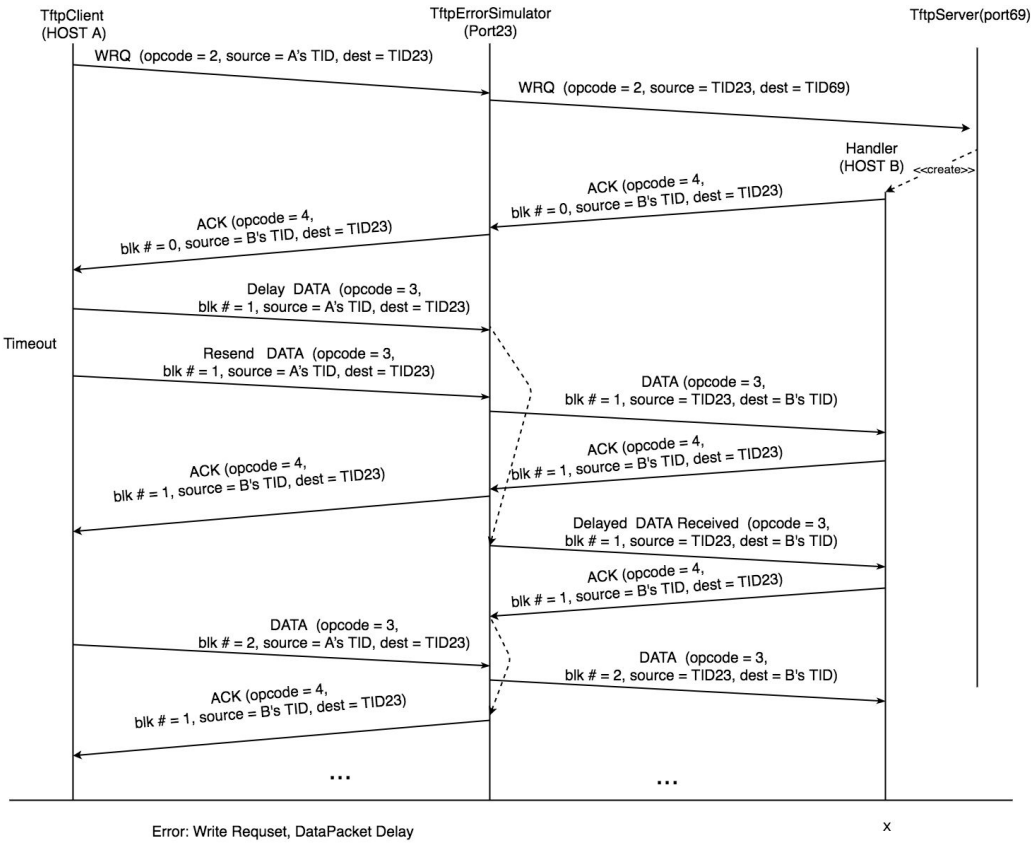

Figure 3.14: Datapacket Delay During Write Request

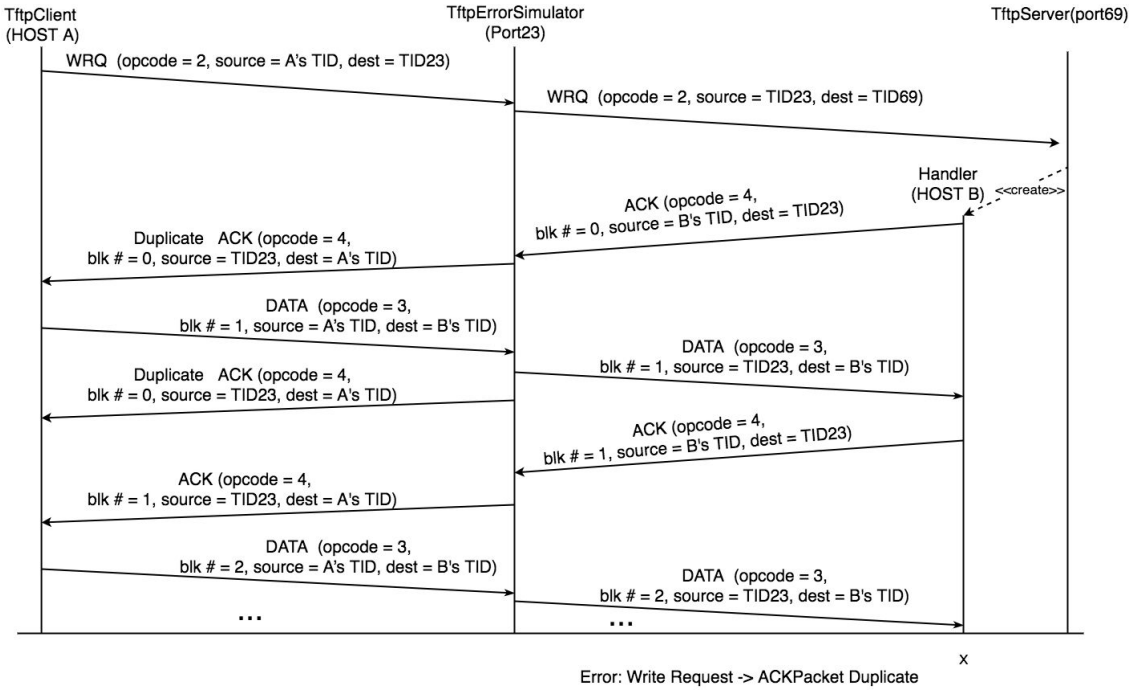

Figure 3.15: Ackpacket Duplicate During Write Request

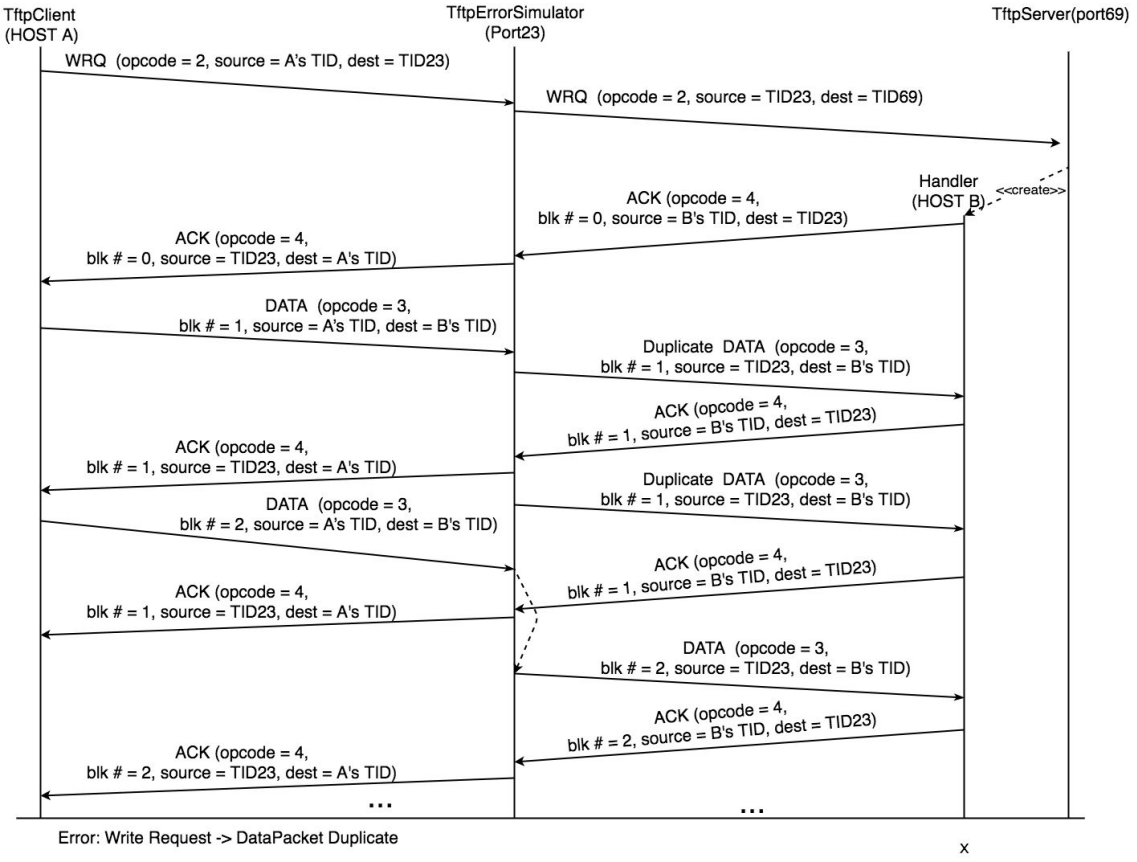

Figure 3.16: Datapacket Duplicate During Write Request

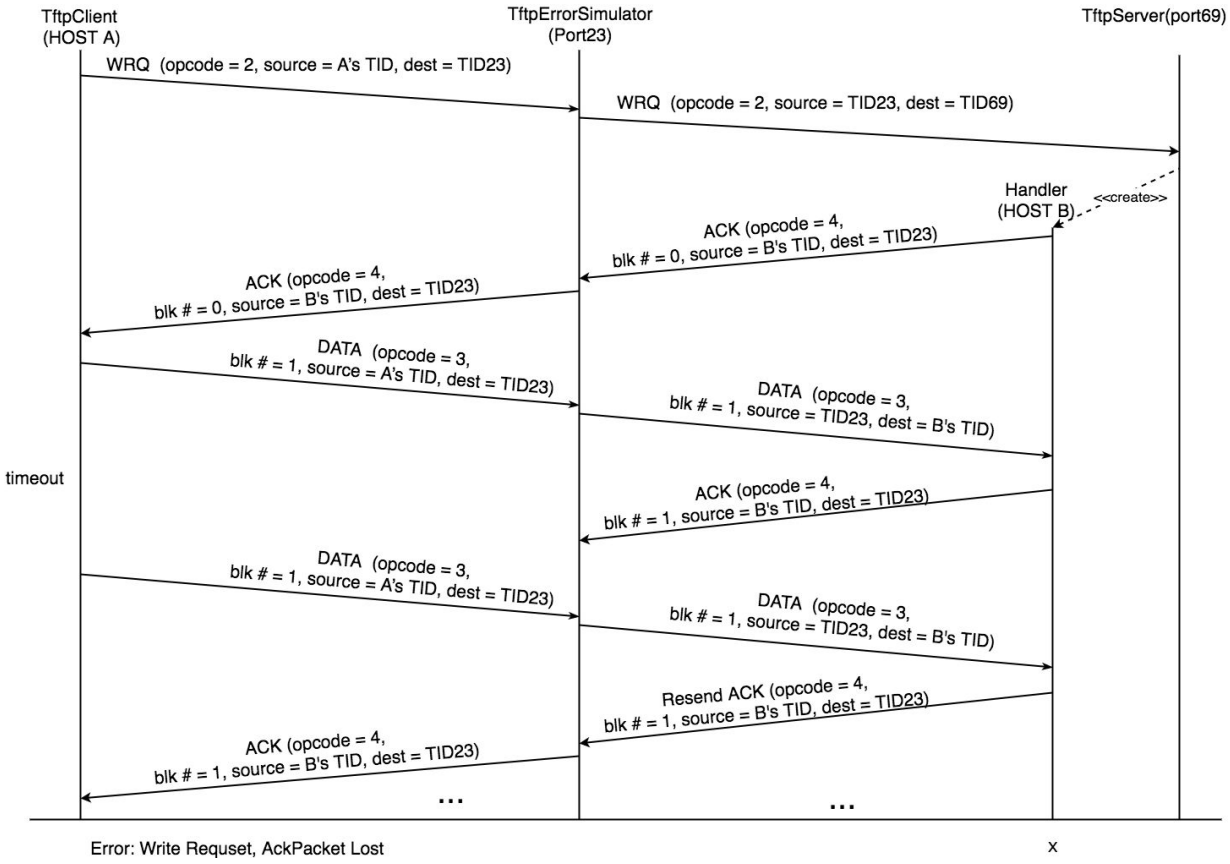

Figure 3.17: Ackpacket Lost During Write Request

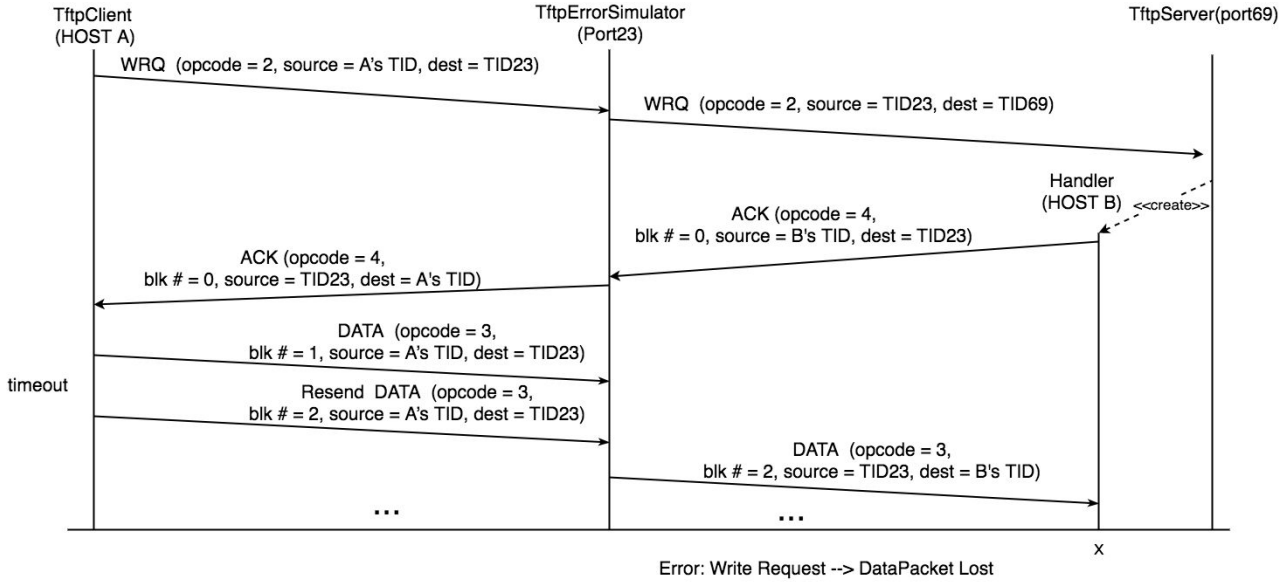

Figure 3.18: Datapacket Lost During Write Request

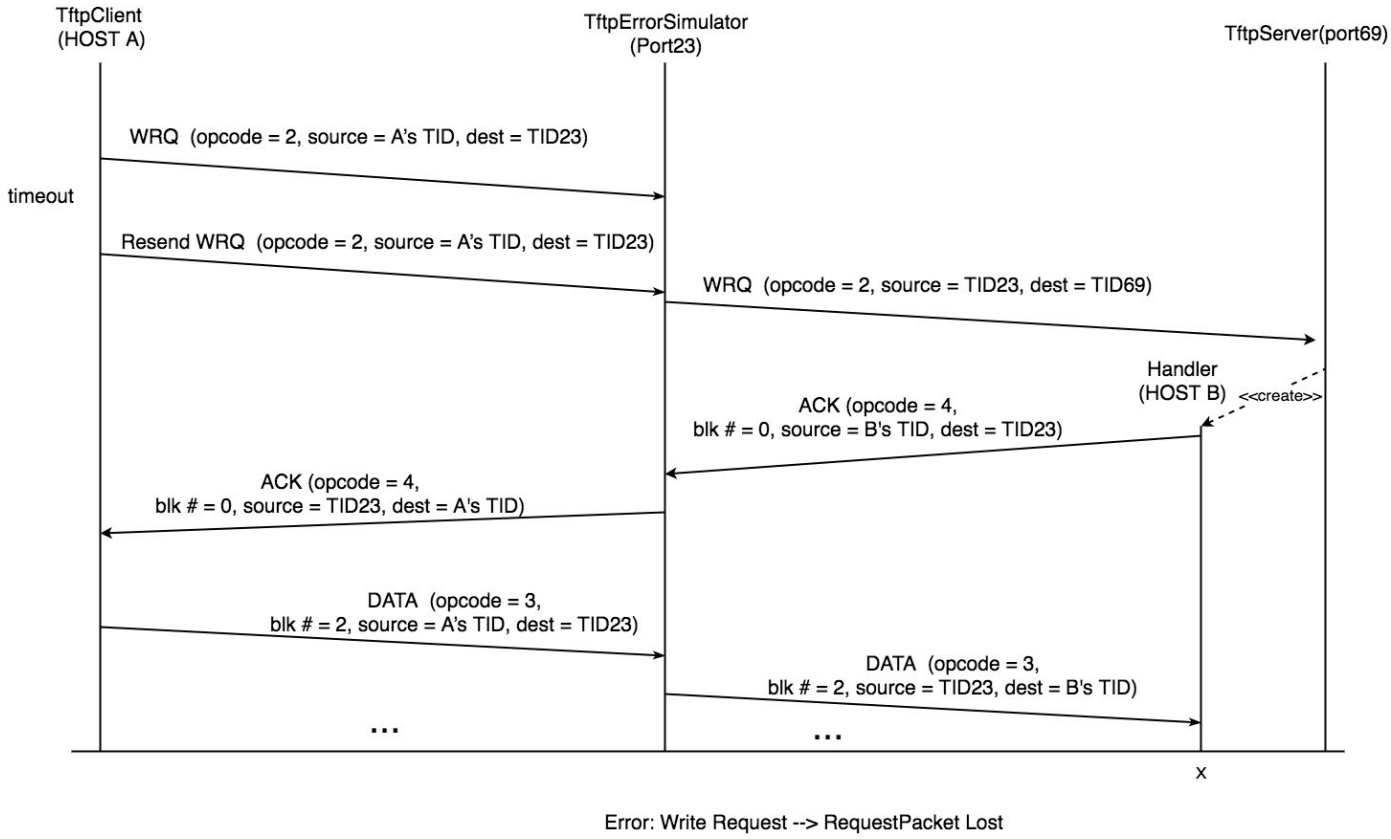

Figure 3.19: Requstpacket Lost During Write Request

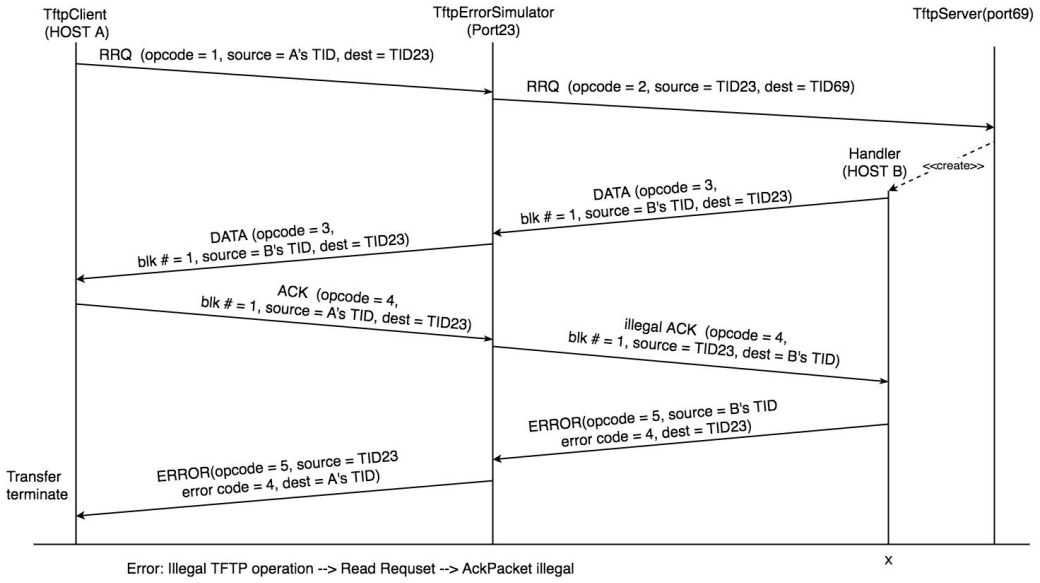

Figure 3.20: Ackpacket Illegal During Read Request

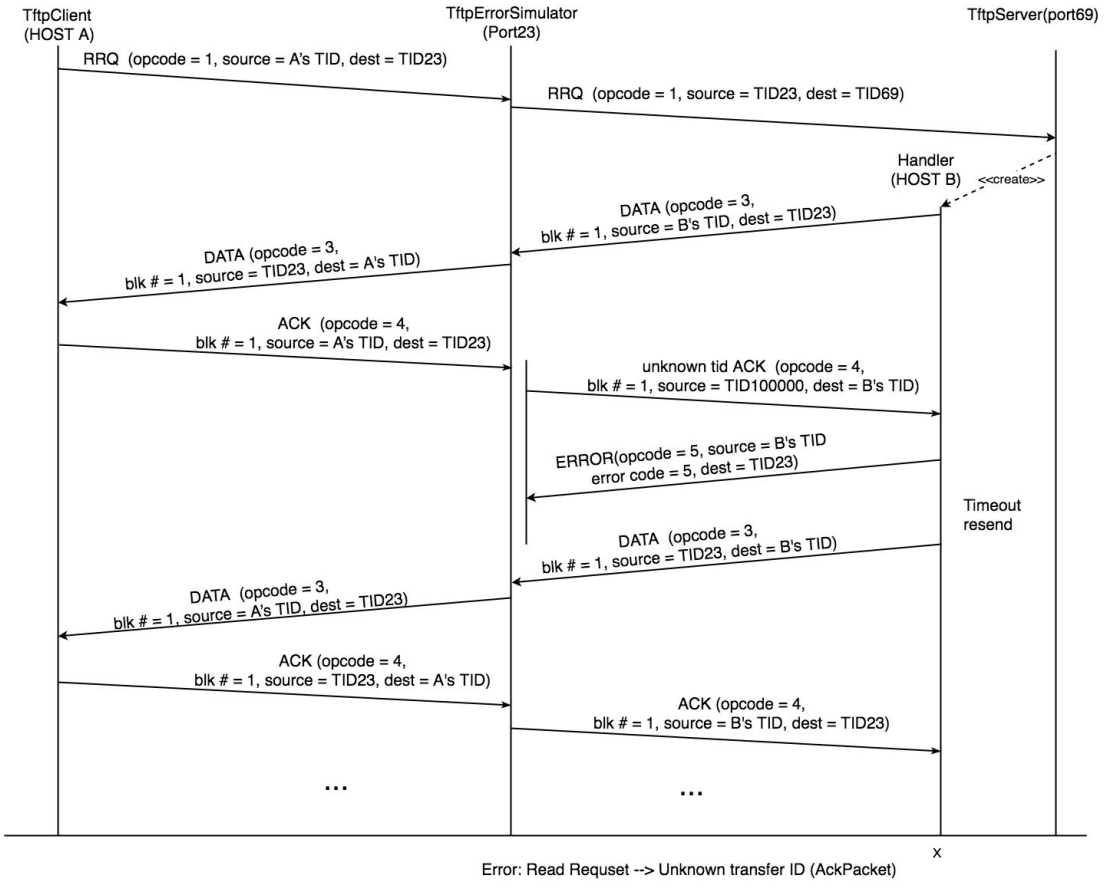

Figure 3.21: Unknown Ackpacket transfer ID During Read Request

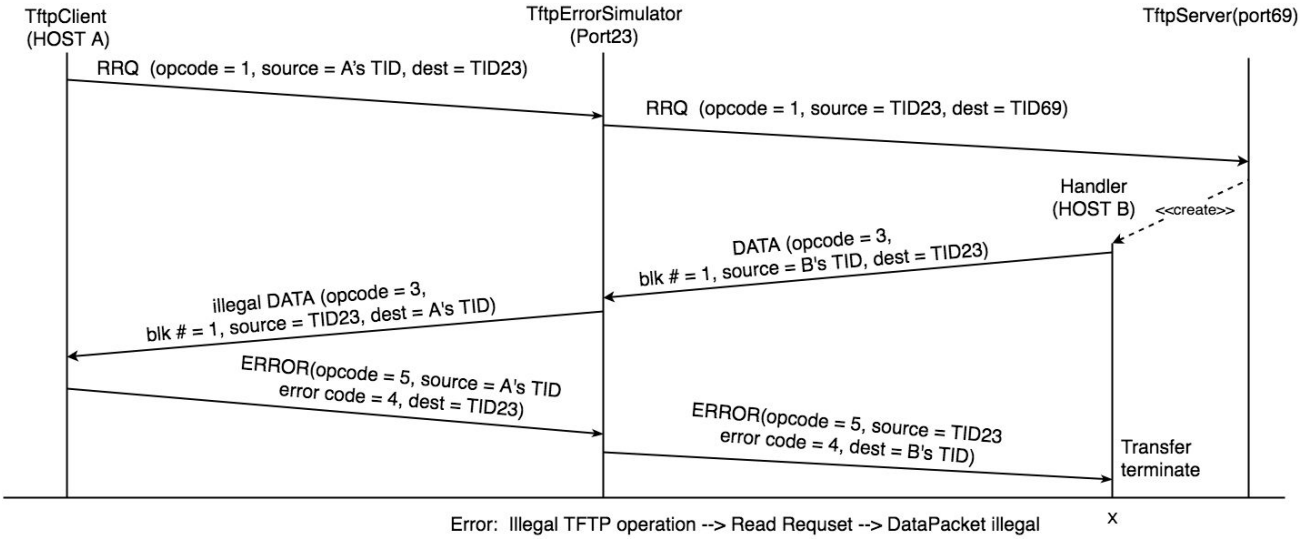

Figure 3.22: Datapacket Illegal During Read Request

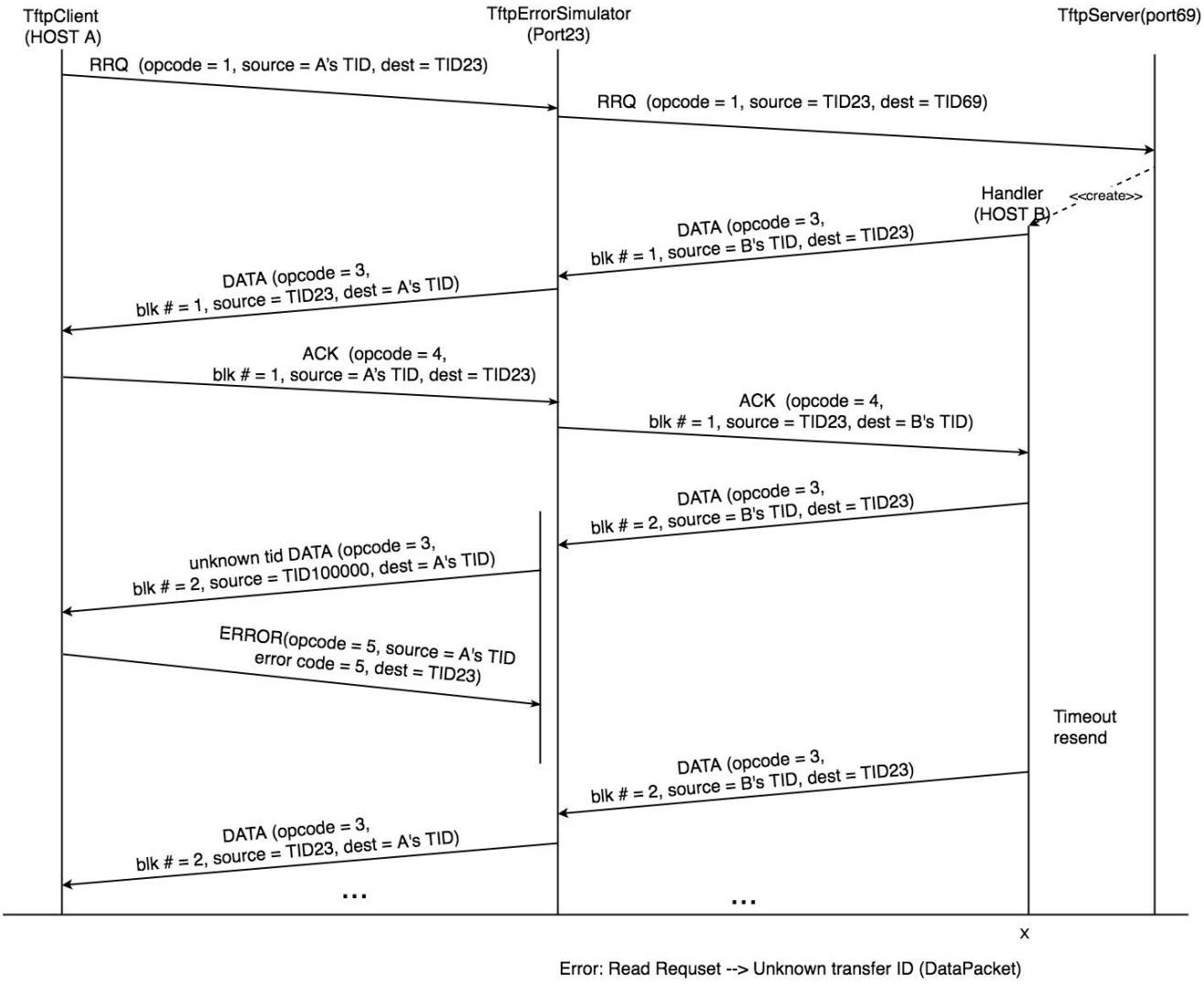

Figure 3.23: Unknown Datapacket transfer ID During Read Request

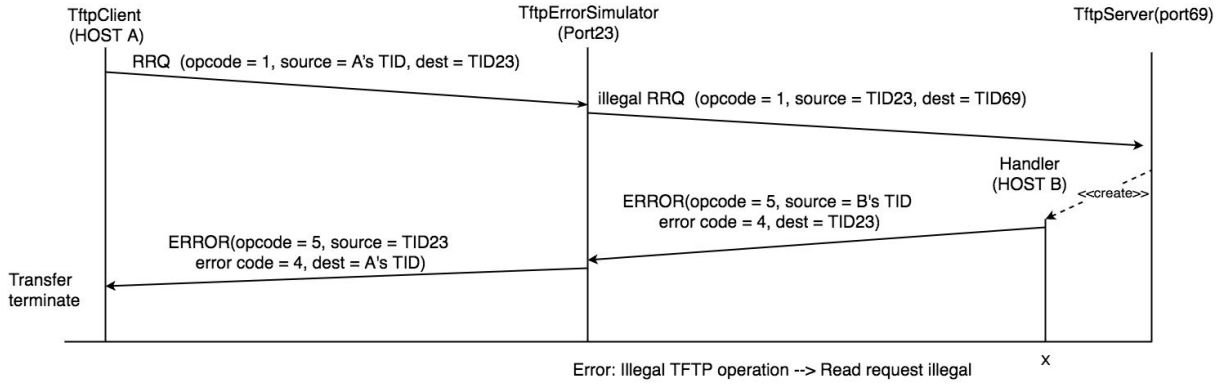

Figure 3.24: Requestpacket Illegal During Read Request

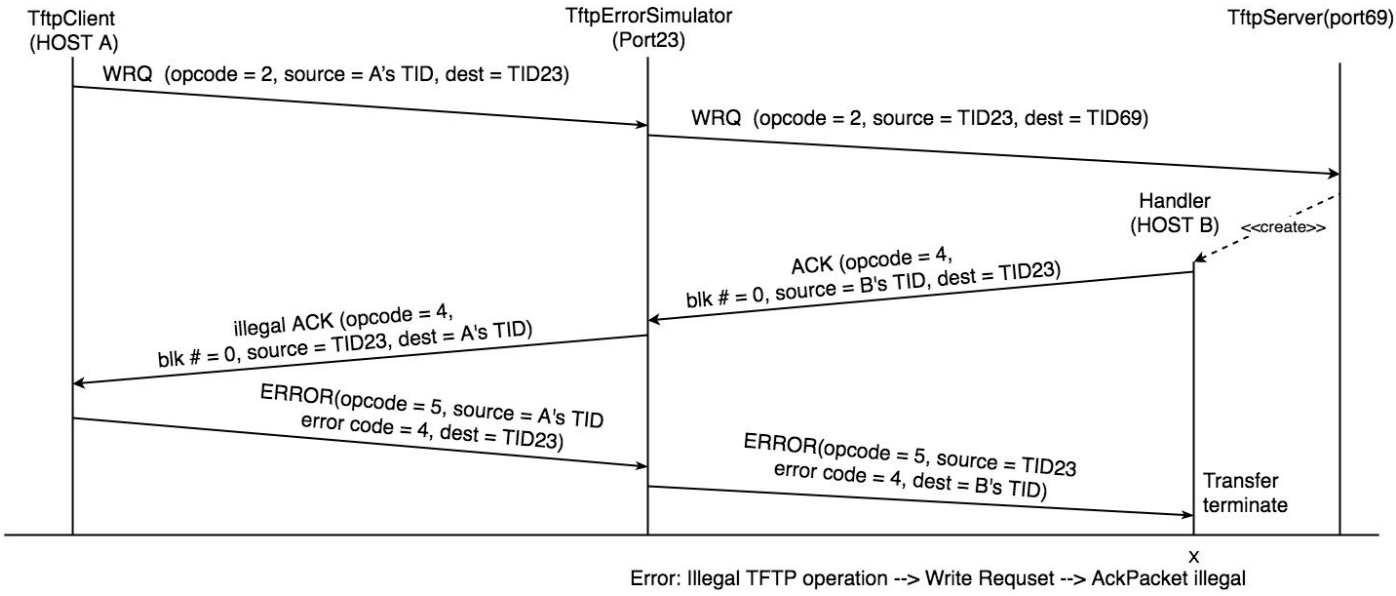

Figure 3.25: Ackpacket Illegal During Write Request

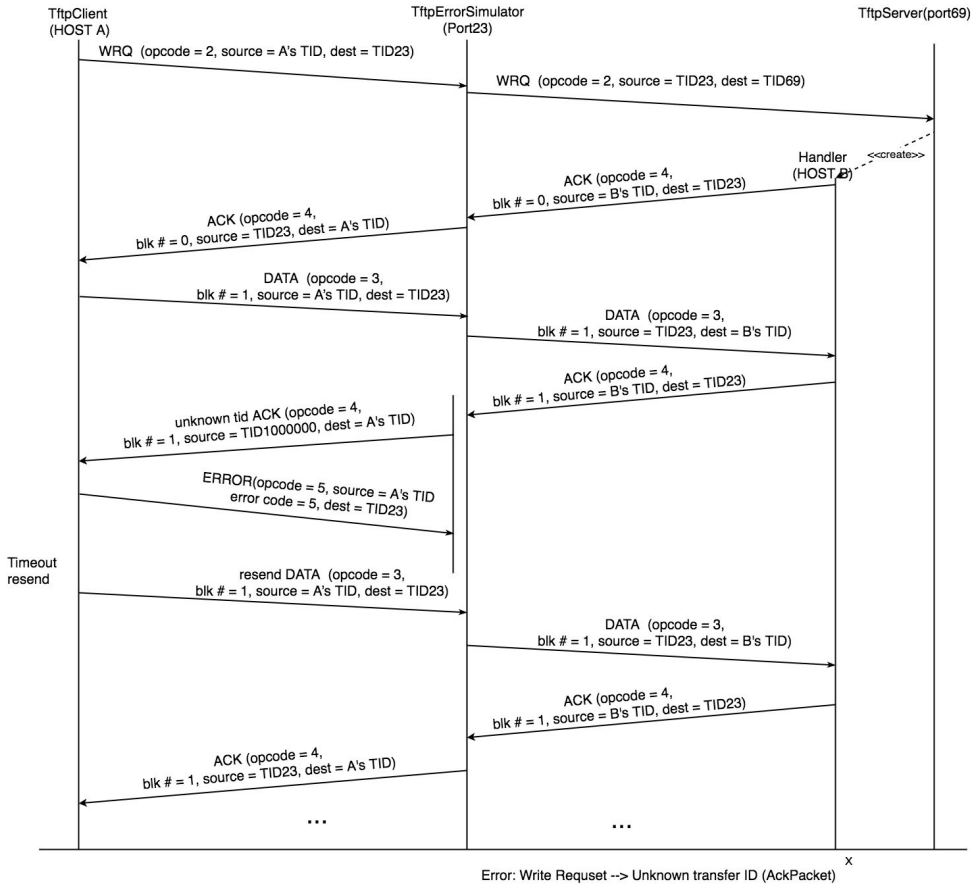

Figure 3.26: Unknown Ackpacket transfer ID During Write Request

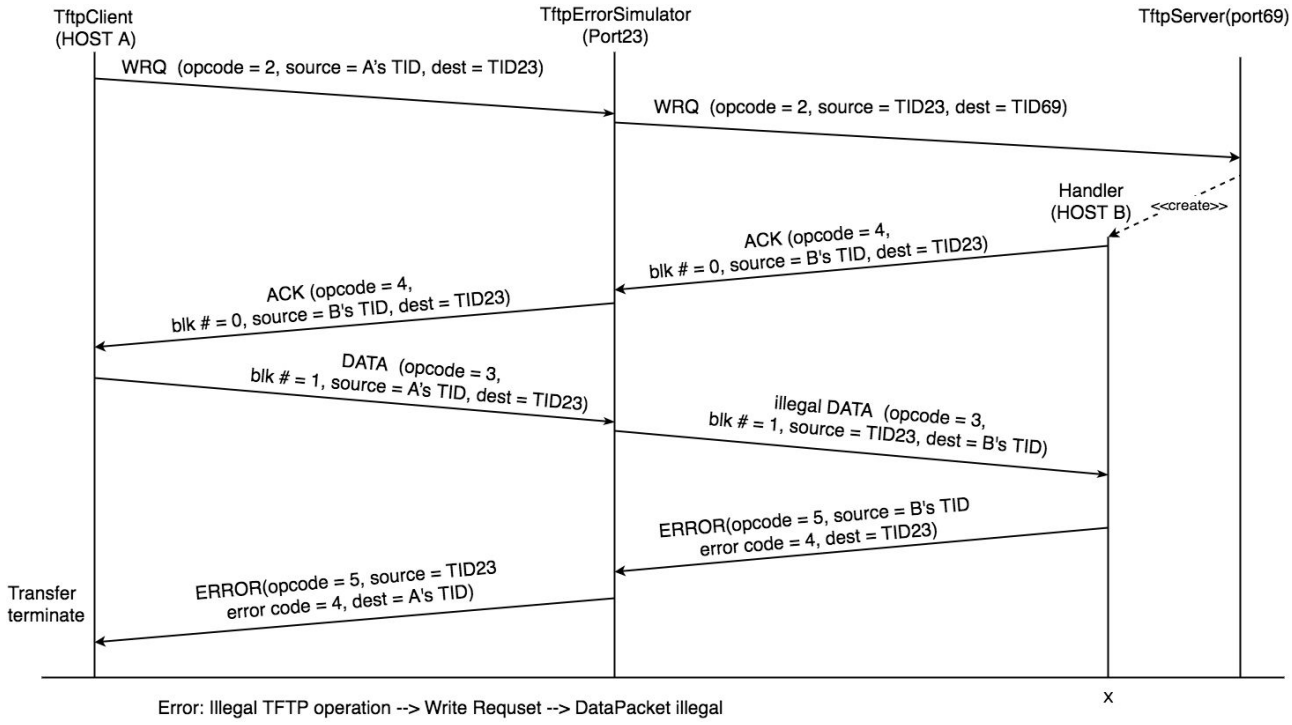

Figure 3.27: Datapacket Illegal During Write Request

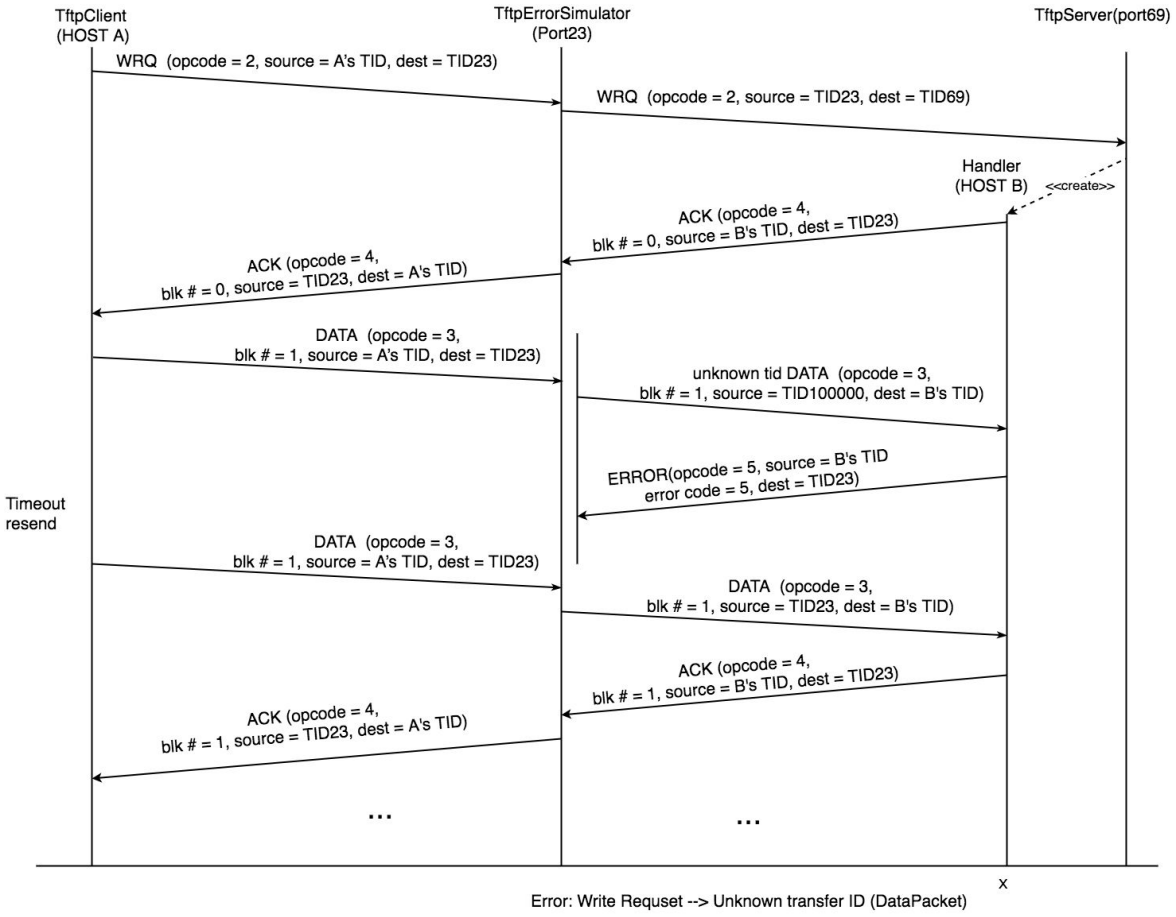

Figure 3.28: Unknown Datapacket transfer ID During Write Request

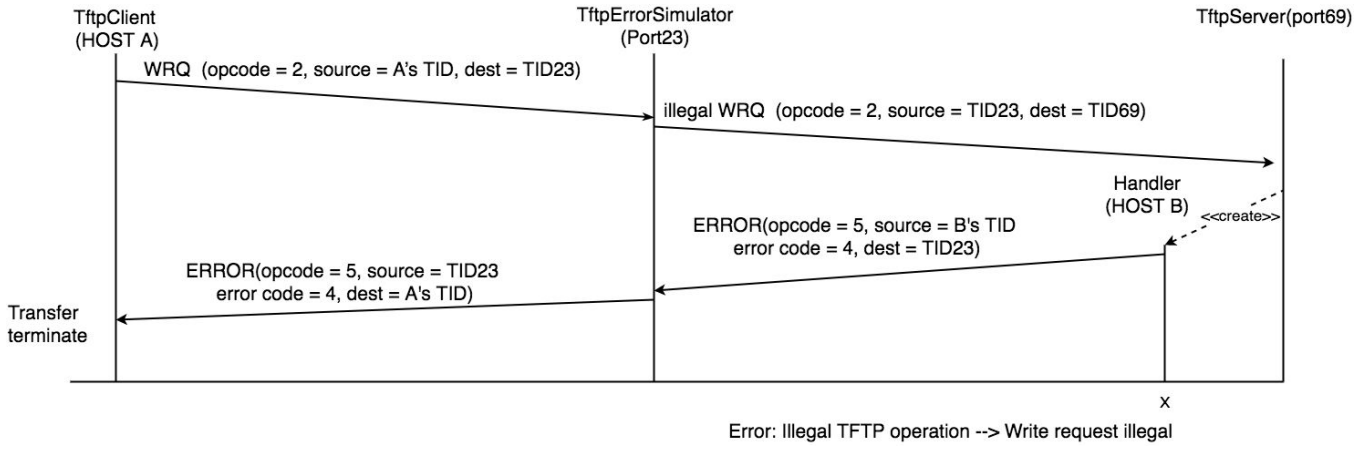

Figure 3.27: Requestpacket Illegal During Write Request