



Linnéuniversitetet

Kalmar Vaxjö

Report

Assignment 3

IDV701



Author: Nedko Nedkov, Ibrahim
Groshar

Semester: Spring 2023

Email1 nn222mx@student.lnu.se

Email2 ig222je@student.lnu.se

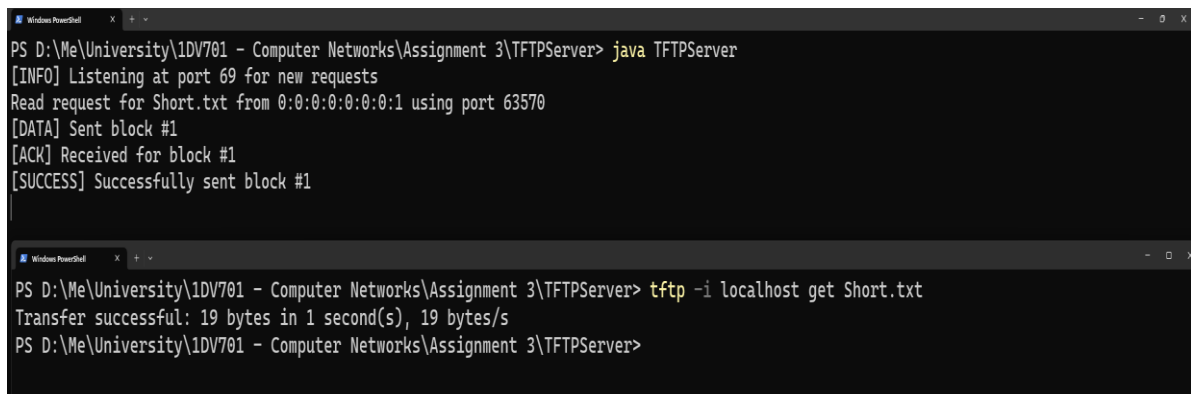
Contents

1 Problem 1 _____ **I**
 1.1 Discussion_____ I

2 Problem 2 _____ **II**
 2.1 Discussion_____ II

3 Problem 3 _____ **III**
 3.1 Discussion_____ VI

1 Problem 1



```
PS D:\Me\University\1DV701 - Computer Networks\Assignment 3\TFTPServer> java TFTPServer
[INFO] Listening at port 69 for new requests
Read request for Short.txt from 0:0:0:0:0:1 using port 63570
[DATA] Sent block #1
[ACK] Received for block #1
[SUCCESS] Successfully sent block #1

PS D:\Me\University\1DV701 - Computer Networks\Assignment 3\TFTPServer> tftp -i localhost get Short.txt
Transfer successful: 19 bytes in 1 second(s), 19 bytes/s
PS D:\Me\University\1DV701 - Computer Networks\Assignment 3\TFTPServer>
```

1.1 Discussion

The **socket** we have is server-sided and the client connects to it by opening a new socket called **sendSocket**. The **socket** has the port of the server (69) and stays all the time. **sendSocket** is initialized everytime a client connects to the server, different ports and is closed after the request has been processed.

The following steps provide the solution for handling read and write requests:

First step to handling a read request is to create a socket which is bound to the server, then establish connection on that socket. Once connection is set receive the first datagram packet and create a `InetSocketAddress` `clientAddress` to be used by the sockets. Then the packet is dissected with its content being a request for a read or write function, the file name and the mode. Continue only if mode is `octet`. Then a new thread is created in which a new socket is created in order to send data through the connection with the `clientAddress`. File path is adjusted according to the type of operations (RRQ/WRQ).

RRQ:

Initializing the read request starts with creating a `fileInputStream` that reads bytes from the file. Then, a data packet is created using the operation `DATA (03)` with block number and data as bytes. A `DatagramPacket` is created to receive the reply from the client. When the data is sent, a reply is received to acknowledge that the data is received. Error handling is crucial here as the acknowledge could be an error or not existing which results in the data being resent or connection is closed. Data can be resend 7 times in total since acknowledgments are sent 8 times when sending next block is late. If attempts exceed 7, the code sends error and return false. Acknowledgments represent the number of the block that was received successfully thus when n number of ack is received by the server, it knows that the previous block was received and sends the next block (n+1 block). In the code, If true is returned, move on to the next block and repeat, otherwise send error and return.

If a datagram packet is sent that has less than 512 bytes of data, data sending is stopped as it is the max amount of data that can be sent in a tftp packet.

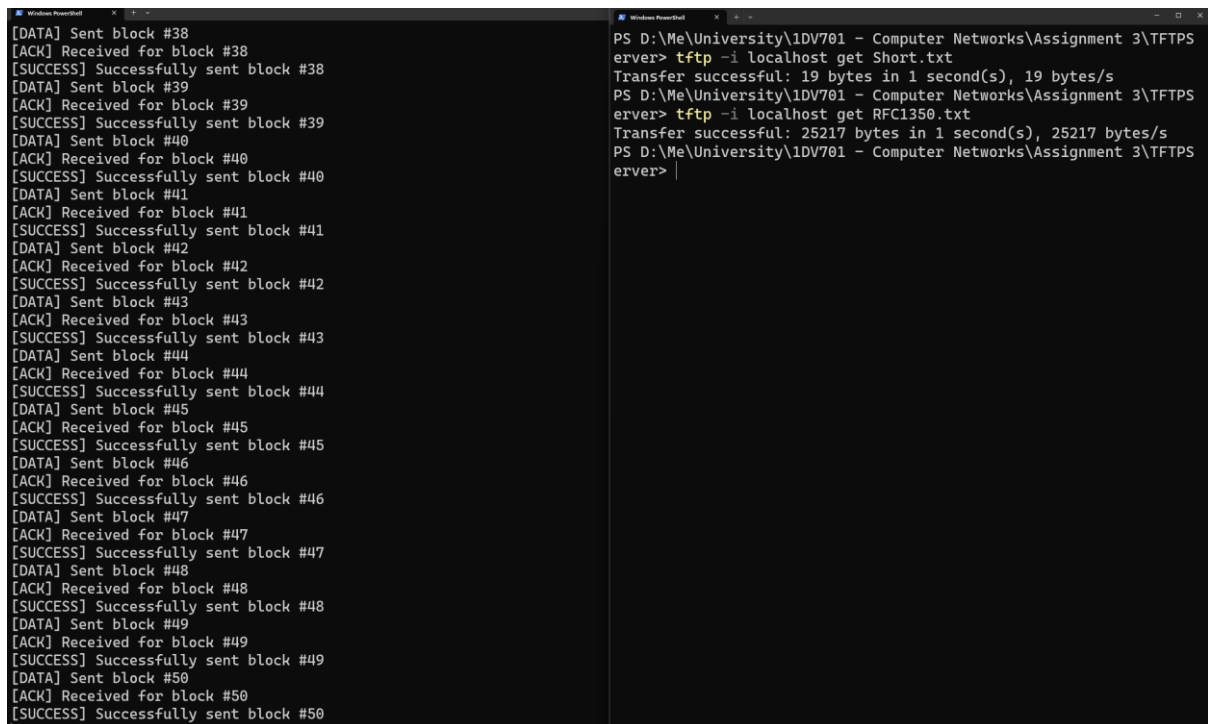
WRQ:

First the file that the user wants to write is validated. Then a `fileOutputStream` is opened to write on to the new file. The first ack that establishes the connection is ack 0, a special case. The next data packet should be the response from the client which is read from the server. A confirmation is done so that the block number of the received packet is that of the acknowledgement last sent + 1. If the block number is one after the last acknowledgement number sent, check if the data is less than 512 bytes, meaning that it is the last one, and if it is, return false, otherwise return true.

If the block number is -1, then an error is sent and it returns false. If the block number is not acknowledgement + 1, it is a duplicated packet, which then gets resent to max number of attempts being 7 with an error before each resend. If the attempts exceed 7, return false, otherwise return true and move on to the next acknowledgement number.

If neither RRQ, nor WRQ, send error and return, which results in closing the `sendSocket` connection.

2 Problem 2



The image shows two side-by-side screenshots of a Windows command prompt window. The left window displays the server's log output, showing a sequence of data blocks being sent and acknowledged. The right window shows the client's commands and the server's responses.

```

[DATA] Sent block #38
[ACK] Received for block #38
[SUCCESS] Successfully sent block #38
[DATA] Sent block #39
[ACK] Received for block #39
[SUCCESS] Successfully sent block #39
[DATA] Sent block #40
[ACK] Received for block #40
[SUCCESS] Successfully sent block #40
[DATA] Sent block #41
[ACK] Received for block #41
[SUCCESS] Successfully sent block #41
[DATA] Sent block #42
[ACK] Received for block #42
[SUCCESS] Successfully sent block #42
[DATA] Sent block #43
[ACK] Received for block #43
[SUCCESS] Successfully sent block #43
[DATA] Sent block #44
[ACK] Received for block #44
[SUCCESS] Successfully sent block #44
[DATA] Sent block #45
[ACK] Received for block #45
[SUCCESS] Successfully sent block #45
[DATA] Sent block #46
[ACK] Received for block #46
[SUCCESS] Successfully sent block #46
[DATA] Sent block #47
[ACK] Received for block #47
[SUCCESS] Successfully sent block #47
[DATA] Sent block #48
[ACK] Received for block #48
[SUCCESS] Successfully sent block #48
[DATA] Sent block #49
[ACK] Received for block #49
[SUCCESS] Successfully sent block #49
[DATA] Sent block #50
[ACK] Received for block #50
[SUCCESS] Successfully sent block #50
  
```

```

PS D:\Me\University\1DV701 - Computer Networks\Assignment 3\TFTPS
erver> tftp -i localhost get Short.txt
Transfer successful: 19 bytes in 1 second(s), 19 bytes/s
PS D:\Me\University\1DV701 - Computer Networks\Assignment 3\TFTPS
erver> tftp -i localhost get RFC1350.txt
Transfer successful: 25217 bytes in 1 second(s), 25217 bytes/s
PS D:\Me\University\1DV701 - Computer Networks\Assignment 3\TFTPS
erver>
  
```

2.1 Discussion

Testing timeouts and retransmissions:

After sending a packet, a timeout is set to (number of attempts) seconds, so each next attempt has more time to wait. If the time runs out, throws an exception, which is caught in the `IOException` catch and an error is sent before continuing with the longer timeout. After each receive execution the timeout is reset to 0.

8 is the upper limit of attempts, as the 9th attempt would cause the while to stop, sending an error and returning false. To test the timeouts, we stopped the program mid-way, so

the client was not receiving an acknowledgment and kept resending the data, which resulted in the server then reading the data packet duplicates and getting a timeout.

3 Problem 3

No.	Time	Source	Destination	Protocol	Length	Info
45	7.855045	::1	::1	TFTP	72	Read Request, File: RFC1350.txt, Transfer type: octet
46	7.855672	::1	::1	TFTP	568	Data Packet, Block: 1
47	7.855788	::1	::1	TFTP	56	Acknowledgement, Block: 1
48	7.892411	::1	::1	TFTP	568	Data Packet, Block: 2
49	7.892469	::1	::1	TFTP	56	Acknowledgement, Block: 2
50	7.892715	::1	::1	TFTP	568	Data Packet, Block: 3
51	7.892746	::1	::1	TFTP	56	Acknowledgement, Block: 3
52	7.892956	::1	::1	TFTP	568	Data Packet, Block: 4
53	7.892993	::1	::1	TFTP	56	Acknowledgement, Block: 4
54	7.893184	::1	::1	TFTP	568	Data Packet, Block: 5
55	7.893221	::1	::1	TFTP	56	Acknowledgement, Block: 5
56	7.893438	::1	::1	TFTP	568	Data Packet, Block: 6
57	7.893465	::1	::1	TFTP	56	Acknowledgement, Block: 6
58	7.893665	::1	::1	TFTP	568	Data Packet, Block: 7
59	7.893703	::1	::1	TFTP	56	Acknowledgement, Block: 7
60	7.893890	::1	::1	TFTP	568	Data Packet, Block: 8
61	7.893916	::1	::1	TFTP	56	Acknowledgement, Block: 8
62	7.894140	::1	::1	TFTP	568	Data Packet, Block: 9
63	7.894163	::1	::1	TFTP	56	Acknowledgement, Block: 9
64	7.894435	::1	::1	TFTP	568	Data Packet, Block: 10
65	7.894460	::1	::1	TFTP	56	Acknowledgement, Block: 10
66	7.894824	::1	::1	TFTP	568	Data Packet, Block: 11
67	7.894851	::1	::1	TFTP	56	Acknowledgement, Block: 11
68	7.895181	::1	::1	TFTP	568	Data Packet, Block: 12
69	7.895210	::1	::1	TFTP	56	Acknowledgement, Block: 12
70	7.895559	::1	::1	TFTP	568	Data Packet, Block: 13
71	7.895583	::1	::1	TFTP	56	Acknowledgement, Block: 13
72	7.895887	::1	::1	TFTP	568	Data Packet, Block: 14
73	7.895911	::1	::1	TFTP	56	Acknowledgement, Block: 14
74	7.896141	::1	::1	TFTP	568	Data Packet, Block: 15
75	7.896165	::1	::1	TFTP	56	Acknowledgement, Block: 15
76	7.896416	::1	::1	TFTP	568	Data Packet, Block: 16
77	7.896444	::1	::1	TFTP	56	Acknowledgement, Block: 16
78	7.896730	::1	::1	TFTP	568	Data Packet, Block: 17
79	7.896758	::1	::1	TFTP	56	Acknowledgement, Block: 17
80	7.897048	::1	::1	TFTP	568	Data Packet, Block: 18
81	7.897077	::1	::1	TFTP	56	Acknowledgement, Block: 18
82	7.897349	::1	::1	TFTP	568	Data Packet, Block: 19
83	7.897390	::1	::1	TFTP	56	Acknowledgement, Block: 19
84	7.897647	::1	::1	TFTP	568	Data Packet, Block: 20
85	7.897680	::1	::1	TFTP	56	Acknowledgement, Block: 20
86	7.897972	::1	::1	TFTP	568	Data Packet, Block: 21
87	7.898029	::1	::1	TFTP	56	Acknowledgement, Block: 21
88	7.898393	::1	::1	TFTP	568	Data Packet, Block: 22
89	7.898433	::1	::1	TFTP	56	Acknowledgement, Block: 22
90	7.898756	::1	::1	TFTP	568	Data Packet, Block: 23
91	7.898787	::1	::1	TFTP	56	Acknowledgement, Block: 23
92	7.899165	::1	::1	TFTP	568	Data Packet, Block: 24

Fig.1 1-24 sent block

45	7.855045	::1	::1	TFTP	72	Read Request, File: RFC1350.txt, Transfer type: octet
> Frame 45: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface \Device\NPF_{Loopback}, id 0 > Null/Loopback > Internet Protocol Version 6, Src: ::1, Dst: ::1 > User Datagram Protocol, Src Port: 52970, Dst Port: 69 Source Port: 52970 Destination Port: 69 Length: 28 Checksum: 0xf766 [unverified] [Checksum Status: Unverified] [Stream index: 0] > [Timestamps] UDP payload (20 bytes) > Trivial File Transfer Protocol Opcode: Read Request (1) Source File: RFC1350.txt Type: octet						

Fig.2 The read request packet

```

46 7.885672      ::1      ::1      TFTP      568 Data Packet, Block: 1
▼ Frame 46: 568 bytes on wire (4544 bits), 568 bytes captured (4544 bits) on interface \Device\NPF_{...}, id 0
  Section number: 1
  > Interface id: 0 (\Device\NPF_{...})
  Encapsulation type: NULL/Loopback (15)
  Arrival Time: Mar 13, 2023 19:21:00.667579000 W. Europe Standard Time
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1678731600.667579000 seconds
  [Time delta from previous captured frame: 0.030627000 seconds]
  [Time delta from previous displayed frame: 0.030627000 seconds]
  [Time since reference or first frame: 7.885672000 seconds]
  Frame Number: 46
  Frame Length: 568 bytes (4544 bits)
  Capture Length: 568 bytes (4544 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: null:ipv6:udp:tftp:data]
  [Coloring Rule Name: UDP]
  [Coloring Rule String: udp]
  > Null/Loopback
  > Internet Protocol Version 6, Src: ::1, Dst: ::1
  > User Datagram Protocol, Src Port: 52971, Dst Port: 52970
  ▼ Trivial File Transfer Protocol
    Opcode: Data Packet (3)
    [Destination File: RFC1350.txt]
    [Read Request in frame 45]
    Block: 1
    [Full Block Number: 1]
  ▼ Data (512 bytes)
    Data: 0d0a0d0a0d0a0d0a0d0a0d0a4e6574776f726b2856f726b696e672047726f7570202020...
    [Length: 512]

```

Fig.3 First block of data transfer (during RRQ)

```

47 7.885788      ::1      ::1      TFTP      56 Acknowledgement, Block: 1
▼ Frame 47: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{...}, id 0
  Section number: 1
  > Interface id: 0 (\Device\NPF_{...})
  Encapsulation type: NULL/Loopback (15)
  Arrival Time: Mar 13, 2023 19:21:00.667695000 W. Europe Standard Time
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1678731600.667695000 seconds
  [Time delta from previous captured frame: 0.000116000 seconds]
  [Time delta from previous displayed frame: 0.000116000 seconds]
  [Time since reference or first frame: 7.885788000 seconds]
  Frame Number: 47
  Frame Length: 56 bytes (448 bits)
  Capture Length: 56 bytes (448 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: null:ipv6:udp:tftp]
  [Coloring Rule Name: UDP]
  [Coloring Rule String: udp]
  > Null/Loopback
  > Internet Protocol Version 6, Src: ::1, Dst: ::1
  > User Datagram Protocol, Src Port: 52970, Dst Port: 52971
  ▼ Trivial File Transfer Protocol
    Opcode: Acknowledgement (4)
    [Destination File: RFC1350.txt]
    [Read Request in frame 45]
    Block: 1
    [Full Block Number: 1]

```

Fig.4 First acknowledgement recieved after the first block is delivered

```

    144 7.907244      ::1          :~:         TFTP           185 Data Packet, Block: 50 (last)
▼ Frame 144: 185 bytes on wire (1480 bits), 185 bytes captured (1480 bits) on interface \Device\NPF_{Loopback}, id 0
   Section number: 1
     > Interface id: 0 (\Device\NPF_{Loopback})
       Encapsulation type: NULL/Loopback (15)
       Arrival Time: Mar 13, 2023 19:21:00.689151000 W. Europe Standard Time
       [Time shift for this packet: 0.00000000 seconds]
       Epoch Time: 1678731660.689151000 seconds
       [Time delta from previous captured frame: 0.000208000 seconds]
       [Time delta from previous displayed frame: 0.000208000 seconds]
       [Time since reference or first frame: 7.907244000 seconds]
       Frame Number: 144
       Frame Length: 185 bytes (1480 bits)
       Capture Length: 185 bytes (1480 bits)
       [Frame is marked: False]
       [Frame is ignored: False]
       [Protocols in frame: null:ipv6:udp:tftp:data]
       [Coloring Rule Name: UDP]
       [Coloring Rule String: udp]
     > Null/Loopback
     > Internet Protocol Version 6, Src: ::1, Dst: ::1
▼ User Datagram Protocol, Src Port: 52971, Dst Port: 52970
   Source Port: 52971
   Destination Port: 52970
   Length: 141
   Checksum: 0xdbc5 [unverified]
   [Checksum Status: Unverified]
   [Stream index: 1]
   > [Timestamps]
     UDP payload (133 bytes)
▼ Trivial File Transfer Protocol
   Opcode: Data Packet (3)
   [Destination File: RFC1350.txt]
   [Read Request in frame 45]
   Block: 50
   [Full Block Number: 50]
▼ Data (129 bytes)
   Data: 4f4c4d494e53404c43532e4d49542e45444550d0a0d0a0d0a0d0a0d0a0d0a0d0a...
   [Length: 129]
```

```

145 7.907271      ::1          TFTP      ::1      56 Acknowledgement, Block: 50
▼ Frame 145: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{Loopback}, id 0
  Section number: 1
  > Interface id: 0 (\Device\NPF_{Loopback})
    Encapsulation type: NULL/Loopback (15)
    Arrival Time: Mar 13, 2023 19:21:00.689178000 W. Europe Standard Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1678731660.689178000 seconds
    [Time delta from previous captured frame: 0.000027000 seconds]
    [Time delta from previous displayed frame: 0.000027000 seconds]
    [Time since reference or first frame: 7.907271000 seconds]
    Frame Number: 145
    Frame Length: 56 bytes (448 bits)
    Capture Length: 56 bytes (448 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: null:ipv6:udp:tftp]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
  > Null/Loopback
  > Internet Protocol Version 6, Src: ::1, Dst: ::1
  ▼ User Datagram Protocol, Src Port: 52970, Dst Port: 52971
    Source Port: 52970
    Destination Port: 52971
    Length: 12
    Checksum: 0x61c8 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
    > [Timestamps]
    UDP payload (4 bytes)
  ▼ Trivial File Transfer Protocol
    Opcode: Acknowledgement (4)
    [Destination File: RFC1350.txt]
    [Read Request in frame 45]
    Block: 50
    [Full Block Number: 50]

```

3.1 Discussion

Fig.1. shows the packets sent between the client and the server.

Fig.2. The Read Request (RRQ) packet consists of a frame, null/loopback, IPv6, UDP datagram, containing the source port, destination port, length of the packet and checksum to verify the integrity of the data, as well as a TFTP part containing opcode (01), source file ("RFC1350.txt") and type/mode (octet).

Fig.3. Represents the first block, specified by the given block number in the TFTP part of the packet sent after the connection was established during the RRQ. That is specified in the opcode section where it says Data Packet (03). It represents the first 512 bytes of data transferred from the server to the client. The packet also includes the destination file where the data will be transferred to. After the packet is received by the client, it sends an acknowledgement (Fig. 4.).

Fig.4. That can be seen in the opcode of the TFTP part of the packet which is Acknowledgement (04) It simply alerts the server with ACK # which means that block # was received. The server receives the ACK and sends next block of data.

Fig.5. Last packet of data. Block 50, which is the block with less than 512 bytes of data. This signals the server that block 50 is the last block of data to be sent over the established connection. Everything else is the same as all data packets received.

Fig.6. Acknowledgment of the last block number, which will terminate the connection.

RRQ vs WRQ:

RRQ:

- During read request, the server sends data and receives acknowledgements
- Client sends acknowledgements and receives data
- Server starts sending data directly
- Data is saved to the client's computer

WRQ:

- During the request, the server sends acknowledgements and receives data
- Client sends data and receives acknowledgements
- Client waits to receive ACK 0 before sending data
- Data is saved to the server's computer