Rechnernetze

Labor 3 – Sockets

Sommersemester 2018

SWB

Prof. Herbert Wiese

Simon Weber - 753693

Sascha Zauner - 753335

Inhaltsverzeichnis

1.1 Nicht nebenläufiger Server, 1 Client-PC

Wichtige Funktionen vom Server

TCP socket : 95: socket()
Registrierung : 113: bind()
Schließt den Socket : 116: close(ss)
Buffer : 127: listen()
Verbindungsaufbau : 150: accept()
Lese Buffer : 219: read()
Schreibe an Client : 270: write()

Wichtige Funktionen des Clients

TCP socket : 112: socket()
Registrierung : 149: bind()
Schließt Socket : 155: close(cs)
Schreibe an Server : 244: write()
Lese Buffer : 262: read()

1.1.1 Ein Server / Ein Client

Verbindundung Client – Server → aus Sicht des Servers

1 0.000000 134.108.8.48 134.108.8.49 TCP 74 34102 → 9009 [SYN] Seq=0 Win=2920 Len=0 MSS=1460 SACK_PERM=1 TSval=5185377 TSecr=0 WS=1
2 0.000057 134.108.8.49 134.108.8.48 TCP 74 9009 → 34102 [SYN, ACK] Seq=0 Ack=1 Win=2896 Len=0 MSS=1460 SACK_PERM=1 TSval=3286591 TSecr=5185377 WS=1
3 0.000258 134.108.8.48 134.108.8.49 TCP 66 34102 → 9009 [ACK] Seq=1 Ack=1 Win=2920 Len=0 TSval=5185377 TSecr=3286591
4 33.166312 134.108.8.48 134.108.8.49 TCP 1514 34102 → 9009 [ACK] Seq=1 Ack=1 Win=2920 Len=1448 TSval=5218544 TSecr=3286591
5 0.000060 134.108.8.49 134.108.8.48 TCP 66 9009 → 34102 [ACK] Seq=1 Ack=1449 Win=2896 Len=0 TSval=3319758 TSecr=5218544
6 0.000185 134.108.8.48 134.108.8.49 TCP 1514 34102 → 9009 [PSH, ACK] Seq=1449 Ack=1 Win=2920 Len=1448 TSval=5218544 TSecr=3286591
7 0.000035 134.108.8.49 134.108.8.48 TCP 66 9009 → 34102 [ACK] Seq=1 Ack=2897 Win=2896 Len=0 TSval=3319758 TSecr=5218544
8 0.000251 134.108.8.48 134.108.8.49 TCP 1514 34102 → 9009 [ACK] Seq=2897 Ack=1 Win=2920 Len=1448 TSval=5218544 TSecr=3319758
9 0.000041 134.108.8.49 134.108.8.48 TCP 66 9009 → 34102 [ACK] Seq=1 Ack=4345 Win=2896 Len=0 TSval=3319758 TSecr=5218544
10 0.000226 134.108.8.48 134.108.8.49 TCP 1514 34102 → 9009 [PSH, ACK] Seq=4345 Ack=1 Win=2920 Len=1448 TSval=5218545 TSecr=3319758

```
11 0.000049       134.108.8.49          134.108.8.48          TCP           66
      9009 → 34102 [ACK] Seq=1 Ack=5793 Win=2896 Len=0     TSval=3319758
      TSecr=5218545
12 0.000277       134.108.8.48          134.108.8.49          TCP           1514
      34102 → 9009 [ACK] Seq=5793 Ack=1 Win=2920 Len=1448  TSval=5218545
      Tsecr=3319758
...
98 0.000006       134.108.8.48          134.108.8.49          TCP           66
      34102 → 9009 [FIN, ACK] Seq=60004 Ack=14001 Win=2920 Len=0       TSval=5256704
      TSecr=3357916
99 0.000031       134.108.8.49          134.108.8.48          TCP           66
      9009 → 34102 [ACK] Seq=14001 Ack=60004 Win=2896 Len=0      TSval=3357917
      TSecr=5256704
100 0.039203       134.108.8.49          134.108.8.48          TCP           66
      9009 → 34102 [ACK] Seq=14001 Ack=60005 Win=2896 Len=0      TSval=3357957
      TSecr=5256704
101 0.960936       134.108.8.49          134.108.8.48          TCP           66
      9009 → 34102 [FIN, ACK] Seq=14001 Ack=60005 Win=2896 Len=0       TSval=3358917
      Tsecr=5256704
```

Paket 1-2: Verbindungsaufbau

Paket 3-97: Datenübertragung

Paket 98-101: Verbingungsabau

1.1.2 Ein Server / Zwei Clients

Three-Way-Handshake | Client A

```
1 0.000000       134.108.8.48          134.108.8.49          TCP       74     34408
      → 9009 [SYN] Seq=0 Win=2920 Len=0 MSS=1460     SACK_PERM=1 TSval=6664361
      TSecr=0 WS=1
2 0.000417       134.108.8.49          134.108.8.48          TCP       74     9009 →
      34408 [SYN, ACK] Seq=0 Ack=1 Win=2896 Len=0    MSS=1460 SACK_PERM=1
      TSval=4765574 TSecr=6664361 WS=1
3 0.000040       134.108.8.48          134.108.8.49          TCP       66     34408
      → 9009 [ACK] Seq=1 Ack=1 Win=2920 Len=0  TSval=6664362 Tsecr=4765574
```

Dynamische Buffer Erweiterung | Client A

```
40 0.000009       134.108.8.48          134.108.8.49          TCP           1514
      [TCP Window Full] 34408 → 9009 [PSH, ACK] Seq=34753 Ack=1  Win=2920 Len=1448
      TSval=6670534 TSecr=4771747
41 0.000788       134.108.8.49          134.108.8.48          TCP           66
      9009 → 34408 [ACK] Seq=1 Ack=36201 Win=2896 Len=0    TSval=4771748
      TSecr=6670534
42 0.000027       134.108.8.48          134.108.8.49          TCP           1514
      34408 → 9009 [ACK] Seq=36201 Ack=1 Win=2920 Len=1448       TSval=6670535
      TSecr=4771748
```

43 0.000009 134.108.8.48 134.108.8.49 TCP 1514
[TCP Window Full] 34408 → 9009 [PSH, ACK] Seq=37649 Ack=1 Win=2920 Len=1448 TSval=6670535 TSecr=4771748
44 0.000747 134.108.8.49 134.108.8.48 TCP 66
9009 → 34408 [ACK] Seq=1 Ack=39097 Win=2896 Len=0 TSval=4771748 TSecr=6670535
45 0.000028 134.108.8.48 134.108.8.49 TCP 1514
34408 → 9009 [ACK] Seq=39097 Ack=1 Win=2920 Len=1448 TSval=6670536 TSecr=4771748
46 0.000010 134.108.8.48 134.108.8.49 TCP 1514
[TCP Window Full] 34408 → 9009 [PSH, ACK] Seq=40545 Ack=1 Win=2920 Len=1448 TSval=6670536 TSecr=4771748
47 0.000737 134.108.8.49 134.108.8.48 TCP 66
9009 → 34408 [ACK] Seq=1 Ack=41993 Win=2896 Len=0 TSval=4771749 TSecr=6670536
48 0.000027 134.108.8.48 134.108.8.49 TCP 1514
34408 → 9009 [ACK] Seq=41993 Ack=1 Win=2920 Len=1448 TSval=6670537 TSecr=4771749
49 0.000010 134.108.8.48 134.108.8.49 TCP 1514
[TCP Window Full] 34408 → 9009 [PSH, ACK] Seq=43441 Ack=1 Win=2920 Len=1448 TSval=6670537 Tsecr=4771749

Three-Way-Handshake | Client B
67 0.039809 134.108.8.49 134.108.8.48 TCP 66
9009 → 34408 [ACK] Seq=1 Ack=60001 Win=2896 Len=0 TSval=4771794 TSecr=6670541
68 13.733394 134.108.8.48 134.108.8.49 TCP 74
34410 → 9009 [SYN] Seq=0 Win=2920 Len=0 MSS=1460 SACK_PERM=1 TSval=6684314 TSecr=0 WS=1
69 0.000407 134.108.8.49 134.108.8.48 TCP 74
9009 → 34410 [SYN, ACK] Seq=0 Ack=1 Win=2896 Len=0 MSS=1460 SACK_PERM=1 TSval=4785527 TSecr=6684314 WS=1

Daten empfangen und Verbindung beenden | Client A
100 0.000027 134.108.8.48 134.108.8.49 TCP 1514
34410 → 9009 [PSH, ACK] Seq=27513 Ack=1 Win=2920 Len=1448 TSval=6690973 TSecr=4792186
101 0.000009 134.108.8.48 134.108.8.49 TCP 1514
[TCP Window Full] 34410 → 9009 [PSH, ACK] Seq=28961 Ack=1 Win=2920 Len=1448 TSval=6690973 TSecr=4792186
102 0.000500 134.108.8.49 134.108.8.48 TCP 66
9009 → 34410 [ACK] Seq=1 Ack=30409 Win=2896 Len=0 TSval=4792187 TSecr=6690973
103 0.000027 134.108.8.48 134.108.8.49 TCP 1514
34410 → 9009 [ACK] Seq=30409 Ack=1 Win=2920 Len=1448 TSval=6690974 TSecr=4792187
104 0.000010 134.108.8.48 134.108.8.49 TCP 1514
[TCP Window Full] 34410 → 9009 [PSH, ACK] Seq=31857 Ack=1 Win=2920 Len=1448 TSval=6690974 TSecr=4792187
105 0.000532 134.108.8.49 134.108.8.48 TCP 66
9009 → 34410 [ACK] Seq=1 Ack=33305 Win=2896 Len=0 TSval=4792187 TSecr=6690974

106 0.000027 134.108.8.48 134.108.8.49 TCP 1514
34410 → 9009 [ACK] Seq=33305 Ack=1 Win=2920 Len=1448 TSval=6690974 TSecr=4792187
107 0.000009 134.108.8.48 134.108.8.49 TCP 1514
[TCP Window Full] 34410 → 9009 [PSH, ACK] Seq=34753 Ack=1 Win=2920 Len=1448 TSval=6690974 TSecr=4792187
108 0.000763 134.108.8.49 134.108.8.48 TCP 66
9009 → 34410 [ACK] Seq=1 Ack=36201 Win=2896 Len=0 TSval=4792188 TSecr=6690974
109 0.000026 134.108.8.48 134.108.8.49 TCP 1514
34410 → 9009 [ACK] Seq=36201 Ack=1 Win=2920 Len=1448 TSval=6690975 TSecr=4792188
110 0.000010 134.108.8.48 134.108.8.49 TCP 1514
[TCP Window Full] 34410 → 9009 [PSH, ACK] Seq=37649 Ack=1 Win=2920 Len=1448 TSval=6690975 Tsecr=4792188

Daten empfangen und Verbindung beenden | Client A

148 0.000016 134.108.8.48 134.108.8.49 TCP 66
34408 → 9009 [ACK] Seq=60001 Ack=13033 Win=2920 Len=0 TSval=6715043 TSecr=4816255
149 0.000591 134.108.8.49 134.108.8.48 TCP 1034
9009 → 34408 [PSH, ACK] Seq=13033 Ack=60001 Win=2896 Len=968 TSval=4816256 TSecr=6715043
150 0.039963 134.108.8.48 134.108.8.49 TCP 66
34408 → 9009 [ACK] Seq=60001 Ack=14001 Win=2920 Len=0 TSval=6715084 TSecr=4816256
151 5.716255 134.108.8.48 134.108.8.49 TCP 69
34408 → 9009 [PSH, ACK] Seq=60001 Ack=14001 Win=2920 Len=3 TSval=6720800 TSecr=4816256
152 0.000021 134.108.8.48 134.108.8.49 TCP 66
34408 → 9009 [FIN, ACK] Seq=60004 Ack=14001 Win=2920 Len=0 TSval=6720800 Tsecr=4816256

Daten empfangen und Verbindung beenden | Client B

170 0.000025 134.108.8.48 134.108.8.49 TCP 66
34410 → 9009 [ACK] Seq=60001 Ack=13033 Win=2920 Len=0 TSval=6735915 TSecr=4837127
171 0.000597 134.108.8.49 134.108.8.48 TCP 1034
9009 → 34410 [PSH, ACK] Seq=13033 Ack=60001 Win=2896 Len=968 TSval=4837128 TSecr=6735915
172 0.039998 134.108.8.48 134.108.8.49 TCP 66
34410 → 9009 [ACK] Seq=60001 Ack=14001 Win=2920 Len=0 TSval=6735956 TSecr=4837128
173 2.461125 134.108.8.48 134.108.8.49 TCP 69
34410 → 9009 [PSH, ACK] Seq=60001 Ack=14001 Win=2920 Len=3 TSval=6738417 TSecr=4837128
174 0.000021 134.108.8.48 134.108.8.49 TCP 66
34410 → 9009 [FIN, ACK] Seq=60004 Ack=14001 Win=2920 Len=0 TSval=6738417 Tsecr=4837128

Warum läuft die Datenübertragung (siehe wireshark-Ausgaben) parallel ab?

Die Wireshark Ausgabe zeigt die nebenläufige Arbeitsweise des Servers auf TCP-Ebene an (TCP-Ebene immer parallel).

Terminal Ausgabe Server

```
SERVER: Nicht nebenlaeufiger Server, Version: 1.3 ; Autor: H.Ws
SERVER: Server-Port = 9009
SERVER: socket 3 : Window-Size = 4096
SERVER: socket 3 : MAXSEG = 536

SERVER: socket 4 : Window-Size = 3000
SERVER: socket 4 : MAXSEG = 1448
SERVER: Socket 4 : Mit Client 134.108.8.48 auf Port 34408 Verbindung aufgenommen
SERVER: Socket 4 : Insgesamt 18 Sekunden auf Client gewartet
SERVER: socket 4 : Jetzt vom Client lesen ? (j/n) j
SERVER: socket 4 : Nachricht von Client:
Aabcdefghijklmnopqrstuvwxyz Babcdefghijklmnopqrstuv ...
SERVER: socket 4 : Anzahl gelesener Zeichen in read = 60000
SERVER: socket 4 : Jetzt zum Client schreiben ? (j/n) j
SERVER: socket 4 : Anzahl geschriebener Zeichen in write = 14000
SERVER: Socket 4 : Verbindung zum Client beendet. Return-code close() = 0

SERVER: socket 4 : Window-Size = 3000
SERVER: socket 4 : MAXSEG = 1448
SERVER: Socket 4 : Mit Client 134.108.8.48 auf Port 34410 Verbindung aufgenommen
SERVER: Socket 4 : Insgesamt 75 Sekunden auf Client gewartet
SERVER: socket 4 : Jetzt vom Client lesen ? (j/n) j
SERVER: socket 4 : Nachricht von Client:
Aabcdefghijklmnopqrstuvwxyz Babcdefghijklmnopqrstuv ...
SERVER: socket 4 : Anzahl gelesener Zeichen in read = 60000
SERVER: socket 4 : Jetzt zum Client schreiben ? (j/n) j
SERVER: socket 4 : Anzahl geschriebener Zeichen in write = 14000
SERVER: Socket 4 : Verbindung zum Client beendet. Return-code close() = 0
```

Woran ist eindeutig erkennbar, dass der Server sequentiell arbeitet?

Der Server bearbeitet zuerst Client A und bearbeitet erst Client B, wenn Client A terminiert wurde.
Er kann also nicht A und B gleichzeitig bearbeiten und muss daher diese nacheinander abarbeiten.

Wo blockiert der Server?

Die Funktion „listen()“ blockiert die Verbindung, da nur eine zugelassen ist.

1.1.3 Ein Server / Ein Client (mit vorzeitiger Beendigung des Clients)

```
1 0.000000       134.108.8.48          134.108.8.49          TCP       74     34432
      → 9009 [SYN] Seq=0 Win=2920 Len=0 MSS=1460     SACK_PERM=1 TSval=7176303
      TSecr=0 WS=1
2 0.000042       134.108.8.49          134.108.8.48          TCP       74     9009 →
      34432 [SYN, ACK] Seq=0 Ack=1 Win=2896 Len=0    MSS=1460 SACK_PERM=1
      TSval=5277516 TSecr=7176303 WS=1
3 0.000258       134.108.8.48          134.108.8.49          TCP       66     34432
      → 9009 [ACK] Seq=1 Ack=1 Win=2920 Len=0  TSval=7176304 TSecr=5277516
```

```
4 13.290289      134.108.8.49          134.108.8.48          TCP      1514   9009 →
      34432 [ACK] Seq=1 Ack=1 Win=2896 Len=1448      TSval=5290807 TSecr=7176304
5 0.000041       134.108.8.49          134.108.8.48          TCP      1514   9009 →
      34432 [PSH, ACK] Seq=1449 Ack=1 Win=2896 Len=1448    TSval=5290807
      TSecr=7176304
6 0.000597       134.108.8.48          134.108.8.49          TCP      66     34432
      → 9009 [ACK] Seq=1 Ack=1449 Win=2920 Len=0     TSval=7189594 TSecr=5290807
7 0.000044       134.108.8.49          134.108.8.48          TCP      1514   9009 →
      34432 [ACK] Seq=2897 Ack=1 Win=2896 Len=1448   TSval=5290808 TSecr=7189594
8 0.000668       134.108.8.48          134.108.8.49          TCP      66     34432
      → 9009 [ACK] Seq=1 Ack=4345 Win=2920 Len=0     TSval=7189595 TSecr=5290807
9 0.000065       134.108.8.49          134.108.8.48          TCP      1514   9009 →
      34432 [PSH, ACK] Seq=4345 Ack=1 Win=2896 Len=1448    TSval=5290808
      TSecr=7189595
10 0.000018       134.108.8.49          134.108.8.48          TCP           1514
      9009 → 34432 [ACK] Seq=5793 Ack=1 Win=2896 Len=1448  TSval=5290808
      TSecr=7189595
11 0.000774       134.108.8.48          134.108.8.49          TCP           66
      34432 → 9009 [ACK] Seq=1 Ack=7241 Win=2920 Len=0     TSval=7189596
      TSecr=5290808
12 0.000064       134.108.8.49          134.108.8.48          TCP           1514
      9009 → 34432 [PSH, ACK] Seq=7241 Ack=1 Win=2896 Len=1448   TSval=5290809
      TSecr=7189596
13 0.000018       134.108.8.49          134.108.8.48          TCP           1514
      9009 → 34432 [ACK] Seq=8689 Ack=1 Win=2896 Len=1448  TSval=5290809
      TSecr=7189596
14 0.000781       134.108.8.48          134.108.8.49          TCP           66
      34432 → 9009 [ACK] Seq=1 Ack=10137 Win=2920 Len=0    TSval=7189597
      TSecr=5290809
15 0.000065       134.108.8.49          134.108.8.48          TCP           1514
      9009 → 34432 [PSH, ACK] Seq=10137 Ack=1 Win=2896 Len=1448  TSval=5290810
      TSecr=7189597
16 0.000018       134.108.8.49          134.108.8.48          TCP           1514
      9009 → 34432 [ACK] Seq=11585 Ack=1 Win=2896 Len=1448       TSval=5290810
      TSecr=7189597
17 0.000711       134.108.8.48          134.108.8.49          TCP           66
      34432 → 9009 [ACK] Seq=1 Ack=13033 Win=2920 Len=0    TSval=7189598
      TSecr=5290810
18 0.000065       134.108.8.49          134.108.8.48          TCP           1034
      9009 → 34432 [PSH, ACK] Seq=13033 Ack=1 Win=2896 Len=968   TSval=5290811
      TSecr=7189598
19 0.039769       134.108.8.48          134.108.8.49          TCP           66
      34432 → 9009 [ACK] Seq=1 Ack=14001 Win=2920 Len=0    TSval=7189638
      TSecr=5290811
20 3.879053       134.108.8.48          134.108.8.49          TCP           69
      34432 → 9009 [PSH, ACK] Seq=1 Ack=14001 Win=2920 Len=3     TSval=7193517
      TSecr=5290811
21 0.000022       134.108.8.48          134.108.8.49          TCP           66
      34432 → 9009 [RST, ACK] Seq=4 Ack=14001 Win=2920 Len=0     TSval=7193517
      TSecr=5290811
22 0.000048       134.108.8.49          134.108.8.48          TCP           66
      9009 → 34432 [ACK] Seq=14001 Ack=4 Win=2896 Len=0    TSval=5294730
      TSecr=7193517
```

```
23 0.000266       134.108.8.48          134.108.8.49          TCP           60
      34432 → 9009 [RST] Seq=4 Win=0 Len=0
```

Warum wird kein PDU mit FIN gesendet?
Der Client empfängt zwar die Nachrichten, lehnt diese aber durch unser zutun ab.

Wozu dient die RST-PDU?
RST-Flags werden nur zurückgegeben, wenn Verbindungen abgebrochen werden sollen, Probleme auftreten oder Verbindungen unerwünscht sind.
In unserem Fall bekommt der Server damit mitgeteilt, dass der Client nichts lesen möchte.

Was passiert mit den Daten des Servers?
Diese befinden sich im Kurzspeicher, werden aber – da wir es nicht gewollt haben – nicht ausgelesen.

1.2 Client-Server-Kommunikation über Tunnel

Wie wurde die MSS berechnet?
1500 Bytes - 20 Bytes [IP] - 20 Bytes [TCP] = 1460
Da wir aber noch Tunneln, 1460 - 24 Bytes [Tunneling] = 1436 Bytes

Was sagt die ICMP Nachricht vom Router?
Der Router sendet eine ICMP Nachricht, dass eine Fragmentierung notwendig ist.

```
7 0.001212       134.108.11.254        134.108.8.49          ICMP      70
      Destination unreachable (Fragmentation needed)
```

Warum soll fragmentiert werden?
Da Ethernet max. 1500 Bytes pro Paket versenden kann.

Wie sieht TCP Segmentierung aus? Ist diese Segmentierung sinnvoll?
Es wird in zwei Datenpakete fragmentiert, einmal mit 1424 Bytes
und einmal mit 24 Bytes, wodurch unnötig Datenverkehr produziert wird.

Mit Fragmentierung:

```
1 0.000000       134.108.8.49          134.108.36.102        ICMP      94
      Destination unreachable (Host administratively prohibited)
2 16.866616      134.108.8.49          134.108.190.10        TCP       74     38892
      → 9009 [SYN] Seq=0 Win=2920 Len=0 MSS=1460     SACK_PERM=1 TSval=84107 TSecr=0
      WS=1
3 0.000491       134.108.190.10        134.108.8.49          TCP       74     9009 →
      38892 [SYN, ACK] Seq=0 Ack=1 Win=2896 Len=0    MSS=1460 SACK_PERM=1
      TSval=2382461473 TSecr=84107 WS=1
4 0.000039       134.108.8.49          134.108.190.10        TCP       66     38892
      → 9009 [ACK] Seq=1 Ack=1 Win=2920 Len=0 TSval=84107  TSecr=2382461473
```

```
5 2.437138       134.108.8.49          134.108.190.10        TCP       1514   38892
      → 9009 [ACK] Seq=1 Ack=1 Win=2920 Len=1448     TSval=86545 TSecr=2382461473
6 0.000015       134.108.8.49          134.108.190.10        TCP       1514   [TCP
      Window Full] 38892 → 9009 [PSH, ACK] Seq=1449 Ack=1  Win=2920 Len=1448
      TSval=86545 TSecr=2382461473
7 0.001212       134.108.11.254        134.108.8.49          ICMP      70
      Destination unreachable (Fragmentation needed)
8 0.000032       134.108.8.49          134.108.190.10        TCP       1490   [TCP
      Retransmission] 38892 → 9009 [ACK] Seq=1 Ack=1       Win=2920 Len=1424
      TSval=86546 TSecr=2382461473
9 0.000012       134.108.8.49          134.108.190.10        TCP       90     [TCP
      Retransmission] 38892 → 9009 [ACK] Seq=1425 Ack=1    Win=2920 Len=24
      TSval=86546 TSecr=2382461473
10 0.000007       134.108.8.49          134.108.190.10        TCP           1490
      [TCP Retransmission] 38892 → 9009 [ACK] Seq=1449 Ack=1     Win=2920 Len=1424
      TSval=86546 TSecr=2382461473
11 0.000006       134.108.8.49          134.108.190.10        TCP           90
      [TCP Window Full] [TCP Retransmission] 38892 → 9009 [PSH,  ACK] Seq=2873 Ack=1
      Win=2920 Len=24 TSval=86546 TSecr=2382461473
12 0.001007       134.108.190.10        134.108.8.49          TCP           66
      9009 → 38892 [ACK] Seq=1 Ack=1425 Win=2896 Len=0     TSval=2382463913
      TSecr=86546
13 0.000031       134.108.8.49          134.108.190.10        TCP           1490
      [TCP Window Full] 38892 → 9009 [ACK] Seq=2897 Ack=1  Win=2920 Len=1424
      TSval=86547 TSecr=2382463913
14 0.000009       134.108.190.10        134.108.8.49          TCP           66
      9009 → 38892 [ACK] Seq=1 Ack=2873 Win=2896 Len=0     TSval=2382463913
      TSecr=86546
```

Ohne Fragmentierung:

```
1 0.000000       134.108.8.49          134.108.190.10        TCP       74     38896
      → 9009 [SYN] Seq=0 Win=2872 Len=0 MSS=1436     SACK_PERM=1 TSval=566534
      TSecr=0 WS=1
2 0.000490       134.108.190.10        134.108.8.49          TCP       74     9009 →
      38896 [SYN, ACK] Seq=0 Ack=1 Win=2896 Len=0    MSS=1460 SACK_PERM=1
      TSval=2382943909 TSecr=566534 WS=1
3 0.000026       134.108.8.49          134.108.190.10        TCP       66     38896
      → 9009 [ACK] Seq=1 Ack=1 Win=2872 Len=0 TSval=566535       TSecr=2382943909
4 1.548859       134.108.8.49          134.108.190.10        TCP       1490   38896
      → 9009 [ACK] Seq=1 Ack=1 Win=2872 Len=1424     TSval=568084 TSecr=2382943909
5 0.000021       134.108.8.49          134.108.190.10        TCP       1490   38896
      → 9009 [PSH, ACK] Seq=1425 Ack=1 Win=2872 Len=1424   TSval=568084
      TSecr=2382943909
6 0.000997       134.108.190.10        134.108.8.49          TCP       66     9009 →
      38896 [ACK] Seq=1 Ack=1425 Win=2896 Len=0      TSval=2382945458 TSecr=568084
7 0.000032       134.108.8.49          134.108.190.10        TCP       1490   38896
      → 9009 [ACK] Seq=2849 Ack=1 Win=2872 Len=1424  TSval=568085 TSecr=2382945458
8 0.001039       134.108.190.10        134.108.8.49          TCP       66     9009 →
      38896 [ACK] Seq=1 Ack=4273 Win=2896 Len=0      TSval=2382945460 TSecr=568084
9 0.000028       134.108.8.49          134.108.190.10        TCP       1490   38896
      → 9009 [PSH, ACK] Seq=4273 Ack=1 Win=2872 Len=1424   TSval=568086
      TSecr=2382945460
```

```
10 0.000011       134.108.8.49          134.108.190.10        TCP           1490
      38896 → 9009 [ACK] Seq=5697 Ack=1 Win=2872 Len=1424  TSval=568086
      TSecr=2382945460
11 0.001143       134.108.190.10        134.108.8.49          TCP           66
      9009 → 38896 [ACK] Seq=1 Ack=7121 Win=2896 Len=0     TSval=2382945461
      TSecr=568086
12 0.000027       134.108.8.49          134.108.190.10        TCP           1490
      38896 → 9009 [PSH, ACK] Seq=7121 Ack=1 Win=2872 Len=1424   TSval=568087
      TSecr=2382945461
13 0.000010       134.108.8.49          134.108.190.10        TCP           1490
      38896 → 9009 [ACK] Seq=8545 Ack=1 Win=2872 Len=1424  TSval=568087
      TSecr=2382945461
14 0.001149       134.108.190.10        134.108.8.49          TCP           66
      9009 → 38896 [ACK] Seq=1 Ack=9969 Win=2896 Len=0     TSval=2382945462
      TSecr=568087
15 0.000028       134.108.8.49          134.108.190.10        TCP           1490
      38896 → 9009 [PSH, ACK] Seq=9969 Ack=1 Win=2872 Len=1424   TSval=568088
      TSecr=2382945462
```

1.3 Zustand half closed untersuchen

```
1 0.000000       134.108.8.48          134.108.8.49          TCP       74     34624
      → 9009 [SYN] Seq=0 Win=2920 Len=0 MSS=1460     SACK_PERM=1 TSval=12840349
      TSecr=0 WS=1
2 0.001360       134.108.8.49          134.108.8.48          TCP       74     9009 →
      34624 [SYN, ACK] Seq=0 Ack=1 Win=2896 Len=0    MSS=1460 SACK_PERM=1
      TSval=891094 TSecr=12840349 WS=1
3 0.000246       134.108.8.48          134.108.8.49          TCP       66     34624
      → 9009 [ACK] Seq=1 Ack=1 Win=2920 Len=0  TSval=12840351 TSecr=891094
4 56.893367      134.108.8.48          134.108.8.49          TCP       66     34624
      → 9009 [FIN, ACK] Seq=1 Ack=1 Win=2920 Len=0   TSval=12897245 TSecr=891094
5 0.000130       134.108.8.49          134.108.8.48          TCP       66     9009 →
      34624 [ACK] Seq=1 Ack=2 Win=2896 Len=0 TSval=947989  TSecr=12897245
6 1.000034       134.108.8.49          134.108.8.48          TCP       66     9009 →
      34624 [FIN, ACK] Seq=1 Ack=2 Win=2896 Len=0    TSval=948989 TSecr=12897245
7 0.000214       134.108.8.48          134.108.8.49          TCP       66     34624
      → 9009 [ACK] Seq=2 Ack=2 Win=2920 Len=0  TSval=12898245 Tsecr=948989
```

In welche Richtung läuft die Übertragung?
Der Client sendet [FIN, ACK] und bekommt fortlaufend Daten, bis der Server ein [ACK] sendet und darauf ein weiteres [FIN, ACK] und [ACK] zum beenden vom Server bekommt.

Wozu wird der Zustand „half close“ gebraucht?
Da der Client die Verbindung vorzeitig beendet hat, haben wir einen „half-closed“ - Status.

1.4 Client mit „bind“

Warum bindet man bei Client die Ports mit bind()?
Mithilfe von „bind“ binden wir den Socket an eine Portnummer.

Was passiert bei Fehlerbehandlung?
Der Zusatz „errorhandling“, schließt dazu die Verbindung, falls der verbundene Port von einem anderen Programm benutzt wird.

Welchen Port bekommt der Client ohne Fehlerbehandlung?
Wenn „errorhandling“ nicht verwendet wurde, verwendet das Programm einen anderen Port.

Bind werte:
Erfolgreich == 0.
Ansonsten == negativer wert (-1)

Ohne Fehlerbehandlung:

```
1 0.000000      134.108.8.48        134.108.8.49        TCP     74    9010 →
    9009 [SYN] Seq=0 Win=2920 Len=0 MSS=1460 SACK_PERM=1 TSval=13320050 TSecr=0
    WS=1
2 0.000425      134.108.8.49        134.108.8.48        TCP     74    9009 →
    9010 [SYN, ACK] Seq=0 Ack=1 Win=2896 Len=0 MSS=1460  SACK_PERM=1 TSval=1370795
    TSecr=13320050 WS=1
3 0.000044      134.108.8.48        134.108.8.49        TCP     66    9010 →
    9009 [ACK] Seq=1 Ack=1 Win=2920 Len=0    TSval=13320051 TSecr=1370795
4 21.559243     134.108.8.48        134.108.8.49        TCP     74    34640
    → 9009 [SYN] Seq=0 Win=2920 Len=0 MSS=1460     SACK_PERM=1 TSval=13341610
    TSecr=0 WS=1
5 0.000413      134.108.8.49        134.108.8.48        TCP     74    9009 →
    34640 [SYN, ACK] Seq=0 Ack=1 Win=2896 Len=0    MSS=1460 SACK_PERM=1
    TSval=1392354 TSecr=13341610 WS=1
6 0.000034      134.108.8.48        134.108.8.49        TCP     66    34640
    → 9009 [ACK] Seq=1 Ack=1 Win=2920 Len=0  TSval=13341610 TSecr=1392354
7 43.845897     134.108.8.48        134.108.8.49        TCP     69    9010 →
    9009 [PSH, ACK] Seq=1 Ack=1 Win=2920 Len=3     TSval=13385457 TSecr=1370795
8 0.000023      134.108.8.48        134.108.8.49        TCP     66    9010 →
    9009 [FIN, ACK] Seq=4 Ack=1 Win=2920 Len=0     TSval=13385457 TSecr=1370795
9 0.000388      134.108.8.49        134.108.8.48        TCP     66    9009 →
    9010 [ACK] Seq=1 Ack=4 Win=2896 Len=0 TSval=1436201  TSecr=13385457
10 0.039677      134.108.8.49        134.108.8.48        TCP          66
    9009 → 9010 [ACK] Seq=1 Ack=5 Win=2896 Len=0 TSval=1436241
    TSecr=13385457
11 2.192734      134.108.8.48        134.108.8.49        TCP          69
    34640 → 9009 [PSH, ACK] Seq=1 Ack=1 Win=2920 Len=3   TSval=13387690
    TSecr=1392354
12 0.000024      134.108.8.48        134.108.8.49        TCP          66
    34640 → 9009 [FIN, ACK] Seq=4 Ack=1 Win=2920 Len=0   TSval=13387690
    TSecr=1392354
13 0.000390      134.108.8.49        134.108.8.48        TCP          66
    9009 → 34640 [ACK] Seq=1 Ack=4 Win=2896 Len=0  TSval=1438434 TSecr=13387690
```

```
14 0.039864       134.108.8.49          134.108.8.48          TCP           66
      9009 → 34640 [ACK] Seq=1 Ack=5 Win=2896 Len=0  TSval=1438474 TSecr=13387690
15 3.016270       134.108.8.49          134.108.8.48          TCP           66
      9009 → 9010 [FIN, ACK] Seq=1 Ack=5 Win=2896 Len=0    TSval=1441490
      TSecr=13385457
16 0.000034       134.108.8.48          134.108.8.49          TCP           66
      9010 → 9009 [ACK] Seq=5 Ack=2 Win=2920 Len=0   TSval=13390746 TSecr=1441490
17 5.807920       134.108.8.49          134.108.8.48          TCP           66
      9009 → 34640 [FIN, ACK] Seq=1 Ack=5 Win=2896 Len=0   TSval=1447298
      TSecr=13387690
18 0.000029       134.108.8.48          134.108.8.49          TCP           66
      34640 → 9009 [ACK] Seq=5 Ack=2 Win=2920 Len=0  TSval=13396554 TSecr=1447298
19 79.473906      134.108.8.49          224.0.0.251           MDNS          133
      Standard query response 0x0000 PTR _workstation._tcp.local      PTR
      _ssh._tcp.local
```

Mit Fehlerbehandlung:

```
1 0.000000       134.108.8.48          134.108.8.49          TCP      74     9010 →
      9009 [SYN] Seq=0 Win=2920 Len=0 MSS=1460 SACK_PERM=1 TSval=14010900 TSecr=0
      WS=1
2 0.000402       134.108.8.49          134.108.8.48          TCP      74     9009 →
      9010 [SYN, ACK] Seq=0 Ack=1 Win=2896 Len=0 MSS=1460  SACK_PERM=1 TSval=2061644
      TSecr=14010900 WS=1
3 0.000045       134.108.8.48          134.108.8.49          TCP      66     9010 →
      9009 [ACK] Seq=1 Ack=1 Win=2920 Len=0    TSval=14010901 TSecr=2061644
4 25.149930      134.108.8.48          134.108.8.49          TCP      69     9010 →
      9009 [PSH, ACK] Seq=1 Ack=1 Win=2920 Len=3     TSval=14036051 TSecr=2061644
5 0.000021       134.108.8.48          134.108.8.49          TCP      66     9010 →
      9009 [FIN, ACK] Seq=4 Ack=1 Win=2920 Len=0     TSval=14036051 TSecr=2061644
6 0.000381       134.108.8.49          134.108.8.48          TCP      66     9009 →
      9010 [ACK] Seq=1 Ack=4 Win=2896 Len=0 TSval=2086795  TSecr=14036051
7 0.039776       134.108.8.49          134.108.8.48          TCP      66     9009 →
      9010 [ACK] Seq=1 Ack=5 Win=2896 Len=0 TSval=2086835  TSecr=14036051
8 6.380256       134.108.8.49          134.108.8.48          TCP      66     9009 →
      9010 [FIN, ACK] Seq=1 Ack=5 Win=2896 Len=0     TSval=2093215 TSecr=14036051
9 0.000028       134.108.8.48          134.108.8.49          TCP      66     9010 →
      9009 [ACK] Seq=5 Ack=2 Win=2920 Len=0    TSval=14042471 Tsecr=2093215
```

Bind mit Fehlerbehandlung:

```
./client_s2.out 134.108.8.49 9009

CLIENT: Version: 1.3 ; Autor: H.Ws
CLIENT: Server-Port = 9009
CLIENT: addr = 0.0.0.0 ; Gebundener Port = 9010
CLIENT: Fehler bei (bind), Return-Code = -1
CLIENT: Fehler bei bind, fester Port belegt
: Address already in use
```

1.5 Nebenläufiger Server, 2 Clients

Der Nebenläufige (Concurrent) Server kann mehrere Clients zugleich bedienen. Dies kann man gut im Wiresharktrace, als auch im Server output, sehen..

```
11 5.930079       134.108.8.48          134.108.8.49          TCP           74
      34662 → 9009 [SYN] Seq=0 Win=14600 Len=0 MSS=1460    SACK_PERM=1
      TSval=14334277 TSecr=0 WS=128
12 0.000050       134.108.8.49          134.108.8.48          TCP           74
      9009 → 34662 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0  MSS=1460 SACK_PERM=1
      TSval=2385021 TSecr=14334277 WS=128
13 0.000255       134.108.8.48          134.108.8.49          TCP           66
      34662 → 9009 [ACK] Seq=1 Ack=1 Win=14720 Len=0       TSval=14334278
      TSecr=2385021
14 14.775841      134.108.8.48          134.108.8.49          TCP           74
      34664 → 9009 [SYN] Seq=0 Win=14600 Len=0 MSS=1460    SACK_PERM=1
      TSval=14349053 TSecr=0 WS=128
15 0.000044       134.108.8.49          134.108.8.48          TCP           74
      9009 → 34664 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0  MSS=1460 SACK_PERM=1
      TSval=2399797 TSecr=14349053 WS=128
16 0.000259       134.108.8.48          134.108.8.49          TCP           66
      34664 → 9009 [ACK] Seq=1 Ack=1 Win=14720 Len=0       TSval=14349054
      TSecr=2399797
17 17.014150      134.108.8.48          134.108.8.49          TCP           1514
      34664 → 9009 [ACK] Seq=1 Ack=1 Win=14720 Len=1448    TSval=14366068
      TSecr=2399797
18 0.000075       134.108.8.49          134.108.8.48          TCP           66
      9009 → 34664 [ACK] Seq=1 Ack=1449 Win=17408 Len=0    TSval=2416811
      TSecr=14366068
19 0.000167       134.108.8.48          134.108.8.49          TCP           1618
      34664 → 9009 [PSH, ACK] Seq=1449 Ack=1 Win=14720 Len=1552  TSval=14366068
      TSecr=2399797
20 0.000044       134.108.8.49          134.108.8.48          TCP           66
      9009 → 34664 [ACK] Seq=1 Ack=3001 Win=20352 Len=0    TSval=2416812
      TSecr=14366068
21 0.000105       134.108.8.49          134.108.8.48          TCP           1514
      9009 → 34664 [ACK] Seq=1 Ack=3001 Win=20352 Len=1448       TSval=2416812
      TSecr=14366068
22 0.000010       134.108.8.49          134.108.8.48          TCP           618
      9009 → 34664 [PSH, ACK] Seq=1449 Ack=3001 Win=20352  Len=552 TSval=2416812
      TSecr=14366068
23 0.000651       134.108.8.48          134.108.8.49          TCP           66
      34664 → 9009 [ACK] Seq=3001 Ack=2001 Win=17536 Len=0       TSval=14366069
      TSecr=2416812
24 15.606927      134.108.8.48          134.108.8.49          TCP           1514
      34662 → 9009 [ACK] Seq=1 Ack=1 Win=14720 Len=1448    TSval=14381676
      TSecr=2385021
25 0.000074       134.108.8.49          134.108.8.48          TCP           66
      9009 → 34662 [ACK] Seq=1 Ack=1449 Win=17408 Len=0    TSval=2432419
      TSecr=14381676
```

Terminal Server Ausgabe:

```
[sazait00@itpc3110 sock_tcp]$ ./server_n.out 9009
SERVER_N: PID = 7069 : Nebenlaeufiger Server, Version: 1.3 ; Autor: H.Ws
SERVER_N 17:03:19.5 > PID = 7069 : server_port = 9009

SERVER_N 17:03:26.7 > PID = 7069 ; Parent-Socket = 3 : Mit Client 134.108.8.48 auf
Port 34662 Verbindung aufgenommen

SERVER_N 17:03:26.7 > PID = 7070 : Local socket  in child  = 4

SERVER_N 17:03:40.8 > PID = 7069 ; Parent-Socket = 3 : Mit Client 134.108.8.48 auf
Port 34664 Verbindung aufgenommen

SERVER_N 17:03:40.8 > PID = 7071 : Local socket  in child  = 4

SERVER_N 17:03:57.8 > PID = 7071 : clientport 34664 : Nachricht von Client:
AabcdefghijklmnopqrstuvwxyzBabcdefghijkl ...
SERVER_N 17:03:57.8 > PID = 7071 : clientport 34664 : Anzahl gelesener Zeichen in
read = 1448

SERVER_N 17:03:57.8 > PID = 7071 : clientport 34664 : Nachricht von Client:
qrstuvwxyzCabcdefghijklmnopqrstuvwxyzDab ...
SERVER_N 17:03:57.8 > PID = 7071 : clientport 34664 : Anzahl gelesener Zeichen in
read = 1552
SERVER_N 17:03:57.8 > PID = 7071 : clientport 34664 : Anzahl geschriebener Zeichen
in write = 2000

SERVER_N 17:04:13.4 > PID = 7070 : clientport 34662 : Nachricht von Client:
AabcdefghijklmnopqrstuvwxyzBabcdefghijkl ...
SERVER_N 17:04:13.4 > PID = 7070 : clientport 34662 : Anzahl gelesener Zeichen in
read = 1448

SERVER_N 17:04:13.4 > PID = 7070 : clientport 34662 : Nachricht von Client:
qrstuvwxyzCabcdefghijklmnopqrstuvwxyzDab ...
SERVER_N 17:04:13.4 > PID = 7070 : clientport 34662 : Anzahl gelesener Zeichen in
read = 1552
SERVER_N 17:04:13.4 > PID = 7070 : clientport 34662 : Anzahl geschriebener Zeichen
in write = 2000

SERVER_N 17:04:26.9 > PID = 7070 : clientport 34662 : Nachricht von Client:
AabcdefghijklmnopqrstuvwxyzBabcdefghijkl ...
SERVER_N 17:04:26.9 > PID = 7070 : clientport 34662 : Anzahl gelesener Zeichen in
read = 1448

SERVER_N 17:04:26.9 > PID = 7070 : clientport 34662 : Nachricht von Client:
qrstuvwxyzCabcdefghijklmnopqrstuvwxyzDab ...
SERVER_N 17:04:26.9 > PID = 7070 : clientport 34662 : Anzahl gelesener Zeichen in
read = 1552
SERVER_N 17:04:26.9 > PID = 7070 : clientport 34662 : Anzahl geschriebener Zeichen
in write = 2000
SERVER_N 17:04:27.9 > PID = 7070 : Verbindung mit Client 134.108.8.48 auf Port 34662
beendet
```

```
SERVER_N 17:04:30.6 > PID = 7071 : clientport 34664 : Nachricht von Client:
AabcdefghijklmnopqrstuvwxyzBabcdefghijkl ...
SERVER_N 17:04:30.6 > PID = 7071 : clientport 34664 : Anzahl gelesener Zeichen in
read = 1448

SERVER_N 17:04:30.6 > PID = 7071 : clientport 34664 : Nachricht von Client:
qrstuvwxyzCabcdefghijklmnopqrstuvwxyzDab ...
SERVER_N 17:04:30.6 > PID = 7071 : clientport 34664 : Anzahl gelesener Zeichen in
read = 1552
SERVER_N 17:04:30.6 > PID = 7071 : clientport 34664 : Anzahl geschriebener Zeichen
in write = 2000
SERVER_N 17:04:31.6 > PID = 7071 : Verbindung mit Client 134.108.8.48 auf Port 34664
beendet
```

2.1 Requester und Responder im gleichen Subnet

Erklären Sie den Ablauf bei UDP:
Bei UDP werden die Ports mitgesendet, da UDP ein verbindungsloses, ungesichertes/ungeschütztes Protokoll ist

Was ist anders zu TCP?
Ob das Paket beim Empfänger überhaupt angekommen und dabei noch fehlerlos ist, wird nicht überprüft, wodurch nur Portnummer und Daten übertragen werden. Bei auftretenden Fehlern wird die Nachricht nicht erneut gesendet, sondern geht einfach unter.

```
1 0.000000       134.108.8.49          134.108.8.48          IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=0, ID=afd6)      [Reassembled in #3]
2 0.000077       134.108.8.49          134.108.8.48          IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=1480, ID=afd6)   [Reassembled in #3]
3 0.000007       134.108.8.49          134.108.8.48          UDP      82     9010 →
      9009 Len=3000
4 0.000104       134.108.8.49          134.108.8.48          IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=0, ID=afd7)      [Reassembled in #6]
5 0.000143       134.108.8.49          134.108.8.48          IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=1480, ID=afd7)   [Reassembled in #6]
6 0.000011       134.108.8.49          134.108.8.48          UDP      82     9010 →
      9009 Len=3000
7 19.033556      134.108.8.48          134.108.8.49          IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=0, ID=d011)      [Reassembled in #9]
8 0.000010       134.108.8.48          134.108.8.49          IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=1480, ID=d011)   [Reassembled in #9]
9 0.000003       134.108.8.48          134.108.8.49          UDP      82     9009 →
      9010 Len=3000
```

Die zwei Rechner führen im gleichen Subnet die beiden Programme aus. Der Wireshark-Trace zeigt, dass auf der IP-Ebene die Fragmentierung geschieht, nicht bei UDP.

2.2 Requester und Responder in verschiedenen Subnetzen

Welches Protokoll führt hier die Fragmentierung durch?
Wie man im Wiresharktrace sieht, wird nicht auf UDP-Level sondern auf IP-Level fragmentiert.

```
1 0.000000       134.108.8.48          134.108.190.10        IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=0, ID=4e8f)     [Reassembled in #3]
2 0.000009       134.108.8.48          134.108.190.10        IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=1480, ID=4e8f)  [Reassembled in #3]
3 0.000002       134.108.8.48          134.108.190.10        UDP      82     9009 →
      9010 Len=3000
4 0.000020       134.108.8.48          134.108.190.10        IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=0, ID=4e90)     [Reassembled in #6]
5 0.000006       134.108.8.48          134.108.190.10        IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=1480, ID=4e90)  [Reassembled in #6]
6 0.000002       134.108.8.48          134.108.190.10        UDP      82     9009 →
      9010 Len=3000
7 7.904848       134.108.190.10        134.108.8.48          IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=0, ID=43a8)     [Reassembled in #9]
8 0.000121       134.108.190.10        134.108.8.48          IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=1480, ID=43a8)  [Reassembled in #9]
9 0.000005       134.108.190.10        134.108.8.48          UDP      82     9010 →
      9009 Len=3000
```

TTL = 1:

```
1 0.000000       134.108.8.48          134.108.190.10        IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=0, ID=4e93)     [Reassembled in #3]
2 0.000014       134.108.8.48          134.108.190.10        IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=1480, ID=4e93)  [Reassembled in #3]
3 0.000004       134.108.8.48          134.108.190.10        UDP      82     9009 →
      9010 Len=3000
4 0.000030       134.108.8.48          134.108.190.10        IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=0, ID=4e94)     [Reassembled in #6]
5 0.000028       134.108.8.48          134.108.190.10        IPv4     1514
      Fragmented IP protocol (proto=UDP 17, off=1480, ID=4e94)  [Reassembled in #6]
6 0.000004       134.108.8.48          134.108.190.10        UDP      82     9009 →
      9010 Len=3000
7 0.001028       134.108.11.254        134.108.8.48          ICMP     70     Time-
      to-live exceeded (Time to live exceeded in transit)
8 0.000318       134.108.11.254        134.108.8.48          ICMP     70     Time-
      to-live exceeded (Time to live exceeded in transit)
```

Mit TTL = 1, da TTL zu klein ist, sendet der Router ein „Time-to-live exceeded“ zurück