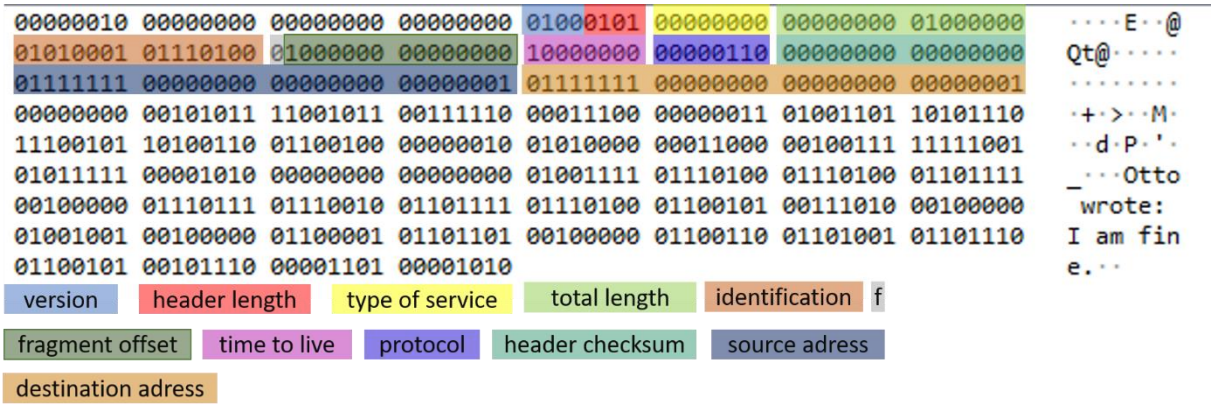
Rechnernetze, Übungsblatt 4, Sommer 2024

Aufgabe 1

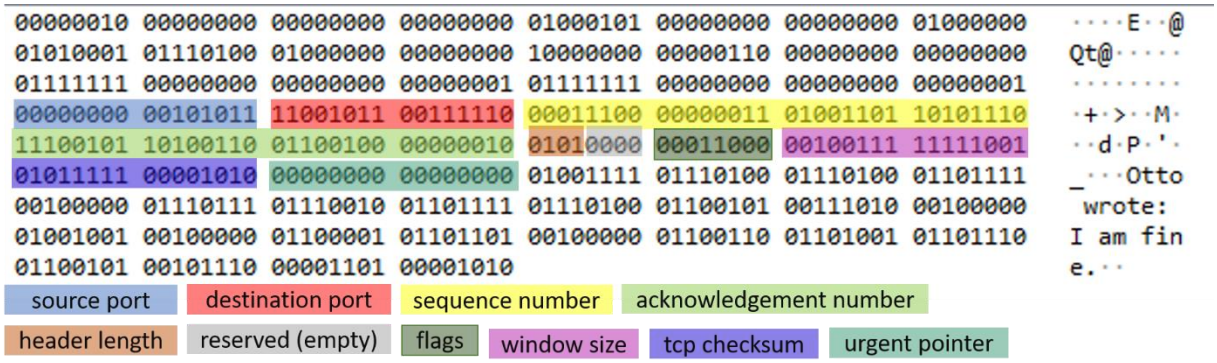
IPv4-Header

Im Folgenden handelt es sich um das Paket 74 der Datei chatTCP.pcapng:



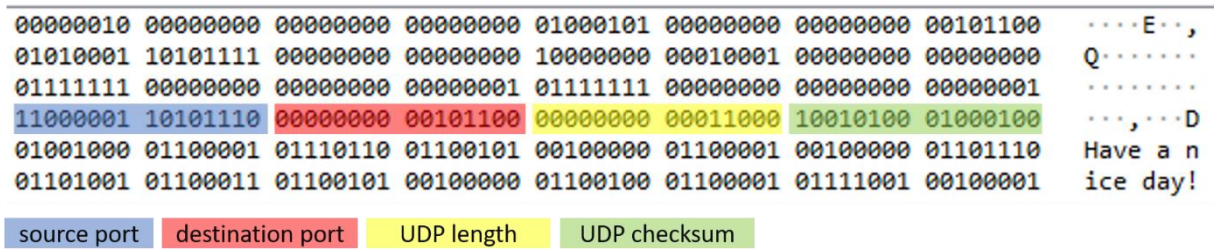
TCP-Header

Im Folgenden handelt es erneut um das Paket 74 der Datei chatTCP.pcapng:



UDP-Header

Im Folgenden handelt es sich um das Paket 69 der Datei ncUDP.pcapng:



Aufgabe 2

Bei 103.161.122.83 handelt es sich um eine IP-Adresse im Adressbereich des anggegebenen Netzes. Die 18. gibt an, dass es sich bei dein letzten 18. Bits um den Hostanteil und bei allen vorherigen Bits um den Netzanteil einer IP-Adresse im spezifizierten Netz handelt. Die Subnetzmaske berechnet man, indem man für jedes Bit des Netzanteils eine 1 und für jedes Bit des Hostanteils eine 0 setzt. In unserem Fall erhalten wir also die Subnetzmaske 255.252.0.0 (binär: 11111111.11111100.00000000.00000000). Die Netzwerkadresse erhält man aus der Verundung

einer IP-Adresse im Netz und der Subnetzmaske (bzw. man setzt alle Bits im Hostanteil auf 0). In unserem Fall erhalten wir: 103.160.0.0 (binär: 01100111.10100000.00000000.00000000). Die Broadcastadresse erhält man aus der Veroderung einer IP-Adresse im Netz mit der negierten Subnetzmaske (bzw. man setzt alle Bits im Hostanteil auf 1): 103.163.255.255 (binär: 01100111.10100011.11111111.11111111).

Der Adressbereich von 103.161.122.83/18 reicht von 103.160.0.0 (Netzwerkadresse) bis 103.163.255.255 (Broadcastadresse). Somit überschneidet sich 103.161.193.83/18 mit diesem.

Aufgabe 3

TCP

Wenn ich Client-Instanzen meiner Implementierung mit einem Server aus dem Branch „Gagranelo“ kommunizieren lasse, reagiert der Server zwar auf die Befehle /Users und /quit, allerdings ist ein Versenden von Nachrichten nicht möglich, da Registrierung der Clientnamen nicht ordnungsgemäß funktioniert. Während der Server zur Registrierung lediglich eine Nachricht bestehend aus dem Namen erwartet, senden meine Clients eine Nachricht der Form „register <Name>“.

```
Enter Username:
Welcome! Type /User to find out who is online and ava
To reach a User type in this form: [Username]:[Messag


/User
register Otto
register Marvin
send Marvin hi
User not Found or invalid command!
Valid Syntax: [Username]:[Message you want to send]
```

Auch umgekehrt könnte keine Kommunikation stattfinden, da mein Server das Schlüsselwort „register“ zwangsläufig erwartet. Damit beide Implementierungen miteinander arbeiten können, muss man sich hier auf einen einheitlichen Standard einigen.

Dahingegen können die Clients meiner Implementierung über den Server der Musterlösung kommunizieren und umgekehrt.

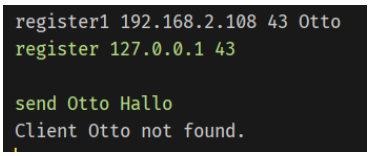
UDP

Wenn ich mein Programm (das ich bei Übung 3 abgegeben habe) mit dem aus dem Branch „masmir“ kommunizieren lasse, scheitert es an der gegenseitigen Registrierung. Diese geschieht bei beiden über den Befehl register <IP-Adresse> <Port>. Allerdings sind die Nachrichten, welche die Programme an die jeweils andere Instanz schicken verschieden. Während das fremde Programm eine Registrierungsanfrage der Form *„Hallo, hier ist …, meine IP-Adresse ist die 192.168.2.108 und du kannst mich unter Port-Nummer … erreichen.“* erwartet, erwartet mein Programm eine Anfrage der Form *„register1* [oder bei Gegenregistrierung *register2*] *IP-Adresse Name*“

Konsole meines Programms:

```
register 127.0.0.1 57
Hallo, hier ist Max, meine IP-Adresse ist die 192.168.2.108 und du kannst mich unter Port-Nummer 57 erreichen.
send Max Hi!
Exception in thread "main" java.lang.NullPointerException Create breakpoint : Cannot read field "ip" because "ep" is null
    at udp.chat_udp.sendMessage(chat_udp.java:141)
    at udp.chat_udp.connectAndTalk(chat_udp.java:91)
    at udp.chat_udp.main(chat_udp.java:34)
```

Konsole des anderen Programms:



Selbiges Problem besteht, wenn mein Programm mit der Musterlösung kommunizieren soll. Damit die verschiedenen Implementationen miteinander kommunizieren können, muss man sich auf eine einheitliche Form der Registrierungsanfrage festlegen.

Aufgabe 4

Siehe Programme im Package ueb4