Iteration Test Plan

This document contains the test plan used to ensure compliance with the TFTP standard (RFC 1350). A description of the command line arguments and CLI commands available to each component is also included. Test plans from all iterations completed so far are included.

Description of Command Line Arguments and CLI commands

Client command line arguments

-t Enable test mode (transfers pass through the error simulator)
-q Enable quiet logging mode

Client CLI commands

help Prints this message
cd <directory> Change the directory that files will be written to or read from in the client
read <file> Reads a file from a tftp server to the current working directory.
Optionally, the local destination file path may be specified.
write <file> Writes a file from the current working directory to a tftp server
server <address> changes the ip address that the client will send packets to
shutdown Exits the client

Server CLI commands

help Prints this message
cd <directory> Change the directory that files will be written to or read from on the server
shutdown Exits the server

Error Simulator CLI commands

help	Prints a help message
shutdown	Exits the simulator
norm	Forward packets through without alteration
rend	Removes the end byte of the next request packet. Ie Removes the 0 Byte after Mode
rrs	Removes the Request Seperator of the next request packet. ie Removes 0 Byte after Filename
mode <mode>	Changes the mode of the next request packet, <mode> should be a string to use as a replacement
csa <type> <packetNum>	Changes the sender TID of a specified packet
op <type> <packetNum> <opCode>	Changes the opcode of the specified packet
cl <type> <packetNum> <packetLen>	Changes the length of a specified packet

delay <type> <packetNum> <numTimeouts>	Delays the specified packet by a number of timeouts
duplicate <type> <packetNum>	Sends a duplicate of the specified packet
drop <type> <packetNum>	Drops the specified packet

Basic Transfer Tests (Iteration 1)

File Sizes:
0 byte
200 byte
512 byte
2048 byte
100 000 byte

After each transfer the following checks will be performed:
1.Ensure that the MD5 checksum of the file at the source and destination match
2.Attempt to move the file out of the source and destination directories to ensure the file is not in use

Test Set Up

Ensure that the server and client run in different directories. This is the case in Eclipse by default
1. Start the server
2. Start the client
3. Create a file in the server directory corresponding to each of the file sizes listed below

Test Steps

For each file:
1. Read the file using the client.
2. Write the file using the client.

Concurrent Connections Tests

The following will be repeated for both read and write
1. Start the server
2. Start one instance of the client
3. Start a second instance of the client
4. Start the transfer of a 1 MB file using the first client instance
5. Start the transfer of a small (200 byte) file using the second client instance while the first client is still running the transfer
6. Ensure that both transfers run to completion

Independent Implementation Tests

The purpose of these tests is to verify that the TFTP specification is followed by reading and writing files using an independent TFTP client and server

Server

1. Using an independent TFTP client, read a 2048 byte file

2. Using an independent TFTP client, write a 2048 byte file

Client

1. Using an independent TFTP server, read a 2048 byte file
2. Using an independent TFTP server, write a 2048 byte file

Iteration 2

1. Start the server
2. Start the error simulator
3. Start the client, providing it the -t command line option
4. Use the client to write a 2048 byte file
5. Use the client to read a 2048 byte file
6. Start the transfer of a file with the 'rend' command enabled
7. Start the transfer of a file with the 'norm' command enabled
8. Start the transfer of a file with the 'rrs' command enabled
9. For transfers with the 'mode' command enabled
    1. Set the mode to '.'
    2. Set the mode to 'netascii'
10. For transfers the 'csa' command enabled, set the packet number to:
    1. 0
    2. n = 1
    2. n < 1
    3. n < 0
    4. n = 65536

11. For transfers with the 'op' command enabled:
    1. Repeat packet number tests from step 9.
       Set the op code to:
    2. 255
    3. 266
    4. 32728
    5. -32727
    6. 32729
    7. -32728
12. For transfers with the 'cl' flag enabled:
    1. Repeat the packet number tests from step 9.
    Set the packet length to:
    2. 0
    3. length < 0
    4. length > 0

Perform the Concurrent Connections test with the –t option passed to both client

Iteration 3

Case 1 - Lose a Packet

All of the following steps will be run using the 'drop <type> <packetNum>' command.

1. Drop the first ack packet
2. Drop the last ack packet
3. Drop an ack packet in the middle of a transfer
4. Drop the first data packet
5. Drop the last data packet
6. Drop a data packet in the middle of a transfer
7. Drop the request packet

Case 2 - Delay a Packet

All of the following steps will be run using the 'delay <type> <packetNum> <numOfTimeouts>' command. For each step, repeat it using a timeout of less than 5 and a timeout of more than 5.

1. Delay the first ack packet
2. Delay the last ack packet
3. Delay an ack packet in the middle of a transfer
4. Delay the first data packet
5. Delay the last data packet
6. Delay a data packet in the middle of a transfer
7. Delay the request packet

Case 3 - Duplicate a Packet

All of the following steps will be run using the 'duplicate <type> <packetNum>' command.

1. Duplicate the first ack packet
2. Duplicate the last ack packet
3. Duplicate an ack packet in the middle of a transfer
4. Duplicate the first data packet
5. Duplicate the last data packet
6. Duplicate a data packet in the middle of a transfer
7. Duplicate the request packet

Iteration 4

For each of the following test cases, the client and server *cd* commands may be used to specify the source and destination directory of both the client and the server. The command "cd ." may be used to return the source and destination directory to the current working directory of the program.

Error Code 1 Scenarios - File not Found

Case 1 - WRQ File not present in client directory

8. Attempt to write a file from the client which does not exist in the client's working directory
9. Ensure that the client CLI displays a message informing the user that the file could not be found
10. Ensure that the server did not receive a WRQ from the client

Case 2 - RRQ File not present in server directory

1. Attempt to read a file which does not exist in the server's working directory
2. Ensure that the client CLI displays an error message informing the user that the file could not be found
3. Check the client logs to ensure that an Error packet with code 1 was received by the client

Error Code 2 Scenarios - Access Violation

Case 3 - WRQ Write to a read-only folder on server

1. Set up a directory which the current user does not have permissions to write to
2. Use the server *cd* command to change to the directory created in step 1.
3. Start a write transfer from the client.
4. Check the server log to ensure that the transfer has stopped.
5. Ensure that the client CLI displays an appropriate error message (Access violation)
6. Check the client logs to ensure that an Error packet with code 2 was received by the client

Case 4 - WRQ Write from a client directory without read permission

1. Set up a directory which the current user does not have permissions to read from
2. Use the client *cd* command to change to the directory created in step 1
3. Start a read transfer from the client. In the CLI command to start this transfer, specify a filename in the directory from step 3 as the source file name.
4. Check the server log to ensure that the server did not receive a WRQ packet
5. Ensure that the client CLI displays an appropriate error message (Access violation)

Case 5 - RRQ Read from server directory without read permission

1. Set up a directory which the current user does not have permissions to read from
2. Use the server *cd* command to change to the directory created in step 1.
3. Start a read transfer from the client.
4. Ensure that the transfer terminates on the server side
5. Ensure that the client displays an appropriate error message (Access violation)
6. Check the client log to ensure that an Error packet with code 2 was received by the client

Case 6 - RRQ Read to client directory without write permission

1. Set up a directory which the current user does not have permissions to write to
2. Use the client *cd* command to switch to the directory created in step 1.

3. Start a read transfer from the client.
4. Ensure that the client displays an appropriate error message (Access violation)
5. Check the server log to ensure that the server did not receive a RRQ packet

Error Code 3 Scenarios - Disk full or allocation exceeded

Each of the following scenarios require the use of a disk which has very little space remaining. Such a disk can be created quickly using a small USB stick. Use fsutil to create a large empty file to occupy most remaining space.

Case 7 - WRQ disk full

1. Use the server's *cd* command to set the output directory to a disk which is full
2. Start a write transfer from the client
3. Check the server log to ensure that the transfer has terminated with an appropriate error message
4. Ensure that an appropriate error message is displayed by the client
5. Check the client logs to ensure that an Error packet with code 3 was received by the client
6. Ensure that no empty or incomplete file exists in the server's output directory

Case 8 - RRQ disk full

1. Use the client *cd* command to change the output directory to a disk which is full
2. Start a read transfer.
3. Ensure that the transfer on the client side has terminated with an appropriate error message
4. Ensure that the transfer on the server side has terminated with an appropriate error message
5. Check the server logs to ensure that an Error packet with code 3 was received
6. Check the destination path of the Client to ensure that no empty or incomplete file exists there

Error Code 6 Scenarios - File already exists

Case 9 - WRQ file already exists on server

1. Start a write transfer. Specify a file which already exists in the server's output directory.
2. Ensure that the transfer on the server side has terminated with an appropriate error message
3. Ensure that the transfer on the client side has terminated with an appropriate error message
4. Check the client log to ensure that the client received an error message with code 6

Case 10 - RRQ file already exists on client

1. Start a read transfer. Specify a file which already exists in the client's working directory.
2. Check the server log to ensure that no RRQ was received by the server
3. Ensure that the transfer on the client side has terminated with an appropriate error message.

Iteration 5

Each of the following scenarios require that the client and server be on separate computers and that the error simulator is on the same computer as the server. All of the following use the *server* <*address*> command

Case 1- no error simulator

1. Start the server
2. Start the client
3. Use the *server* command and specify the server's ip address
4. Start a transfer

Case 2- with error simulator

1. Start the server
2. Start the errorSimulator
3. Start the client with –t flag
4. Use the *server* command and specify the server's ip address
5. Start a transfer

Case 3- concurrent transfers

1. Start the server
2. Start the errorSimulator
3. Start a client with –t flag
4. Star a client with no –t flag
4. Use the *server* command and specify the server's ip address
5. Start a transfer from each client with files large enough that the transfers run concurrently