

NSCOM01

Balcueva, Escalona, Fadrigo, Fortiz

Project Rationale

To create a TFTP Client program that complies with RFC documents: 1350, 2347, 2358, 2349

Program Specifications

1. Programming Language: Java
2. Interface: GUI

Features Implemented

1. TFTP protocol-based features
 - a. Uploading and downloading files (at unlimited file sizes)
 - b. Error detection and handling
 - c. Blocksize modification
 - d. Options recognition and compliance (blksize and tsize only)
2. Non-TFTP protocol-based features
 - a. Internal timeouts (3 seconds)
 - b. Network verification before transmission

Limitations

1. Does not use the official TFTP timeout option
2. Does not verify ACKs (for both send and receive) due to byte conversion limitations

TFTP Packet Assembly: According to RFC 1350

- REQUEST (READ/WRITE, W/O OPTVALS)

REQUEST (READ/WRITE, W/O OPTVALS)					
Length	2 Bytes		Length of String	1 Byte	Length of String
Segment	PADDING 0	Type 01/02	Filename (assuming without \0)	PADDING 0	Mode
					PADDING 0

- REQUEST (READ/WRITE, W OPTVALS)

REQUEST (READ/WRITE, W OPTVALS)									
Length	2 Bytes		Length of String	1 Byte	Length of String	1 Byte	Length of String	Length of String	...
Segment	PADDING 0	Type 01/02	Filename (assuming without \0)	PADDING 0	Mode	PADDING 0	Opt1 (assuming w/\0)	Val1 (assuming w/\0)	...
							OptN (assuming w/\0)	ValN (assuming w/\0)	

- DATA

DATA			
Length	2 Bytes		Length of Byte[]
Segment	PADDING 0	Type 01/02	Byte[] Data

- ACK

ACK				
Length	2 Bytes		2 Bytes	
Segment	PADDING 0	Type 01/02	PADDING 0	Block #

TFTP Packet Assembly: According to RFC 1350

- OACK

OACK						
Length	2 Bytes		Length of String	Length of String	...	Length of String
Segment	PADDING 0	Type 01/02	Opt1 (assuming w/ \0)	Val1 (assuming w/ \0)	...	OptN (assuming w/ \0)

- ERROR PACKET

ERROR PACKET					
Length	2 Bytes		2 Bytes		1 Byte
Segment	PADDING 0	Type 01/02	PADDING 0	Error Message (assuming w/ \0)	PADDING 0

TFTP Packet Assembly: Results and Reference

Error Packet

```
Error Packet
System:
System Hex from Processed Byte: 0005000146696c65206e6f7420666f7756640000
System Bits: 00000000 00000101 00000000 00000001 01000110 01101001 01101100 01100101 00100
000 01101110 01101111 01101000 00100000 01100110 01101111 01101010 01101110 01100100 00000
000 00000000
Wireshark:
Wireshark Hex Raw: 0005000146696c65206e6f7420666f7756640000
Wireshark Bits: 00000000 00000101 00000000 00000001 01000110 01101001 01101100 01100101 00
100000 01101110 01101111 01101000 00100000 01100110 01101111 01101010 01101110 01100100 00
000000 00000000
isError: true
Extract Error: 1 = File not found

▼ Trivial File Transfer Protocol
  Opcode: Error Code (5)
  [Destination File: nenechi.png]
  [Read Request in frame 425]
  Error code: File not found (1)
  Error message: File not found
  > [Expert Info (Warning/Response): TFTP ERROR packet]
```

0000	10 63 c8 5f 57 11 30 9c	23 63 6f c3 08 00 45 00	-c-_W-0- #co...E-
0010	00 30 62 93 00 00 80 11	24 e0 c0 a8 18 fd c0 a8	-0b-.... \$.....
0020	18 fc f1 3a c4 26 00 1c	56 13 00 05 00 01 46 69:&.. V....Fi
0030	6c 65 20 6e 6f 74 20 66	6f 75 6e 64 00 00	le not f ound..

Data Packet

```
Data Packet
System:
System Hex from Processed Byte: 030168656c6c6f20776f726c64
System Bits: 00000011 00000001 01101000 01100101 01101100 01101100 01101111 00100000 01110
111 01101111 01110010 01101100 01100100
Wireshark:
Wireshark Hex Raw: 0003000168656c6c6f20776f726c64
Wireshark Bits: 00000000 00000011 00000000 00000001 01101000 01100101 01101100 01101100 01
101111 00100000 01110111 01101111 01110010 01101100 01100100
getOpCode: 3
(2022/06/16 10:13:15) TFTP.extractData(): 01101000 01100101 01101100 01101100 01101111 001
00000 01110111 01101111 01100100 01101100 01100100
Extract Data: 01101000 01100101 01101100 01101100 01101111 00100000 01110111 01101111 0111
0010 01101100 01100100

▼ Trivial File Transfer Protocol
  Opcode: Data Packet (3)
  [Destination File: abc.txt]
  [Read Request in frame 97]
  Block: 1
  [Full Block Number: 1]
  ▼ Data (11 bytes)
    Data: 68 65 6c 6c 6f 20 77 6f 72 6c 64
    [Length: 11]
```

0000	10 63 c8 5f 57 11 30 9c	23 63 6f c3 08 00 45 00	-c-_W-0- #co...E-
0010	00 2b 63 39 00 00 80 11	24 3f c0 a8 18 fd c0 a8	..+c9.... \$?.....
0020	18 fc f4 d3 c3 bc 00 17	02 13 00 03 00 01 68 65he
0030	6c 6c 6f 20 77 6f 72 6c	64 00 00 00	llo worl d...

TFTP Packet Assembly: Results and Reference

ACK Packet

```
ACK Packet
System:
System Hex from Processed Byte: 04054
System Bits: 00000100 00000101
Wireshark:
Wireshark Hex Raw: 00040054
Wireshark Bits: 00000000 00000100 00000000 01010100
isACK: true
extractACK: Block 84
```

```
▼ Trivial File Transfer Protocol
  Opcode: Acknowledgement (4)
  [Destination File: tote_tilt.jpg]
  [Write Request in frame 563]
  Block: 84
  [Full Block Number: 84]
```

0000	10 63 c8 5f 57 11 30 9c	23 63 6f c3 08 00 45 00	·c·_W·0· #co···E·
0010	00 20 62 e9 00 00 80 11	24 9a c0 a8 18 fd c0 a8	· b·.... \$·.....
0020	18 fc c1 1a df 53 00 0c	ab c5 00 04 00 54 00 00S·T·
0030	00 00 00 00 00 00 00 00	00 00 00 00

OACK Packet

```
OACK Packet
System:
System Hex from Processed Byte: 067473697a6500383139363700
System Bits: 00000110 01110100 01110011 01101001 01111010 01100101 00000011 10000011 00010
011 10010011 01100011 01110000
Wireshark:
Wireshark Hex Raw: 00067473697a6500383139363700
Wireshark Bits: 00000000 00000110 01110100 01110011 01101001 01111010 01100101 00000000 00
111000 00110001 00111001 00110110 00110111 00000000
extractOACK: {tsize}, {81967}
```

```
▼ Trivial File Transfer Protocol
  Opcode: Option Acknowledgement (6)
  [Destination File: tote_tilt.jpg]
  [Write Request in frame 563]
  > Option: tsize = 81967
```

0000	10 63 c8 5f 57 11 30 9c	23 63 6f c3 08 00 45 00	·c·_W·0· #co···E·
0010	00 20 62 e9 00 00 80 11	24 e4 c0 a8 18 fd c0 a8	·*b·.... \$·.....
0020	18 fc c1 1a df 53 00 0c	c0 ad 00 06 74 73 69 7aS·tsiz
0030	65 00 38 31 39 36 37 00	00 00 00 00	e·81967·

TFTP Packet Assembly: Results and Reference

Read Request (With and Without OptsVals)

```
RRQ Packet
System:
System Hex from Processed Byte: 016e656e656368692e706e6706f6374657407473697a650300
System Bits: 00000001 01101110 01100101 01101110 01100101 01100011 01101000 01101001 00101
110 01110000 01101110 01100111 00000110 11110110 00110111 01000110 01010111 01000000 01110
100 01110011 01101001 01111010 01100101 00000011 00000000
Wireshark:
Wireshark Hex Raw: 00016e656e656368692e706e6706f6374657407473697a650300
Wireshark Bits: 00000000 00000001 01101110 01100101 01101110 01100101 01100011 01101000 01
101001 00101110 01110000 01101110 01100111 00000000 01101111 01100011 01110100 01100101 01
110100 00000000 01110100 01110011 01101001 01111010 01100101 00000000 00110000 00000000
RQHasOACK: true
extractOACKFromRQ: {tsize}, {}
```

```
RRQ Packet Without Opts & Vals
System:
System Hex from Processed Bytes: 016e656e656368692e706e6706f6374657400
System Bits: 00000001 01101110 01100101 01101110 01100101 01100011 01101000 01101001 00101
110 01110000 01101110 01100111 00000110 11110110 00110111 01000110 01010111 01000000
Wireshark:
Wireshark Hex Raw: 00016e656e656368692e706e6706f6374657400
RQHasOACK: false
Wireshark Bits: 00000000 00000001 01101110 01100101 01101110 01100101 01100011 01101000 01
101001 00101110 01110000 01101110 01100111 00000000 01101111 01100011 01110100 01100101 01
110100 00000000
```

Trivial File Transfer Protocol			
Opcode: Read Request (1)			
Source File: nenechi.png			
Type: octet			
> Option: tsize = 0			
0000	30 9c 23 63 6f c3 10 63	c8 5f 57 11 08 00 45 00	0-#co...c..._W...E-
0010	00 38 12 e1 00 00 80 11	74 8a c0 a8 18 fc c0 a8	-8-.....t-.....
0020	18 fd c4 26 00 45 00 24	3c 67 00 01 6e 65 6e 65	...&.E.\$<g..nene
0030	63 68 69 2e 70 6e 67 00	6f 63 74 65 74 00 74 73	chi.png- octet-tsize-0-
0040	69 7a 65 00 30 00		

TFTP Packet Assembly: Results and Reference

Write Request (With and Without OptsVals)

```
WRQ Packet
System:
System Hex from Processed Byte: 02746573742e706e6706f6374657407473697a650038313936370
System Bits: 00000010 0110100 01100101 0110011 0110100 00101110 0110000 01101110 01100
111 00000110 11110110 00110111 01000110 01010111 01000000 0110100 01110011 01101001 01111
010 01100101 00000011 10000011 00010011 10010011 01100011 01110000
Wireshark:
Wireshark Hex Raw: 0002746f74655f74696c742e6a7067006f6374657407473697a6500383139363700
RQHasOACK: true
extractOACKFromRQ: {tsize}, {81967}
Wireshark Bits: 00000000 00000010 0110100 01101111 0110100 01100101 01011111 0110100 01
101001 01101100 0110100 00101110 01101010 01110000 01100111 00000000 01101111 01100011 01
110100 01100101 0110100 00000000 0110100 01110011 01101001 01111010 01100101 00000000 00
111000 00110001 00111001 00110110 00110111 00000000
```

```
WRQ Packet Without Opts & Vals
System:
System Hex from Processed Byte: 02746573742e706e6706f637465740
System Bits: 00000010 0110100 01100101 0110011 0110100 00101110 01110000 01101110 01100
111 00000110 11110110 00110111 01000110 01010111 01000000
Wireshark:
Wireshark Hex Raw: 0002746f74655f74696c742e6a7067006f6374657400
RQHasOACK: false
Wireshark Bits: 00000000 00000010 0110100 01101111 0110100 01100101 01011111 0110100 01
101001 01101100 0110100 00101110 01101010 01110000 01100111 00000000 01101111 01100011 01
110100 01100101 0110100 00000000
```

Trivial File Transfer Protocol			
Opcode: Write Request (2)			
Destination File: tote_tilt.jpg			
Type: octet			
Option: tsize = 81967			
0000	30 9c 23 63 6f c3 10 63	c8 5f 57 11 08 00 45 00	0-#co...c..._W...E-
0010	00 3e 12 e5 00 00 80 11	74 80 c0 a8 18 fc c0 a8	->.....t.....
0020	18 fd df 53 00 45 00 2a	32 41 00 02 74 6f 74 65	...S-E.* 2A..tote
0030	5f 74 69 6c 74 2e 6a 70	67 00 6f 63 74 65 74 00	_tilt.jp g:octet
0040	74 73 69 7a 65 00 38 31	39 36 37 00	tsize=81 967-

Network Sequence

Send

```
1 public boolean send(File f, String[] opts, String[] vals) {
2     boolean state = false;
3     if(f == null)
4         return state;
5     openConnection();
6     if(f.exists() && socket.isConnected())
7         if(askWritePermission(f, opts, vals))
8             state = writeToServer(f, opts, vals);
9     closeConnection();
10    reset();
11    return state;
12 }
```

Receive

```
1 public File receive(String filename, String saveAs, String[] opts, String[] vals) {
2     if(filename == null)
3         return null;
4     File tempFile = new File(saveAs); //To save on a temp folder of the program.
5     int tsize = askReadPermission(filename, opts, vals);
6     if(tsize > -1) {
7         openConnection();
8         tempFile = readFromServer(filename, tempFile, opts, vals);
9         closeConnection();
10    }
11    reset();
12    return tempFile;
13 }
```

GUI Layout

The screenshot displays a graphical user interface titled "NSCOM01 - TFTP". The interface is organized into several sections:

- Server Information:** Located on the left, it includes three input fields labeled "Server IP:", "Server Port:", and "Block Size:". The "Block Size:" field has a dropdown menu currently set to "Default".
- File Selection:** On the right, there are two text boxes. The top one is labeled "Locally Selected File:" and contains the text "No File Selected". The bottom one is labeled "Remote Selected File:" and is currently empty.
- Actions:** Below the file selection fields are two buttons: "Send File" and "Receive File".
- Console:** A large rectangular area on the right is labeled "Console:". Above it is a small "Window Size" button. The console area is currently blank.
- Navigation:** On the left side, below the server information, are three buttons: "Ping Server", "Open File", and "About".
- System Controls:** At the bottom center are two buttons: "Reset" and "Exit".