

# **Studienarbeit in Datenkommunikation**

## **Implementierung eines TFTP**

Referenten: Nurdan Eren, Natalia Pavlik

# Agenda

1

2

3

4

5

6

1. TFTP

2. Einführung - Projektbasis

3. Beispiele aus der Umsetzung

4. Write Request und Read Request

5. Error Handling

6. Fazit



# Agenda

1

## 1. TFTP

2

## 2. Einführung - Projektbasis

3

## 3. Beispiele aus der Umsetzung

4

→ Vorführung der Implementation

## 4. Write Request und Read Request

5

→ Wireshark - Analyse

## 5. Error Handling

6

→ Wireshark - Analyse

## 6. Fazit



# Agenda

1

2

3

4

5

6

## 1. TFTP

2. Einführung - Projektbasis

3. Beispiele aus der Umsetzung

4. Write Request und Read Request

5. Error Handling

6. Fazit





# 1. TFTP

## 1.1 Allgemeines

- TFTP = Trivial File Transfer Protocol
- Protokoll zur Übertragung von Dateien
- Setzt meist auf UDP (User Datagram Protocol) auf
- Klein, einfach implementierbar
- Unterstützt Schreiben und Lesen von Daten auf dem Server
- Keine Verschlüsselung oder Authentifizierung

# 1. TFTP

## 2.1 Protokolleigenschaften I

- 5 Pakettypen:

Opcode	Operation	
1	RRQ	Read Request
2	WRQ	Write Request
3	DATA	Data
4	ACK	Acknowledgement
5	ERROR	Error

- Bsp. Aufbau eines Pakets:

2 Bytes	RRQ	1 Byte		1 Byte
RRQ=1	„File Name“	0	„Mode“	0

# 1. TFTP

## 2.1 Protokolleigenschaften II

- Pakete haben eine aufsteigende Blocknummer
- Gleiche Blocknummer bei Bestätigungen und Nachricht
- Fehlercodes:

0	Not defined
<b>1</b>	<b>File not found</b>
2	Access violation
3	Disk full or allocation needed
<b>4</b>	<b>Illegal TFTP Operation</b>
5	Unknown transfer ID
<b>6</b>	<b>File already exists</b>
7	No such user

# 1. TFTP

## 2.3 UDP

- User Data Protocol (UDP)
- Verbindungsloses, unzuverlässiges Netzwerkprotokoll
- Versand von Datagrammen in IP-basierten Rechnernetzen
- Keine Garantie, dass Pakete ankommen
- Anwendung benötigt Verbindungsmanagement
- Kommunikation über Datagram Sockets (IP + Port)



# Agenda

1

2

3

4

5

6

1. TFTP

## **2. Einführung Projektbasis**

3. Beispiele aus der Umsetzung

4. Write Request und Read Request

5. Error Handling

6. Fazit



# 2. Einführung – Projektbasis

## 2.1 Einführung

- Zwei Hauptprozesse: Client und Server
- Aufgaben des Clients:
  - Konsoleingaben akzeptieren und im Terminal anzeigen
  - Versenden der Eingaben an den Server mittels TFTP
  - Eingaben des Server empfangen und im Terminal anzeigen
- Aufgaben des Servers:
  - Eingaben vom Client mittels TFTP empfangen
  - Eingaben an Client versenden
- Fehlerbehandlung

# 2. Einführung – Projektbasis

## 2.3 Projektbasis

- Implementation eines TFTP basierend auf UDP in Java
- Entwicklungsumgebung: Eclipse
- Projektbasis:
  - Anwendungsschicht: TFTP
  - Transportschicht: UDP (Server-Port: 2018, well-known 69)
  - Netzwerkschicht: IP (127.0.0.1 – localhost)

# 2. Einführung – Projektbasis

## 2.3 Implementierung

- Ausführung vorgegebener Verhaltensweisen nach Start des Servers und Clients
- WRQ: ACK → Texteingabe durch Client → ACK → RRQ
- RRQ: fest implementierter Text an Client → ACK
- Timer und Wiederholzähler (im Bsp.)
- Liste mit allen bereits enthaltenen Paketen (im Bsp.)
- Kennzahl erwartete Blocknummer (im Bsp.)

# Agenda

1

2

3

4

5

6

1. TFTP

2. Einführung - Projektbasis

**3. Beispiele aus der Umsetzung**

4. Write Request und Read Request


5. Error Handling

6. Fazit



# 3. Beispiele aus der Umsetzung

## 3.1 – Bsp. 1 – Timer



```
while (true) {
    // checks if an ACK is needed
    if (ackNeeded) {

        // checks if an ACK was received; if it was received, everything goes on as expected
        try {
            clientSocket.receive(incomingPacket);

            if (incomingData[1] == OP_ACK) {

                // stops/resets the timer when an ACK was received
                clientSocket.setSoTimeout(0);
                System.out.println("CLIENT: " + " " + "ACK received.");
                // checks if a RRQ was already sent
                if (!RRQsent) {
                    // checks if the last package was already sent
                    if (lastPackage) {
                        // sends a RRQ to the server in case no RRQ was sent yet and if the last package of the DataMessage was sent
                        message.sendRequestMessage(OP_RRQ, "filename.txt", "octet", sendData, ipAddress, SERVER_PORT, clientSocket);
                        RRQsent = true;
                        ackNeeded = false;

                    } else {
                        sendDataMessage();
                        // starts the timer
                        clientSocket.setSoTimeout(3000);
                    }
                }
            }
        }
    }
}
```

# 3. Beispiele aus der Umsetzung

## 3.2 – Bsp. 2 – Data Messages

```
if (incomingData[1] == OP_DATA) {  
    // initiates a data output stream for incoming messages  
    dos = new DataOutputStream(baos);  
  
    // gets the blocknumber from the received DataMessage and saves it in a byteArray, so it can be used in sendAckMessage  
    byte[] receivedBlockNumber = helper.getBlockNumberFromReceivedMessageAsByteArray(incomingData);  
    actualReceivedBlockNumber = helper.byteArrayToInt(receivedBlockNumber);  
  
    // handles received messages  
    // checks if it can be added to the dataOutputStream or if message has to be discarded/ignored since it was already received  
    handleReceivedMessage();  
  
    /* checks if the received blocknumber equals the expected blocknumber  
    * if the package was already received, handleReceivedMessage() takes care of it  
    * if received package has a higher blocknumber than expected message, an error will be send (package missing)  
    */  
  
    if(actualReceivedBlockNumber > blockNumberExpected){  
        // ERROR NUMBER 4 - Illegal TFTP operation (here: package got lost)  
        errorNumber = 4;  
        message.sendErrorMessage(errorNumber, "Received blocknumber is higher than expected BN -> Package missing.", sendData, ipAddress, SERVER_PORT, clientSocket);  
    }  
  
    // sends the ACK for received Message  
    message.sendAckMessage(receivedBlockNumber, sendData, ipAddress, SERVER_PORT, clientSocket);  
  
    // checks if the last package was received and then prints the completely received data in a byteArrayOutputStream  
    // also prints a list of all received blocks  
    if (incomingPacket.getLength() < 516) {  
        System.out.println("CLIENT: " + " " + "Message from server received (Received blocks: " + receivedBlockNumbers.toString() + ").");  
        System.out.println("\n" + "CLIENT: " + " " + "Complete Message: \n" + new String(baos.toByteArray()) + "\n");  
    }  
}
```

# 3. Beispiele aus der Umsetzung

## 3.3 Vorführung der Implementierung

- **LIVE.**



# Agenda

1

2

3

4

5

6

1. TFTP

2. Einführung - Projektbasis

3. Beispiele aus der Umsetzung

**4. Write Request und Read Request**

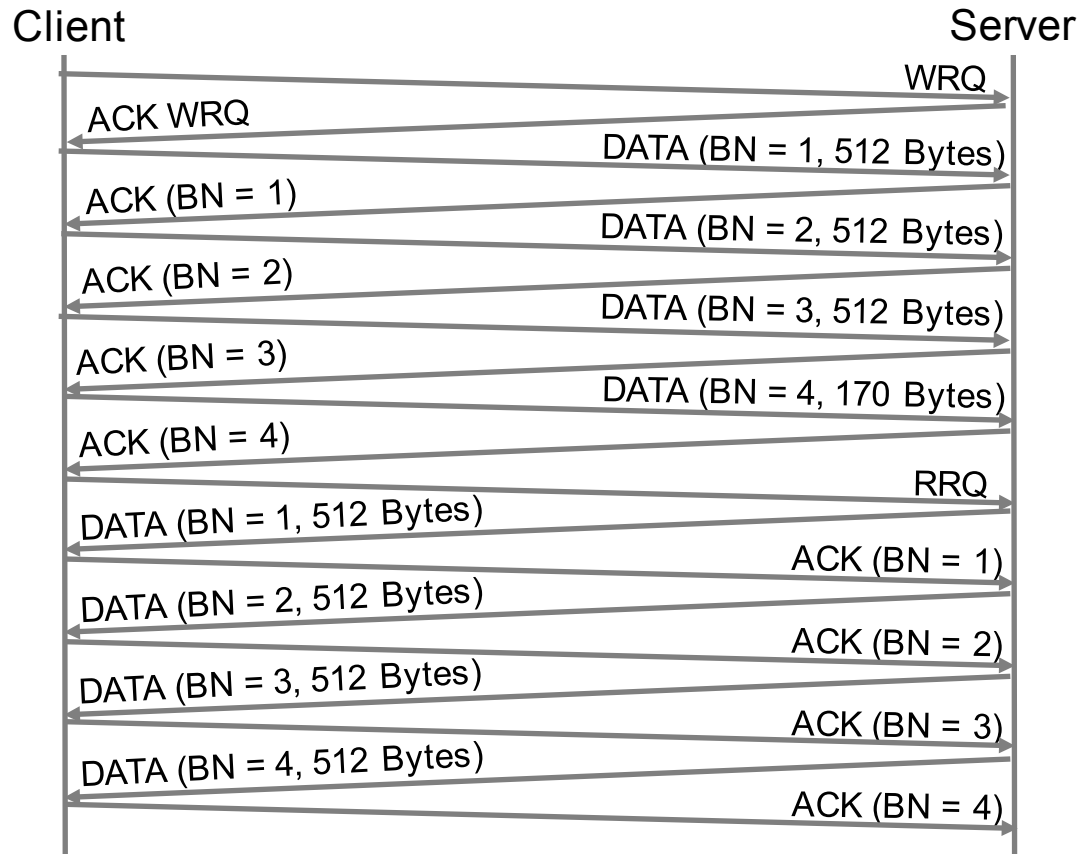
5. Error Handling

6. Fazit



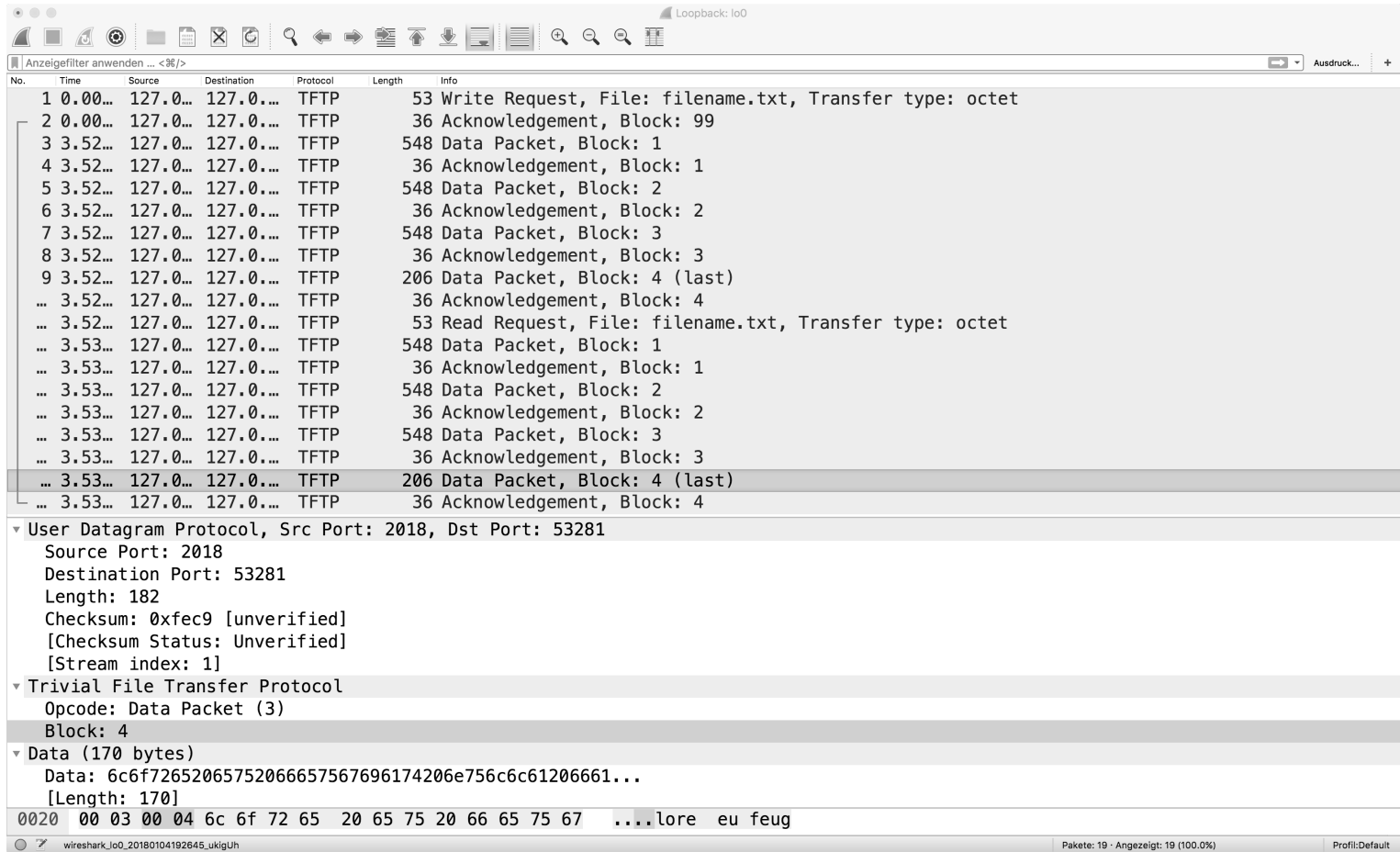
# 4. Write Request und Read Request

## 4.1 Fehlerfreier WRQ



# 4. Write Request und Read Request

## 4.2 Fehlerfreier WRQ - Wireshark



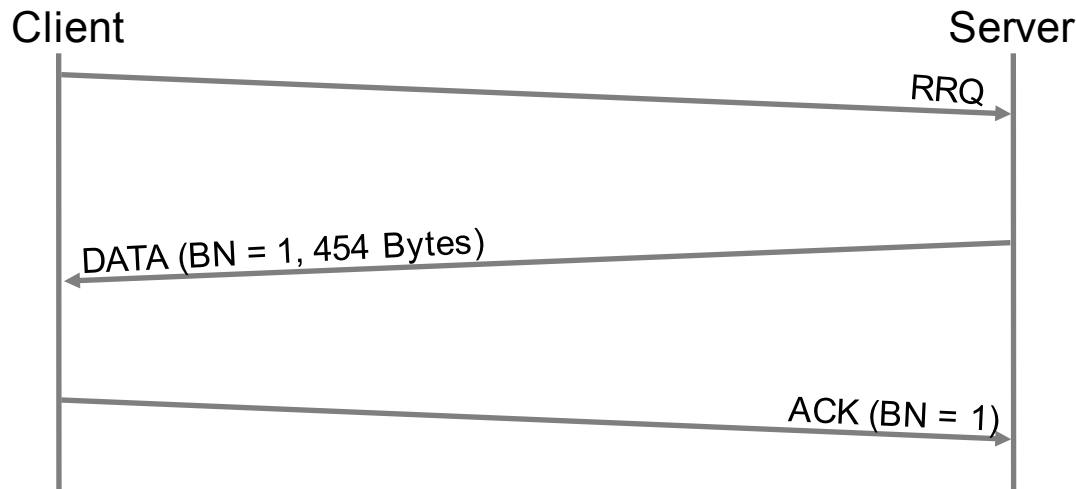
The screenshot shows a Wireshark packet capture on the 'Loopback: lo0' interface. The packet list on the left shows 19 packets. Packet 13 is selected, which is a Read Request (WRQ) from 127.0.0.1 to 127.0.0.1. The packet details pane on the right shows the structure of the TFTP message: User Datagram Protocol (Source Port: 2018, Destination Port: 53281), Trivial File Transfer Protocol (Opcode: Data Packet (3), Block: 4), and Data (170 bytes). The data field shows a hexadecimal representation of the file content, starting with '0020 00 03 00 04 6c 6f 72 65 20 65 75 20 66 65 75 67' and ending with '....lore eu feug'.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00...	127.0...	127.0...	TFTP	53	Write Request, File: filename.txt, Transfer type: octet
2	0.00...	127.0...	127.0...	TFTP	36	Acknowledgement, Block: 99
3	3.52...	127.0...	127.0...	TFTP	548	Data Packet, Block: 1
4	3.52...	127.0...	127.0...	TFTP	36	Acknowledgement, Block: 1
5	3.52...	127.0...	127.0...	TFTP	548	Data Packet, Block: 2
6	3.52...	127.0...	127.0...	TFTP	36	Acknowledgement, Block: 2
7	3.52...	127.0...	127.0...	TFTP	548	Data Packet, Block: 3
8	3.52...	127.0...	127.0...	TFTP	36	Acknowledgement, Block: 3
9	3.52...	127.0...	127.0...	TFTP	206	Data Packet, Block: 4 (last)
...	3.52...	127.0...	127.0...	TFTP	36	Acknowledgement, Block: 4
...	3.52...	127.0...	127.0...	TFTP	53	Read Request, File: filename.txt, Transfer type: octet
...	3.53...	127.0...	127.0...	TFTP	548	Data Packet, Block: 1
...	3.53...	127.0...	127.0...	TFTP	36	Acknowledgement, Block: 1
...	3.53...	127.0...	127.0...	TFTP	548	Data Packet, Block: 2
...	3.53...	127.0...	127.0...	TFTP	36	Acknowledgement, Block: 2
...	3.53...	127.0...	127.0...	TFTP	548	Data Packet, Block: 3
...	3.53...	127.0...	127.0...	TFTP	36	Acknowledgement, Block: 3
...	3.53...	127.0...	127.0...	TFTP	206	Data Packet, Block: 4 (last)
...	3.53...	127.0...	127.0...	TFTP	36	Acknowledgement, Block: 4

▼ User Datagram Protocol, Src Port: 2018, Dst Port: 53281  
Source Port: 2018  
Destination Port: 53281  
Length: 182  
Checksum: 0xfec9 [unverified]  
[Checksum Status: Unverified]  
[Stream index: 1]  
▼ Trivial File Transfer Protocol  
Opcode: Data Packet (3)  
Block: 4  
▼ Data (170 bytes)  
Data: 6c6f7265752066657567696174206e756c6c612066661...  
[Length: 170]  
0020 00 03 00 04 6c 6f 72 65 20 65 75 20 66 65 75 67 ....lore eu feug

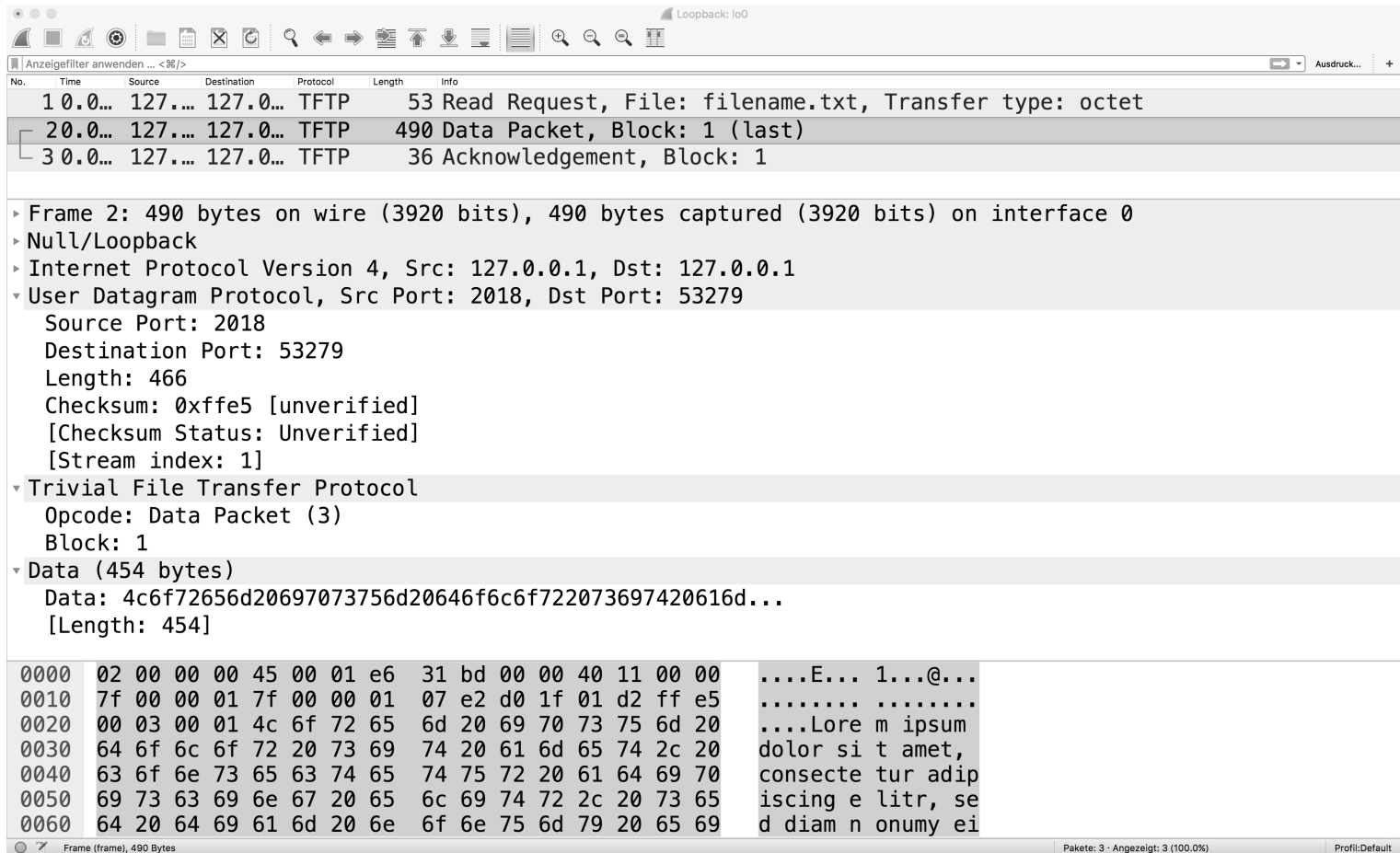
# 4. Write Request und Read Request

## 4.3 Fehlerfreier RRQ



# 4. Write Request und Read Request

## 4.4 Fehlerfreier RRQ - Wireshark



Wireshark packet capture showing a TFTP Read Request (RRQ) and its response. The packet list shows three packets: a Read Request (53 bytes), a Data Packet (490 bytes), and an Acknowledgement (36 bytes). The packet details pane shows the structure of the Data Packet, including the Trivial File Transfer Protocol header and the data payload. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0...	127.0.0.1	127.0.0.1	TFTP	53	Read Request, File: filename.txt, Transfer type: octet
2	0.0...	127.0.0.1	127.0.0.1	TFTP	490	Data Packet, Block: 1 (last)
3	0.0...	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 1

Frame 2: 490 bytes on wire (3920 bits), 490 bytes captured (3920 bits) on interface 0

- Null/Loopback
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- User Datagram Protocol, Src Port: 2018, Dst Port: 53279
  - Source Port: 2018
  - Destination Port: 53279
  - Length: 466
  - Checksum: 0xffe5 [unverified]  
[Checksum Status: Unverified]
  - [Stream index: 1]
- Trivial File Transfer Protocol
  - Opcode: Data Packet (3)
  - Block: 1
- Data (454 bytes)
  - Data: 4c6f72656d20697073756d2064666c6f722073697420616d...
  - [Length: 454]

0000	02 00 00 00 45 00 01 e6	31 bd 00 00 40 11 00 00	....E... 1...@...
0010	7f 00 00 01 7f 00 00 01	07 e2 d0 1f 01 d2 ff e5	.....
0020	00 03 00 01 4c 6f 72 65	6d 20 69 70 73 75 6d 20	....Lore m ipsum
0030	64 6f 6c 6f 72 20 73 69	74 20 61 6d 65 74 2c 20	dolor si t amet,
0040	63 6f 6e 73 65 63 74 65	74 75 72 20 61 64 69 70	consecte tur adip
0050	69 73 63 69 6e 67 20 65	6c 69 74 72 2c 20 73 65	iscing e litr, se
0060	64 20 64 69 61 6d 20 6e	6f 6e 75 6d 79 20 65 69	d diam n onumy ei

# Agenda

1

2

3

4

5

6

1. TFTP

2. Einführung - Projektbasis

3. Beispiele aus der Umsetzung

4. Write Request und Read Request

**5. Error Handling**

6. Fazit



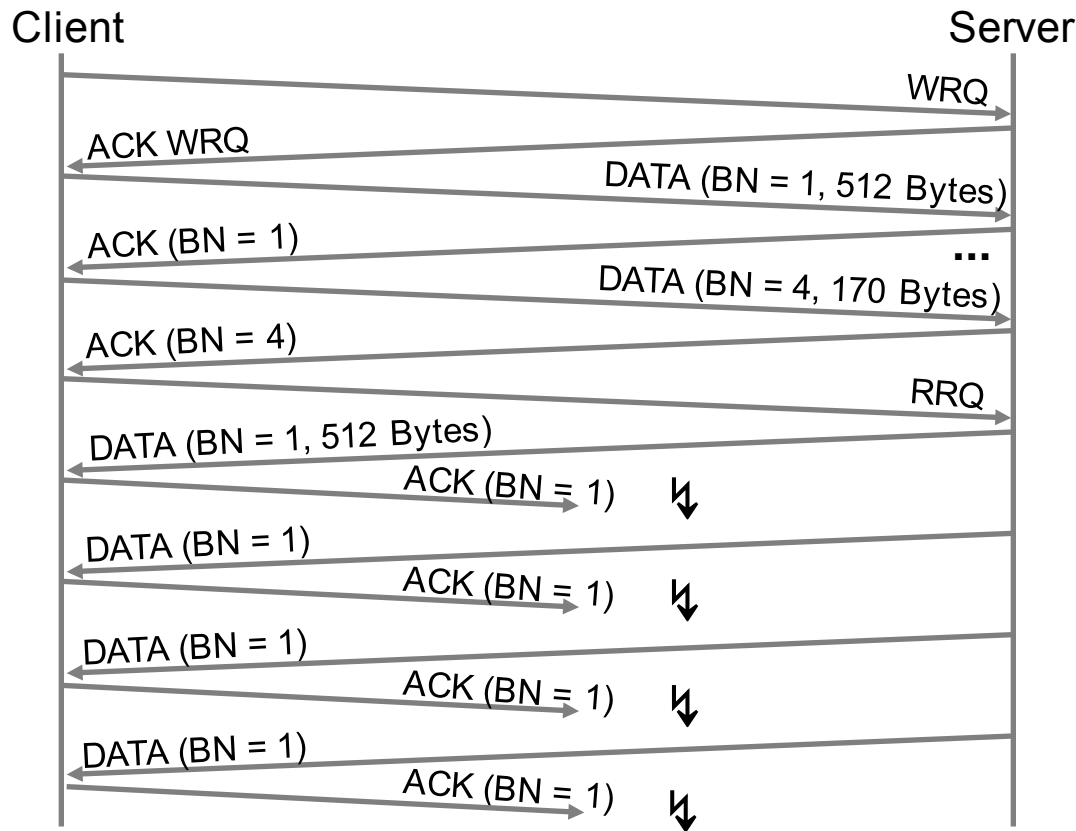
# 5. Error Handling

## 5.1 Error Nachrichten

- Bestätigung (ACK) kommt nicht an
- Paket mit gleicher Blocknummer (BN) bereits vorhanden  
→ 3 erneute Sendeveruche  
→ anschließend Programmabbruch
- Pakete der Message sind verloren gegangen

# 5. Error Handling

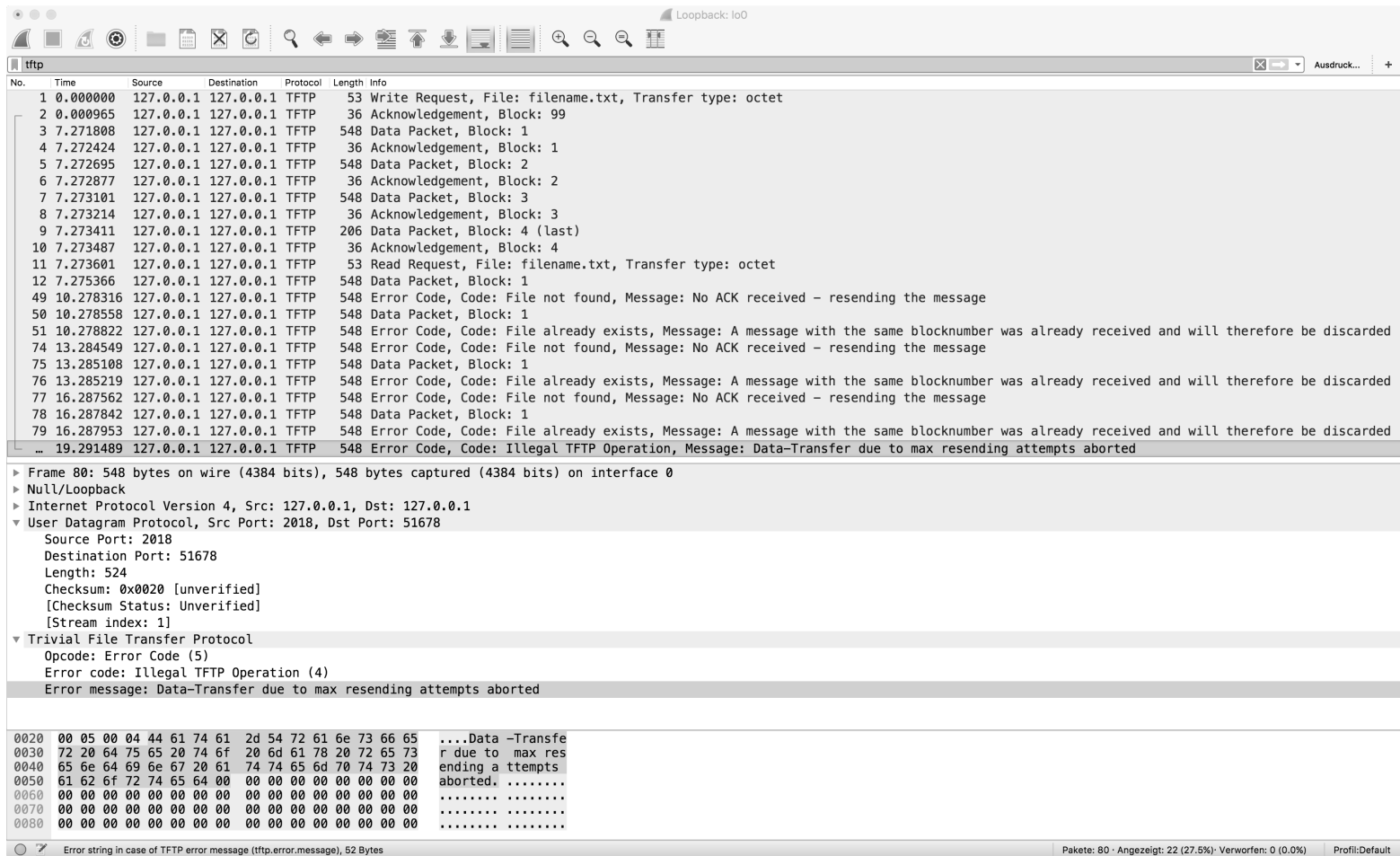
## 5.2 Error Handling





# 5. Error Handling

## 5.3 Error Handling – Wireshark – Bsp. 1



Wireshark packet capture showing TFTP error handling. The packet list shows a sequence of requests and acknowledgments, followed by several error messages (File not found, File already exists, and Illegal TFTP Operation). The packet details pane shows the structure of the TFTP error message, including the opcode (Error Code) and the error code (Illegal TFTP Operation). The packet bytes pane shows the raw data of the error message.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TFTP	53	Write Request, File: filename.txt, Transfer type: octet
2	0.000965	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 99
3	7.271808	127.0.0.1	127.0.0.1	TFTP	548	Data Packet, Block: 1
4	7.272424	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 1
5	7.272695	127.0.0.1	127.0.0.1	TFTP	548	Data Packet, Block: 2
6	7.272877	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 2
7	7.273101	127.0.0.1	127.0.0.1	TFTP	548	Data Packet, Block: 3
8	7.273214	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 3
9	7.273411	127.0.0.1	127.0.0.1	TFTP	206	Data Packet, Block: 4 (last)
10	7.273487	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 4
11	7.273601	127.0.0.1	127.0.0.1	TFTP	53	Read Request, File: filename.txt, Transfer type: octet
12	7.275366	127.0.0.1	127.0.0.1	TFTP	548	Data Packet, Block: 1
49	10.278316	127.0.0.1	127.0.0.1	TFTP	548	Error Code, Code: File not found, Message: No ACK received - resending the message
50	10.278558	127.0.0.1	127.0.0.1	TFTP	548	Data Packet, Block: 1
51	10.278822	127.0.0.1	127.0.0.1	TFTP	548	Error Code, Code: File already exists, Message: A message with the same blocknumber was already received and will therefore be discarded
74	13.284549	127.0.0.1	127.0.0.1	TFTP	548	Error Code, Code: File not found, Message: No ACK received - resending the message
75	13.285108	127.0.0.1	127.0.0.1	TFTP	548	Data Packet, Block: 1
76	13.285219	127.0.0.1	127.0.0.1	TFTP	548	Error Code, Code: File already exists, Message: A message with the same blocknumber was already received and will therefore be discarded
77	16.287562	127.0.0.1	127.0.0.1	TFTP	548	Error Code, Code: File not found, Message: No ACK received - resending the message
78	16.287842	127.0.0.1	127.0.0.1	TFTP	548	Data Packet, Block: 1
79	16.287953	127.0.0.1	127.0.0.1	TFTP	548	Error Code, Code: File already exists, Message: A message with the same blocknumber was already received and will therefore be discarded
...	19.291489	127.0.0.1	127.0.0.1	TFTP	548	Error Code, Code: Illegal TFTP Operation, Message: Data-Transfer due to max resending attempts aborted

Frame 80: 548 bytes on wire (4384 bits), 548 bytes captured (4384 bits) on interface 0

Null/Loopback

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

User Datagram Protocol, Src Port: 2018, Dst Port: 51678

- Source Port: 2018
- Destination Port: 51678
- Length: 524
- Checksum: 0x0020 [unverified]  
[Checksum Status: Unverified]
- [Stream index: 1]

Trivial File Transfer Protocol

- Opcode: Error Code (5)
- Error code: Illegal TFTP Operation (4)
- Error message: Data-Transfer due to max resending attempts aborted

0020 00 05 00 04 44 61 74 61 2d 54 72 61 6e 73 66 65 ....Data -Transfe  
0030 72 20 64 75 65 20 74 6f 20 6d 61 78 20 72 65 73 r due to max res  
0040 65 6e 64 69 6e 67 20 61 74 74 65 6d 70 74 73 20 ending a ttempts  
0050 61 62 6f 72 74 65 64 00 00 00 00 00 00 00 00 aborted. ....  
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

Error string in case of TFTP error message (tftp.error.message), 52 Bytes

Pakete: 80 - Angezeigt: 22 (27.5%): Verworfen: 0 (0.0%) Profil: Default

# 5. Error Handling

## 5.4 Fallbeispiele

- Was passiert wenn Server und Client erneut gestartet werden?

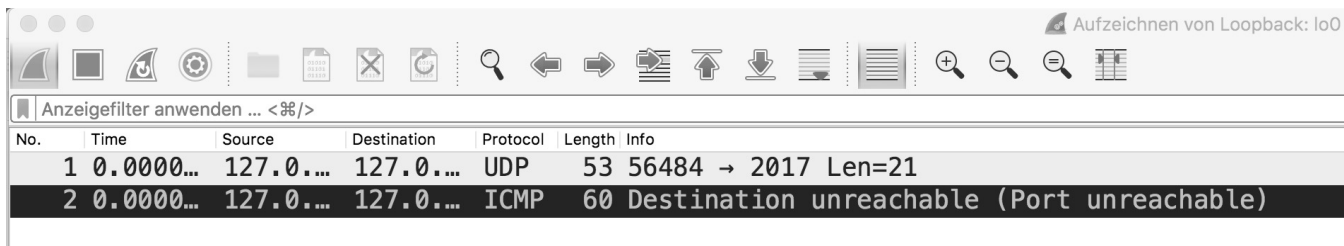
- Bei Neustart des Servers:

```
8 7.839595 127.0.0.1 127.0.0.1 TFTP 53 Write Request, File: filename.txt, Transfer type: octet
9 7.839720 127.0.0.1 127.0.0.1 TFTP 36 Acknowledgement, Block: 99
10 8.999919 127.0.0.1 127.0.0.1 TFTP 38 Data Packet, Block: 1 (last)
11 9.000160 127.0.0.1 127.0.0.1 TFTP 548 Error Code, Code: File already exists, Message: A message with the same blocknumber was already received and will therefore be discarded.
```

- Bei Neustart des Clients bei laufendem Server mit vorher gesendetem WRQ

```
8 7.839595 127.0.0.1 127.0.0.1 TFTP 53 Write Request, File: filename.txt, Transfer type: octet
9 7.839720 127.0.0.1 127.0.0.1 TFTP 36 Acknowledgement, Block: 99
10 8.999919 127.0.0.1 127.0.0.1 TFTP 38 Data Packet, Block: 1 (last)
11 9.000160 127.0.0.1 127.0.0.1 TFTP 548 Error Code, Code: File already exists, Message: A message with the same blocknumber was already received and will therefore be discarded.
```

- Was passiert wenn Server-Port im Client geändert wird?



Aufzeichnen von Loopback: lo0

Anzeigefilter anwenden ... <%%/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000...	127.0.0.1	127.0.0.1	UDP	53	56484 → 2017 Len=21
2	0.0000...	127.0.0.1	127.0.0.1	ICMP	60	Destination unreachable (Port unreachable)

# Agenda

1

2

3

4

5

6

1. TFTP

2. Einführung - Projektbasis

3. Beispiele aus der Umsetzung

4. Write Request und Read Request

5. Error Handling

**6. Fazit**



# 6. Fazit

- Zielvorgaben erfüllt (2.2 Aufgaben Client/Server)
- Senden und Empfangen der Daten funktioniert
- Error Handling wurde behandelt
- Implementierung in Wireshark getestet



# Literaturverzeichnis

## Quellen

- Kurose, J. (2014). *Computernetzwerke – Der Top-Down-Ansatz*. Hallbergmoos: Pearson Deutschland GmbH.
- Prof. Dr. A. Soceanu (2017). *Skripte aus der Vorlesung*
- <http://einstein.informatik.uni-oldenburg.de/rechnernetze/tftp.htm> (zuletzt aufgerufen am: 09.01.2018)
- <http://www.webschmoeker.de/grundlagen/udp-user-datagram-protocol/> (zuletzt aufgerufen am 07.01.2018)
- <https://javapapers.com/java/java-nio-tftp-client/> (zuletzt aufgerufen am 09.01.2018)
- <https://javapapers.com/java/java-tftp-client/> (zuletzt aufgerufen am 10.01.2018)

# Ende

Vielen Dank für Ihre Aufmerksamkeit!

Fragen?