

**Aufgabe 1:**

Um ein beliebiges IPv4-Protokoll zu finden, kann der Suchstring „ip“ verwendet werden. Ein beliebiges Paket ist:

```

5375 108.770196 52.114.244.3 192.168.178.175 TLSv1.2 100 Application Data
5376 108.819206 192.168.178.175 52.114.244.3 TCP 54 21144 → 443 [ACK] Seq=115 Ack=93 Win=511 Len=0
5377 108.924285 162.159.135.234 192.168.178.175 TLSv1.2 213 Application Data
5378 108.975366 192.168.178.175 162.159.135.234 TCP 54 21253 → 443 [ACK] Seq=163 Ack=5055 Win=513 Len=0
Frame 5375: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface \Device\NPF_{DABA4F92-5DAC-4457-ACCF-367E77D81BAA}, id
Ethernet II, Src: AVM_14:b2:03 (9c:c7:a6:14:b2:03), Dst: MicroStarINT_8b:2c:ff (d8:bb:c1:8b:2c:ff)
Internet Protocol Version 4, Src: 52.114.244.3, Dst: 192.168.178.175
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 86
  Identification: 0x2375 (9077)
  010. .... = Flags: 0x2, Don't fragment
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 118
  Protocol: TCP (6)
  Header Checksum: 0x455f [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 52.114.244.3
  Destination Address: 192.168.178.175
Transmission Control Protocol, Src Port: 443, Dst Port: 21144, Seq: 47, Ack: 115, Len: 46
Transport Layer Security

```

Es können die folgenden Daten ausgelesen werden:

- Version: 4
- Header Length: 20 bytes (5)
- Identification: 0x2375 (9077)
- Time to Live: 118
- Protocol: TCP (6)
- Header Checksum: 0x455f [validation disabled]
- Source Address: 52.114.244.3
- Destination Address: 192.168.178.175
- Differentiated Services Field: 0x00 (In der VL als „tos“ betitelt)

Für UDP wird analog vorgefahren:

```

124... 222.550880 192.168.178.175 192.168.178.175 UDP 718 51933 → 3702 Len=656
124... 223.161179 192.168.178.175 192.168.178.175 DNS 97 Standard query 0xf1a2 A package-search.services.jetbrains.com
124... 223.175242 192.168.178.175 192.168.178.175 DNS 145 Standard query response 0xf1a2 A package-search.services.jetbrains.com
Frame 12400: 718 bytes on wire (5744 bits), 718 bytes captured (5744 bits) on interface \Device\NPF_{DABA4F92-5DAC-4457-ACCF-367E77D81BAA}, id
Ethernet II, Src: MicroStarINT_8b:2c:ff (d8:bb:c1:8b:2c:ff), Dst: IPv6mcast_0c (33:33:00:00:00:0c)
Internet Protocol Version 6, Src: fe80::4a11:e70f:368b:11c3, Dst: ff02::c
User Datagram Protocol, Src Port: 51933, Dst Port: 3702
  Source Port: 51933
  Destination Port: 3702
  Length: 664
  Checksum: 0x3989 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 69]
  [Timestamps]
  UDP payload (656 bytes)
Data (656 bytes)

```

Es können die folgenden Daten ausgelesen werden:

- Source Port: 51933
- Destination Port: 3702
- Length: 664
- Checksum: 0x3989 [unverified]

Für TCP wird analog verfahren:

```

124. 223.213508 192.168.178.175 51.21.117.220 TCP 54 21415 → 443 [ACK] Seq=1 Ack=1 Win=132096 Len=0
124. 223.213508 192.168.178.175 51.21.117.220 TLSv1.2 471 Client Hello (SHA256, cipher: aes128gcm, comp: none)
Frame 12408: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{DABA4F92-5DAC-4457-ACCF-367E77D81BAA}, id 0
Ethernet II, Src: MicroStarINT 8b:2c:ff (d8:bb:c1:8b:2c:ff), Dst: AVM_14:b2:03 (9c:c7:a6:14:b2:03)
Internet Protocol Version 4, Src: 192.168.178.175, Dst: 51.21.117.220
Transmission Control Protocol, Src Port: 21415, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
  Source Port: 21415
  Destination Port: 443
  [Stream index: 94]
  [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 0]
  Sequence Number: 1 (relative sequence number)
  Sequence Number (raw): 2931986968
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 1 (relative ack number)
  Acknowledgment number (raw): 570316364
  0101 .... = Header Length: 20 bytes (5)
  Flags: 0x010 (ACK)
  Window: 516
  [Calculated window size: 132096]
  [Window size scaling factor: 256]
  Checksum: 0x1c64 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  [Timestamps]
  [SEQ/ACK analysis]

```

Es können die folgenden Daten ausgelesen werden:

- Source Port: 21415
- Destination Port: 443
- Sequence Number: 1 (relative sequence number)
- Sequence Number (raw): 2931986968
- Acknowledgment Number: 1 (relative ack number)
- Acknowledgment number (raw): 570316364
- Window: 516
- Checksum: 0x1c64 [unverified]
- Urgent Pointer: 0

## Aufgabe 2:

Bei der Adresse 103.161.122.83 handelt es sich um eine IPv4-Adresse.

Die 18 gibt an, dass es sich hier um die CIDR-Notation handelt, das bedeutet, dass die ersten 18 Bits der Subnetzmaske für das Netzwerk verwendet werden und die restlichen 14 Bits werden für Hosts im Netzwerk verwendet.

Die **Subnetzmaske** kann wie folgt ermittelt werden:

Mit der CIDR-Notation geht für die Subnetzmaske einher, dass die ersten 18 Bits auf 1 gesetzt, die folgenden 14 Bits auf 0 gesetzt werden. Die 32 Bits werden dann in 8 Bit-Blöcke unterteilt, damit die Subnetzmaske ermittelt werden kann:

11111111 = 255

11111111 = 255

11000000 = 192

00000000 = 0

Die Subnetzmaske ist also: 255.255.192.0

Die **Netzwerkadresse** erhält man, wenn man ein logisches UND zwischen IP-Adresse und der Subnetzmaske nutzt.

IP-Adresse: 103.161.122.83  $\rightarrow$  01100111.10100001.01111010.01010011

Subnetzmaske: 255.255.192.0  $\rightarrow$  11111111.11111111.11000000.00000000

Durch ein logisches UND der beiden erhalten wir:

01100111.10100001.01000000.00000000  $\rightarrow$  103.161.64.0

Die Broadcastadresse erhält man, indem die Host-Bits auf 1 gesetzt werden (die letzten 14 Bits der Adresse):

Netzwerkadresse: 103.161.64.0  $\rightarrow$  01100111.10100001.01000000.00000000

Setzen der Host-Bits auf 1: 01100111.10100001.01111111.11111111

Das ergibt die Broadcastadresse:

01100111.10100001.01111111.11111111  $\rightarrow$  103.161.127.255

Um herauszufinden, ob 103.161.122.83/18 und 103.161.193.83/18 im gleichen Netzwerk liegen, werden die Netzwerkadressen ermittelt.

Aus der vorigen Beschreibung ist bekannt, dass die Subnetzmaske für /18 wie folgt ist:

255.255.192.0

Damit kann die Netzwerkadresse für 103.161.122.83/18 berechnet werden, diese ist nach obigen Verfahren:

103.161.64.0

Gleiches wird durchgeführt für 103.161.193.83/18, wir erhalten die Netzwerkadresse:

103.161.192.0

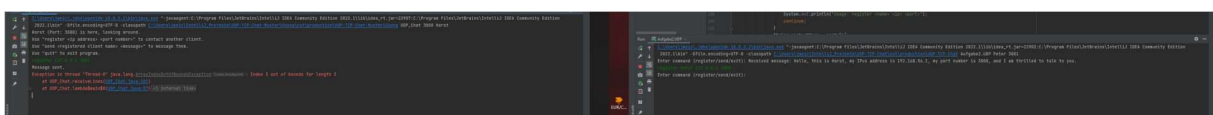
Daraus resultiert, dass 103.161.122.83/18 und 103.161.193.83/18 **nicht** im gleichen Netzwerk liegen, da sie nicht identisch sind.

### Aufgabe 3:

**UDP:** Problematisch ist der Verbindungsaufbau. In meinem Versuch nutze ich die Implementierung von Herrn Sturm und meine eigene. Die Registrierung eines Chatteilnehmers von meiner Implementierung (Peter) bei der Sturm-Implementierung (Horst) ist kein Problem:



Versuche ich jedoch den Sturm-Chatpartner bei mir zu registrieren, so wird eine `IndexOutOfBoundsException` in der Sturm-Implementierung geworfen:



Dies liegt an unterschiedlichen Registrierungsimplementierungen. Das Verfahren von Sturm erhält 2 Argumente und registriert sich dann bei mir. Meine Implementierung nutzt 3 Argumente und registriert sich dann bei der Sturm-Implementierung. Problematisch ist hier, dass die Sturm-Implementierung nur 2 Argumente erwartet und daher eine IOOB geworfen wird.

Verläuft die Registrierung einwandfrei, dann macht das Senden auch kein Problem. Wichtig ist, dass *register* und *send* bei beiden Implementierungen gleich aufgebaut sind.

**TCP:** Ebenso wie bei UDP ist der ausschlaggebende Punkt, dass die Registrierung bei beiden Implementierungen gleich ist. Steht die Registrierung, so kann gechattet werden.