TFTP Implementation

CS349 : Networks Lab
Name: Nikhil Agarwal
Roll No : 11012323
IIT Guwahati

---

1 Sample Results:

1.1 Client executing read option

```
nikhil@nikhil-HP-G42-Notebook-PC:~/Downloads/test1$ g++ -std=c++0x  tftpclient.cpp
nikhil@nikhil-HP-G42-Notebook-PC:~/Downloads/test1$ ./a.out 10.1.2.49 r nik.txt
Sending [Read Request] for file nik.txt with packet size 21 bytes
Received block #1 of data with packet size 516 bytes
Sending Ack #1
Received block #2 of data with packet size 516 bytes
Sending Ack #2
Received block #3 of data with packet size 516 bytes
Sending Ack #3
Received block #4 of data with packet size 516 bytes
Sending Ack #4
Received block #5 of data with packet size 516 bytes
Sending Ack #5
Received block #6 of data with packet size 516 bytes
Sending Ack #6
Received block #7 of data with packet size 516 bytes
Sending Ack #7
Received block #8 of data with packet size 516 bytes
Sending Ack #8
Received block #9 of data with packet size 516 bytes
Sending Ack #9
Received block #10 of data with packet size 212 bytes
Sending Ack #10
file closed
nikhil@nikhil-HP-G42-Notebook-PC:~/Downloads/test1$
```

Figure 1: The client executing read option

When the client executed read option we can see from the above figure that the client is receiving the packets of 516 bytes out of which 4 bytes are reserved for opcode and block number and hence only 512 bytes of data are received. After receiving the first block of data, it sends the acknowledgement to the server confirming the receive of block. Now the client will continue in this way until he

receives the packet of size less than 516. Then,it will come to know that it is the last packet and after sending acknowledgement it will close the file.

```
nikhil@nikhil-HP-G42-Notebook-PC:~/Downloads/test1$ g++ -std=c++0x tftpserver.cpp
nikhil@nikhil-HP-G42-Notebook-PC:~/Downloads/test1$ ./a.out
waiting on port 12345
Received [Read Request] for file nik.txt with packet size 21 bytes
File size is 4816bytes
Sending block #1 of data with packet size 516 bytes
Received Ack #1
Sending block #2 of data with packet size 516 bytes
Received Ack #2
Sending block #3 of data with packet size 516 bytes
Received Ack #3
Sending block #4 of data with packet size 516 bytes
Received Ack #4
Sending block #5 of data with packet size 516 bytes
Received Ack #5
Sending block #6 of data with packet size 516 bytes
Received Ack #6
Sending block #7 of data with packet size 516 bytes
Received Ack #7
Sending block #8 of data with packet size 516 bytes
Received Ack #8
Sending block #9 of data with packet size 516 bytes
Received Ack #9
Sending block #10 of data with packet size 212 bytes
Received Ack #10
waiting on port 12345
```

Figure 2: The server after client executed read option

When the client executed read option we can see from the above figure that the server is sending the packets of 516 bytes out of which 4 bytes are reserved for opcode and block number and hence only 512 bytes of data are sent. After sending the first block of data, it waits for the acknowledgement packet. After receiving the acknowledgement packet, it will again send the next block of data and it will continue doing the above process until it has not transmitted the entire file.

1.2 Client executing write option

```
nikhil@nikhil-HP-G42-Notebook-PC:~/Downloads/test1$ g++ -std=c++0x  tftpclient.cpp
nikhil@nikhil-HP-G42-Notebook-PC:~/Downloads/test1$ ./a.out 10.1.2.49 w nik.txt
File size is 4816bytes
Sending [Write Request] for file nik.txt with packet size 21 bytes
Received Ack #0
Sending block #1 of data with packet size 516 bytes
Received Ack #1
Sending block #2 of data with packet size 516 bytes
Received Ack #2
Sending block #3 of data with packet size 516 bytes
Received Ack #3
Sending block #4 of data with packet size 516 bytes
Received Ack #4
Sending block #5 of data with packet size 516 bytes
Received Ack #5
Sending block #6 of data with packet size 516 bytes
Received Ack #6
Sending block #7 of data with packet size 516 bytes
Received Ack #7
Sending block #8 of data with packet size 516 bytes
Received Ack #8
Sending block #9 of data with packet size 516 bytes
Received Ack #9
Sending block #10 of data with packet size 212 bytes
Received Ack #10
nikhil@nikhil-HP-G42-Notebook-PC:~/Downloads/test1$
```

Figure 3: The client executing write option

When the client requests for the read option , it will receive an acknowledgement from the server showing that it is ready to receive data. As we can see that ack 0 is received in the above figure. Now the client sends first block of data and it waits for the acknowledgement packet to arrive. After receiving the acknowledgement packet, it will again send the next block of data and it will continue doing the above process until it has not transmitted the entire file. After transmitting the entire file it will receive the last acknowledgement.

```
nikhil@nikhil-HP-G42-Notebook-PC:~/Downloads/test1$ g++ -std=c++0x tftpserver.cpp
nikhil@nikhil-HP-G42-Notebook-PC:~/Downloads/test1$ ./a.out
waiting on port 12345
Received [Write Request] for file nik.txt with packet size 21 bytes
Sending Ack #0
Received block #1 of data with packet size 516 bytes
Sending Ack #1
Received block #2 of data with packet size 516 bytes
Sending Ack #2
Received block #3 of data with packet size 516 bytes
Sending Ack #3
Received block #4 of data with packet size 516 bytes
Sending Ack #4
Received block #5 of data with packet size 516 bytes
Sending Ack #5
Received block #6 of data with packet size 516 bytes
Sending Ack #6
Received block #7 of data with packet size 516 bytes
Sending Ack #7
Received block #8 of data with packet size 516 bytes
Sending Ack #8
Received block #9 of data with packet size 516 bytes
Sending Ack #9
Received block #10 of data with packet size 212 bytes
Sending Ack #10
file closed
waiting on port 12345
```

Figure 4: Server's response after client executed write

When the client requests for the read option , the server will send an acknowledgement to the client showing that it is ready to receive data. As we can see that ack 0 is sent in the above figure. Now the client sends first packet of data and the server will display the received message and send the acknowledgement packet, the client then send the next packet and server repeats the above process unless received packet size becomes less than 516 bytes at which it will send the final acknowledgement.

1.3 Client executing read option in case of timeouts

I was not sure of how to check the cases for timeouts since the server was at my own pc. So to check my output I stopped sending the first acknowledgement.

```
nikhil@nikhil-HP-G42-Notebook-PC:~/Downloads/test1$ g++ -std=c++0x  tftpclient.cpp
nikhil@nikhil-HP-G42-Notebook-PC:~/Downloads/test1$ ./a.out 10.1.2.49 r nik.txt
Sending [Read Request] for file nik.txt with packet size 21 bytes
Received block #1 of data with packet size 516 bytes
expecting block 2, received block 1
Sending Ack #1
Received block #2 of data with packet size 516 bytes
expecting block 3, received block 2
Sending Ack #2
Received block #3 of data with packet size 516 bytes
expecting block 4, received block 3
Sending Ack #3
Received block #4 of data with packet size 516 bytes
expecting block 5, received block 4
Sending Ack #4
Received block #5 of data with packet size 516 bytes
expecting block 6, received block 5
Sending Ack #5
Received block #6 of data with packet size 516 bytes
expecting block 7, received block 6
Sending Ack #6
Received block #7 of data with packet size 516 bytes
expecting block 8, received block 7
Sending Ack #7
Received block #8 of data with packet size 516 bytes
expecting block 9, received block 8
Sending Ack #8
Received block #9 of data with packet size 516 bytes
expecting block 10, received block 9
Sending Ack #9
Received block #10 of data with packet size 212 bytes
file closed
nikhil@nikhil-HP-G42-Notebook-PC:~/Downloads/test1$
```

Figure 5: The client executing read in case of time-out

From the above figure we know that after receiving the first packet, the client didn't send any acknowledgement and it was expecting block 2 . The received block again was 1 because the server did not get any acknowledgement. So this time the client responds with the acknowledgement and then the server sends 2nd block of data . The client continues in similar fashion until it gets a packet less than 516 bytes at which point it stops.

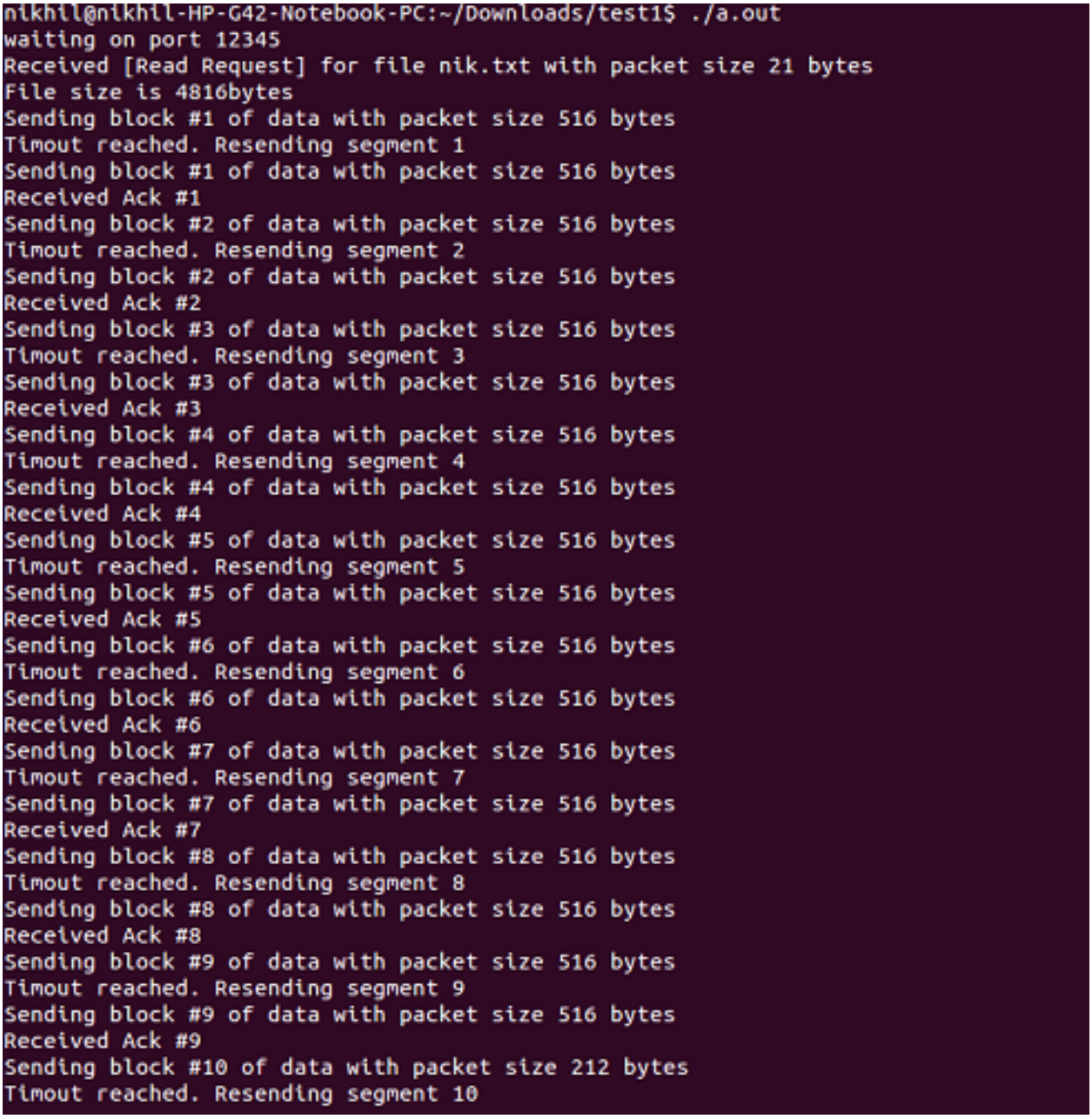

Figure 6: The server after client executed read option in case of time-out

From the above figure we know that after sending the first packet and not getting any acknowledgement the server times out. It then resends the same packet . As, this time it receives acknowledgement it sends another block of data. The server continues in similar fashion until it sends a packet less than 516 bytes at which point it stops.

1.4 Client executing write option in case of timeouts

To check my output I stopped sending the first acknowledgement.

```
Sending block #1 of data with packet size 516 bytes
Timout reached. Resending segment 1
Sending block #1 of data with packet size 516 bytes
Received Ack #1
Sending block #2 of data with packet size 516 bytes
Timout reached. Resending segment 2
Sending block #2 of data with packet size 516 bytes
Received Ack #2
Sending block #3 of data with packet size 516 bytes
Timout reached. Resending segment 3
Sending block #3 of data with packet size 516 bytes
Received Ack #3
Sending block #4 of data with packet size 516 bytes
Timout reached. Resending segment 4
Sending block #4 of data with packet size 516 bytes
Received Ack #4
Sending block #5 of data with packet size 516 bytes
Timout reached. Resending segment 5
Sending block #5 of data with packet size 516 bytes
Received Ack #5
Sending block #6 of data with packet size 516 bytes
Timout reached. Resending segment 6
Sending block #6 of data with packet size 516 bytes
Received Ack #6
Sending block #7 of data with packet size 516 bytes
Timout reached. Resending segment 7
Sending block #7 of data with packet size 516 bytes
Received Ack #7
Sending block #8 of data with packet size 516 bytes
Timout reached. Resending segment 8
Sending block #8 of data with packet size 516 bytes
Received Ack #8
Sending block #9 of data with packet size 516 bytes
Timout reached. Resending segment 9
Sending block #9 of data with packet size 516 bytes
Received Ack #9
Sending block #10 of data with packet size 212 bytes
Timout reached. Resending segment 10
Sending block #10 of data with packet size 212 bytes
Timout reached. Resending segment 10
Sending block #10 of data with packet size 212 bytes
Timout reached. Resending segment 10
```

Figure 7: The client executing write option in case of time-out

From the above figure we know that after sending the first packet and not getting any acknowledgement the client times out. It then resends the same packet . As, this time it receives acknowledgement it sends another packet. The client continues in similar fashion until it sends a packet less than 516 bytes at which point it stops.

```
waiting on port 12345
Received [Write Request] for file nik.txt with packet size 21 bytes
Sending Ack #0
Received block #1 of data with packet size 516 bytes
expecting block no. 2, received block no. 1
Sending Ack #1
Received block #2 of data with packet size 516 bytes
expecting block no. 3, received block no. 2
Sending Ack #2
Received block #3 of data with packet size 516 bytes
expecting block no. 4, received block no. 3
Sending Ack #3
Received block #4 of data with packet size 516 bytes
expecting block no. 5, received block no. 4
Sending Ack #4
Received block #5 of data with packet size 516 bytes
expecting block no. 6, received block no. 5
Sending Ack #5
Received block #6 of data with packet size 516 bytes
expecting block no. 7, received block no. 6
Sending Ack #6
Received block #7 of data with packet size 516 bytes
expecting block no. 8, received block no. 7
Sending Ack #7
Received block #8 of data with packet size 516 bytes
expecting block no. 9, received block no. 8
Sending Ack #8
Received block #9 of data with packet size 516 bytes
expecting block no. 10, received block no. 9
Sending Ack #9
Received block #10 of data with packet size 212 bytes
file closed
waiting on port 12345
expected packet with opcode 1 or 2, received with opcode 3
waiting on port 12345
expected packet with opcode 1 or 2, received with opcode 3
waiting on port 12345
expected packet with opcode 1 or 2, received with opcode 3
waiting on port 12345
expected packet with opcode 1 or 2, received with opcode 3
waiting on port 12345
expected packet with opcode 1 or 2, received with opcode 3
```

Figure 8: Server's response after client executed write in case of time-out

From the above figure we know that after receiving the first packet, the server didn't send any acknowledgement and it was expecting block 2 . The received block again was 1 because the client did not get any acknowledgement. So this time the server responds with the acknowledgement and

then the server sends 2nd block of data . The client continues in similar fashion until it gets a packet less than 516 bytes at which point it stops. As the server , receives the last block of data , it does not send any acknowledgement and hence the client didn't know whether server has received or not and it resends again and again. On the other hand the server has already closed the file so it expects a read or write request from the client but the client goes on sending the same packet 10 times until the client finally decides to drop it.