



# Computer Networks Coursework - 231006

## Source Code Design and Protocol Description

To implement the protocol correctly, I began by creating an encoder and a decoder for the TFTP packets, this way I was able to reliably encode and decode information into TFTP packets. These files could be used by both my TFTP client and my TFTP server and helped reduce repetition. The two classes responsible for this are `TFTPRequestDecoder` and `TFTPRequestBuilder`. This is where the packet types as described in [RFC 1350](#) are actually implemented.

### Building TFTP Requests

The `TFTPRequestBuilder` class is responsible for encoding request data into a buffer, which can then be sent via either UDP or TCP.

#### Creating RRQ and WRQ Packets

```
// RRQ/WRQ packet
//
// 2 bytes      string     1 byte      string     1 byte
// -----
// | Opcode |   Filename   |   0   |    Mode    |   0   |
// -----
```

The `RRQ` and `WRQ` packets have relatively similar implementation and so we use a helper method which can perform both:

```
private static int packRRQorWRQ(byte[] buf, OPCODE op, String filename)
```

## Creating DATA Packets

```
// DATA packet
// 2 bytes    2 bytes    n bytes
// -----
// | Opcode | Block # | Data   |
// -----
```

```
public static int packData(byte[] buf, int block, byte[] data)
```

## Creating ACK Packets

```
// ACK packet
// 2 bytes    2 bytes
// -----
// | Opcode | Block # | 0   |
// -----
```

```
public static int packAck(byte[] buf, int block)
```

## Creating Error Packets

```
// Error packet
//
// 2 bytes    2 bytes    string    1 byte
// -----
// | Opcode | ErrorCode | ErrMsg   | 0   |
// -----
```

```
public static int packError(byte[] buf, int errorCode, String errorMessage)
```

## Decoding TFTP Requests

In order for our server and client to interact, we also must be able to decode the requests our server/client sends. This is the responsibility of the `TFTPRequestDecoder` class.

Our request decoder is able to unpack data packets that are received as `byte[]` and returns the data needed for that packet in a more understandable form. For example, the `unpackWRQorRRQ` method returns a `WrqOrRrqPacket`. Whilst a data packet is decoded into an instance of the `DataPacket` class.

## Handling TFTP Requests

Using the decoding and encoding techniques described above for TFTP requests, we now can easily unpack or pack and data or request we may need to send or receive from either the client or the server.

Note: Each `RequestHandler` for TFTP Server is assigned to a client based on its IP Address and stores information about the current client and the `DataPacketsBuilder` which is used to store data packets received. On the TFTP Client, the `Client` class stores the `DataPacketsBuilder`

## Handling WRQ Packets

When an `WRQ` is sent to the server or the client, the recipient begins by unpacking the filename of the request and assigning it to the filename of the `DataPacketsBuilder` so the server knows where to save the data packets once the `DATA` blocks have been received. The recipient then sends back an `ACK` block with a block number 0 to

acknowledge the request and tell the sender that it is ready for `DATA` packets to be sent.

## Handling RRQ Packets

When an `RRQ` request is sent to the server or the client, the responder first attempts to find that file in its directory, if that file is not found, an `ERROR` packet is sent to the sender of the request.

If the file does exist, the recipient of the request will immediately begin sending the data packets each of maximum size of 512 bytes (excluding headers). After each `DATA` block is sent, the request recipient will be expected to send an `ACK` packet acknowledging the receipt of the `DATA` packet.

## Handling DATA Packets

When a `DATA` packet is received, the data is unpacked from the packet and is added to the `DataPacketsBuilder` instance on the request handler's class using the `addDataPacket` method. If the size of the unpacked data in the packet is less than 512 bytes, then this tells us we have reached the end of the transfer and should save the file to the current directory.

# Server Implementation

## UDP Server

The UDP version of the server uses the `DatagramSocket` class to receive and send data over a socket. It runs concurrently by storing request handlers in a Hash-map with a key to denote the IP address and port of the client. When a request is received, the server checks if the client has an existing `TFTPRequestHandler` in the Hash-map and if not, they create one. Using the request handler, we then run the `handle` method on it, passing it the packet.

## TCP Server

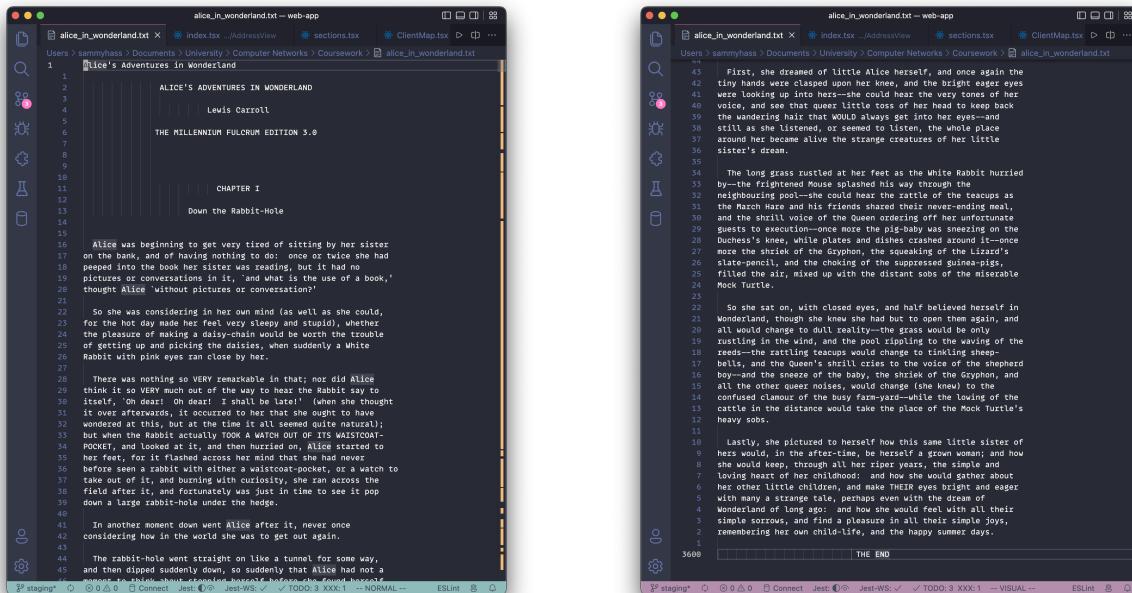
The TCP version of the TFTP server uses a `ServerSocket` to accept requests from a client socket. The clients request is then handled by a `TFTPRequestHandler` thread. This server listens for connections on port 8888.

# Client Implementation

Both clients I have implemented have a simple command line interface with four commands: `upload`, `download`, `help` and `exit`. The command line interface is implemented in the `Cmd` class, whilst the actual requests are sent from the `TFTPClient` class.

## Interoperability

To show that the UDP version of the TFTP Server implements correctly the protocol as described in RFC 1350, I opted to use the built in unix TFTP client - `tftp`. To demonstrate that the server works correctly with a third-party client, I chose to transfer back and forth between the server and client the entire text of the book *Alice in Wonderland* as a text file.



The image shows two side-by-side terminal windows. Both windows have the title bar "alice\_in\_wonderland.txt - web-app". The left window shows the beginning of the file, starting with the title "ALICE'S ADVENTURES IN WONDERLAND" by Lewis Carroll. The right window shows the end of the file, with the last line being "THE END". The file contains approximately 3600 lines of text describing Alice's adventures in Wonderland.

```

alice_in_wonderland.txt - web-app
Users : sammyhass : Documents > University > Computer Networks > Coursework > alice_in_wonderland.txt
1 Alice's Adventures in Wonderland
2
3 ALICE'S ADVENTURES IN WONDERLAND
4
5 Lewis Carroll
6
7 THE MILLENIUM FULCRUM EDITION 3.0
8
9
10
11 CHAPTER I
12
13 Down the Rabbit-Hole
14
15
16 Alice was beginning to get very tired of sitting by her sister
17 on the bank, and of nothing having to do: once or twice she had
18 peeped into the book her sister was reading, but it had no
19 pictures or conversations in it, "and what is the use of a book,"
20 thought Alice "without pictures or conversation?"
21
22 So she was considering in her own mind (as well as she could,
23 for the hot day made her feel very sleepy and stupid), whether
24 the pleasure of making a daisy-chain would be worth the trouble
25 of getting up and picking the daisies, when suddenly a white
26 Rabbit with pink eyes ran close by her.
27
28 There was nothing so VERY remarkable in that; nor did Alice
29 think it so VERY much out of the way to hear the Rabbit say to
30 itself, "Oh dear! Oh dear! I shall be late!" (when she thought
31 it over afterwards, it occurred to her that she ought to have
32 wondered why the Rabbit acted as if it were late (it was nearly
33 time for tea); but then the thought occurred to her, "Well, maybe
34 it's got a pocket-watch!" And then it hurried on.) Alice started to
35 her feet, for it flashed across her mind that she had never
36 before seen a rabbit with either a waistcoat-pocket, or a watch to
37 take out of it; and burning with curiosity, she ran across the
38 field after it, and fortunately was just in time to see it pop
39 down a large rabbit-hole under the hedge.
40
41 In another moment down went Alice after it, never once
42 considering how in the world she was to get out again.
43
44 The rabbit-hole went straight on like a tunnel for some way,
45 and then dipped suddenly down, so suddenly that Alice had not a
46 moment to begin to cry out, "Hello! Hello! Mother dear!" before
47 she found herself falling头下脚上 into a pool of water.
```

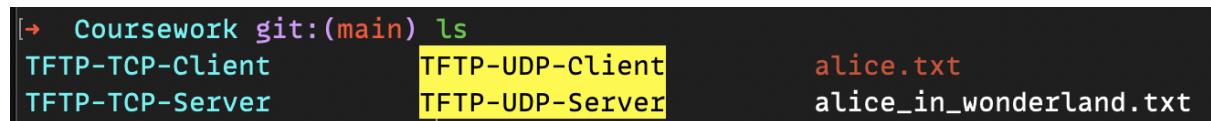
  

```

alice_in_wonderland.txt - web-app
Users : sammyhass : Documents > University > Computer Networks > Coursework > alice_in_wonderland.txt
1
2
3 First, she dreamt of little Alice herself, and once again the
4 tiny hands were clasped upon her knees, and the bright eager eyes
5 were looking up into hers—she could hear the very tones of her
6 voice, and see that queer little toss of her head to keep back
7 the wandering hair that WOULD always get into her eyes—and
8 still as she listened, or seemed to listen, the whole scene
9 of the garden became alive—the strange creatures of her little
10 sister's dream.
11
12 The long grass rustled at her feet as the White Rabbit hurried
13 by, the red hand-mouse scurried his way through the
14 neighbouring pool—she could hear the rattle of the teacups as
15 the March Hare and his friends shared their never-ending meal,
16 and the shrill voice of the Queen ordering off her unfortunate
17 guests to execution—once more the pig-baby was sneezing on the
18 Duchess's knee, while plates and dishes crashed to the ground
19 in the general confusion of the "murder" of the Lizard's
20 slate-pencil, and the choking of the suppressed guinea-pigs,
21 filled the air, mixed up with the distant sobs of the miserable
22 Mock Turtle.
23
24 So she sat on, with closed eyes, and half believed herself in
25 Wonderland, though she knew she had but to open them again, and
26 all would change to dull reality—the grass would be only
27 rustling in the wind, and the pool rippling to the waving of the
28 trees—there would be no noise but the tinkle of sheep
29 bells, and the Queen's shrill cries for the execution of the shorn
30 boy—and the sneeze of the baby, the shriek of the Gryphon, and
31 all the other queer noises, would change (she knew) to the
32 confused clamour of the busy farm-yard—while the lowing of the
33 cattle in the distance would take the place of the Mock Turtle's
34 heavy sobs.
35
36 Lastly, she pictured to herself how this same little sister of
37 hers would, in the after-time, be herself a grown woman; and how
38 she would be as full of thoughts as Alice was now, and how
39 there would be no尽头 to her thoughts; and how she would care about
40 her other little children, and make THEM eyes bright and eager
41 with many a strange tale, perhaps even with the dream of
42 Wonderland of long ago: and how she would feel with all their
43 simple sorrows, and find a pleasure in all their simple joys,
44 remembering her own child-life, and the happy summer days.
```

The start of the file used for testing the TFTP server.

End of the file used for testing the TFTP server, note, the file has 3600 lines exactly.



The image shows a terminal window with the prompt "[→ Coursework git:(main) ls". The directory listing includes files for TFTP clients and servers, and the Alice in Wonderland text file. The file `alice_in_wonderland.txt` is highlighted in yellow.

```

[→ Coursework git:(main) ls
TFTP-TCP-Client          TFTP-UDP-Client
TFTP-TCP-Server          TFTP-UDP-Server
                           alice.txt
                           alice_in_wonderland.txt
```

Directory listing for the coursework root, note, the file `alice_in_wonderland.txt` which contains the entire of the book Alice in Wonderland, as described above.

## Connecting to the Server via TFTP Unix Client

The UDP version of the TFTP Server runs locally on port 8888. To connect to the server using the command line client, we must first launch the `tftp` command line tool and use the `connect` command to connect to `localhost:8888`. In the screenshots below, I also enable trace mode so that we can see the packets received and sent by the client, in addition we must enable `octet` mode on the client side.

```
[→ Coursework git:(main) tftp
[tftp> connect localhost 8888
[tftp> mode octet
[tftp> trace
Packet tracing on.
tftp> |
```

Starting up the `tftp` client, and connecting to a server on `localhost:8888`, enabling octet mode and turning on packet tracing.

## Interoperability - Sending a File to the Server over TFTP with the Unix Client

To send a file from the client to the server, we use the `put` command from within the `tftp` utility.

```
[tftp]> put alice_in_wonderland.txt
sent WRQ <file=alice_in_wonderland.txt, mode=octet>
received ACK <block=0>
sent DATA <block=1, 512 bytes>
received ACK <block=1>
sent DATA <block=2, 512 bytes>
received ACK <block=2>
sent DATA <block=3, 512 bytes>
received ACK <block=3>
sent DATA <block=4, 512 bytes>
received ACK <block=4>
```

Output from `tftp` unix client when sending a write request. As we can see, the WRQ request is sent, and an ACK is received with block no. 0. Then the client starts sending data, waiting for an ACK packet after each block is sent.

```
sent DATA <block=291, 94 bytes>
received ACK <block=291>
Sent 148574 bytes in 0.1 seconds
```

Final line of output from the `tftp` unix client when uploading a file to the server.

Upon examining the output of the server, we can see that the server received the request replied to the client as expected.

```
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received WRQ - Filename: alice_in_wonderland.txt
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent ACK Block 0
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received DATA for alice_in_wonderland.txt - Block no 1
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent ACK Block 1
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received DATA for alice_in_wonderland.txt - Block no 2
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent ACK Block 2
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received DATA for alice_in_wonderland.txt - Block no 3
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent ACK Block 3
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received DATA for alice_in_wonderland.txt - Block no 4
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent ACK Block 4
```

```

TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received DATA for alice_in_wonderland.txt - Block no 290 of size 512 bytes
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent ACK Block 290
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received DATA for alice_in_wonderland.txt - Block no 291 of size 94 bytes
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent ACK Block 291
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received all data from alice_in_wonderland.txt
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Saved file to alice_in_wonderland.txt

```

End of the output from the server. Showing that the number of blocks was identical as was the size of the final packet (meaning that no data was lost)

The server has now saved the file to its own directory, [/TFTP-UDP-Server](#). We can inspect this file and see that it has been transferred correctly with the same number of lines as the original (3600)

```

Alice's Adventures in Wonderland
ALICE'S ADVENTURES IN WONDERLAND
Lewis Carroll
THE MILLENNIUM FULCRUM EDITION 3.0
CHAPTER I
Down the Rabbit-Hole
Alice was beginning to get very tired of sitting by her sister
on the bank, and of having nothing to do: once or twice she had
peeped into the book her sister was reading, but it had no
pictures or conversations in it, 'and what is the use of a book,'
thought Alice 'without pictures or conversation?'
So she was considering in her own mind (as well as she could,
for the hot day made her feel very sleepy and stupid), whether
the pleasure of making a daisy-chain would be worth the trouble
of getting up and picking the daisies, when suddenly a White
Rabbit with pink eyes ran close by her.
There was nothing so VERY remarkable in that; nor did Alice
think it so VERY much out of the way to hear the Rabbit say to
itself, 'Oh dear! Oh dear! I shall be late!' (when she thought
it over afterwards, it occurred to her that she ought to have
wondered at this, but at the time it all seemed quite natural);
but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-
POCKET, and looked at it, and then hurried on, Alice started to
her feet, for it flashed across her mind that she had never

```

Newly created [alice\\_in\\_wonderland.txt](#) file

```

3588 heavy soos.
3589
3590 Lastly, she pictured to herself how this same little sister of
3591 hers would, in the after-time, be herself a grown woman; and how
3592 she would keep, through all her riper years, the simple and
3593 loving heart of her childhood: and how she would gather about
3594 her other little children, and make THEIR eyes bright and eager
3595 with many a strange tale, perhaps even with the dream of
3596 Wonderland of long ago: and how she would feel with all their
3597 simple sorrows, and find a pleasure in all their simple joys,
3598 remembering her own child-life, and the happy summer days.
3599
3600 THE END

```

The end of the file, displaying the same number of lines as the original.

## Interoperability - Receiving a File from the TFTP Server

In order to demonstrate retrieving a file from the server, we will first delete the original `alice_in_wonderland.txt` from the root of the `Coursework` folder. This way we can guarantee that a new file is saved when the request is sent.

```

[→ Coursework git:(main) ✘ rm alice_in_wonderland.txt
[→ Coursework git:(main) ✘ ls
TFTP-TCP-Client TFTP-TCP-Server TFTP-UDP-Client TFTP-UDP-Server alice.txt
→ Coursework git:(main) ✘ |

```

Deleting the original `alice_in_wonderland.txt` file from the coursework folder.

Now from within the same session as we sent the file, we can use the `get` command to retrieve the file from the server.

```

[tftp> get alice_in_wonderland.txt
sent RRQ <file=alice_in_wonderland.txt, mode=octet>
received DATA <block=1, 512 bytes>
sent ACK <block=1>
received DATA <block=2, 512 bytes>
sent ACK <block=2>
received DATA <block=3, 512 bytes>
sent ACK <block=3>
received DATA <block=4, 512 bytes>
sent ACK <block=4>
received DATA <block=5, 512 bytes>
sent ACK <block=5>
received DATA <block=6, 512 bytes>

```

```

received DATA <block=288, 512 bytes>
sent ACK <block=288>
received DATA <block=289, 512 bytes>
sent ACK <block=289>
received DATA <block=290, 512 bytes>
sent ACK <block=290>
received DATA <block=291, 94 bytes>
Received 148574 bytes in 0.1 seconds
tftp> |

```

Output from the unix `tftp` client when sending a get request for `alice_in_wonderland.txt`. Note end of output with final data block number 291 of size 94 bytes, (the same size as the file was when we sent it).

Upon examining the output from our server, we can see the request was fulfilled correctly, the server begins sending DATA blocks immediately and waited for the client to sent an ACK block after each DATA block.

```

TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received RRQ - Filename: alice_in_wonderland.txt
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent DATA block 1/291 for alice_in_wonderland.txt - Block size: 512 bytes
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received ACK Block 1
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent DATA block 2/291 for alice_in_wonderland.txt - Block size: 512 bytes
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received ACK Block 2
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent DATA block 3/291 for alice_in_wonderland.txt - Block size: 512 bytes
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received ACK Block 3
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent DATA block 4/291 for alice_in_wonderland.txt - Block size: 512 bytes
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received ACK Block 4
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent DATA block 5/291 for alice_in_wonderland.txt - Block size: 512 bytes
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received ACK Block 5
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent DATA block 6/291 for alice_in_wonderland.txt - Block size: 512 bytes

```

```

TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent DATA block 291/291 for alice_in_wonderland.txt - Block size: 94 bytes
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Received ACK Block 291
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:49983: Sent all data from alice_in_wonderland.txt

```

As can be seen above, our server sent the file correctly to the client.

We can now look at the root directory for the coursework again, the current working directory for the `tftp` utility and find the `alice_in_wonderland.txt` file as it was originally.

```

→ Coursework git:(main) ✘ ls
TFTP-TCP-Client          TFTP-UDP-Client          alice.txt
TFTP-TCP-Server          TFTP-UDP-Server          alice_in_wonderland.txt

```

When listing the directory, we can see the `alice_in_wonderland.txt` file is back.

```

alice_in_wonderland.txt - web-app
Users > sammyhas3 > Documents > University > Computer Networks > Coursework > alice_in_wonderland.txt
1 ALICE's Adventures in Wonderland
2
3 ALICE'S ADVENTURES IN WONDERLAND
4
5 Lewis Carroll
6
7 THE MILLENNIUM FULCRUM EDITION 3.0.
8
9
10 CHAPTER I
11
12 Down the Rabbit-Hole
13
14
15 Alice was beginning to get very tired of sitting by her sister
16 on the bank, and of having nothing to do: once or twice she had
17 peeped into the book her sister was reading, but it had no
18 pictures or conversations in it, "and what is the use of a book,"
19 thought Alice "without pictures or conversation?"
20
21 So she was considering in her own mind (as well as she could,
22 for the hot day made her feel very sleepy and stupid), whether
23 the pleasure of making a daisy-chain would be worth the trouble
24 of getting up and picking the daisies, when suddenly a white
25 Rabbit with pink eyes ran close by her.
26
27 There was nothing so VERY remarkable in that; nor did Alice
28 think it so VERY much out of the way to hear the Rabbit say to
29 itself, "Oh dear! Oh dear! I shall be late!" (when she thought
30 it over afterwards, it occurred to her that she ought to have
31 known it was late, for the sun had actually gone down by now); but
32 when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-
33 POCKET, and looked at it, and then hurried on, Alice started to
34 her feet, for it flashed across her mind that she had never
35 before seen a rabbit with either a waistcoat-pocket, or a watch to
36 go with it. So she ran across the field after it, and fortunately was just in time to see it pop
37 down a large rabbit-hole under the hedge.
38
39 In another moment down went Alice after it, never once
40 considering how in the world she was to get out again.
41
42 The rabbit-hole went straight on like a tunnel for some way,
43 and then dipped suddenly down, so suddenly that Alice had not a
44 chance to steady herself before she found herself falling
45 down a large rabbit-hole under the hedge.

```

```

alice_in_wonderland.txt - web-app
Users > sammyhas3 > Documents > University > Computer Networks > Coursework > alice_in_wonderland.txt
3600 ━━━━ THE END

```

Start of the new file that's just been retrieved from the server.

End of the new file sent from the server (still with 3600 lines)

## Interoperability - Handling Errors

Our TFTP Server should be able to handle errors where the file is not found by the server upon a read request. We can test this works between the `tftp` utility and the server I have made by sending a get request for a file that doesn't exist on the server.

```
tftp> get this_isnt_here.txt
sent RRQ <file=this_isnt_here.txt, mode=octet>
received ERROR <code=1, msg=File Not Found>
```

Error packet sent when a file isn't found. (`tftp` unix client)

```
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:64818: Received RRQ - Filename: this_isnt_here.txt
TFTP_REQUEST_HANDLER - 0:0:0:0:0:0:0:1:64818: File Not Found
```

Server output when file not found.

As can be seen the, correct error code (`1`) is sent back to the client. This indicates that the error that occurred was a file not found error.