Universität Trier

Fachbereich IV / Informatik

Prof. Dr. Peter Sturm
D-54286 Trier
Telefon 0651/201-3313
Fax 0651/201-3842
Email sturm@uni-trier.de

Philipp Müller
Email s4pimuel@uni-trier.de

Rechnernetze, Übungsblatt 4, Sommer 2025

Abgabe

Die Abgabe in Form eines Pull-Requests in ihren Branch des Repositories https://github.com/syssoft-ds/UDP-TCP-Chat-2025S muss bis zum 11.06.2025 um 23:59 Uhr geschehen.

Aufgabe 1: Protokoll-Header

In der Vorlesung wurden die Kategorien der IPv4-Header vorgestellt. Finden Sie in Wireshark ein beliebiges IPv4-Paket. Ordnen Sie Elemente des gefundenen Paketes den Kategorien des Headers zu (Version, Header Length, Type of Service, etc.).

Finden Sie anschließend jeweils ein UDP- und TCP-Paket und ordnen Sie auch dort die Elemente des Pakets den Kategorien zu. Beachten Sie, dass die Header dieser beiden Protokolle anders aufgebaut sind als der IPv4-Header.

Aufgabe 2: CIDR

Beschreiben Sie 103.161.122.83/18. Was ist bedeuten jeweils die 103.161.122.83 und die 18? Wie kann man daraus Subnetzmaske, Broadcastadresse und Netzwerkadresse ermitteln? Liegt die 103.161.122.83/18 im selben Netz wie 103.161.193.83/18? Notieren Sie Ihre Ergebnisse.

Aufgabe 3: Kommunikation zwischen Implementationen

Nutzen Sie das Chat-Programm der letzten Übung. Sie können dabei als Basis Ihre eigene Implementierung, eine Implementierung Ihrer KommilitonInnen, oder die Musterlösung im Hauptbranch des Repos verwenden.

Versuchen Sie, das UDP-Chat-Programm mit Implementierungen von KommilitonInnen kommunizieren zu lassen. Probieren Sie es anschließend auch mit dem TCP-Chat-Programm aus. Dokumentieren Sie Probleme, die dabei auftreten, und überlegen Sie sich Lösungen dafür.

Aufgabe 4: Programmierung

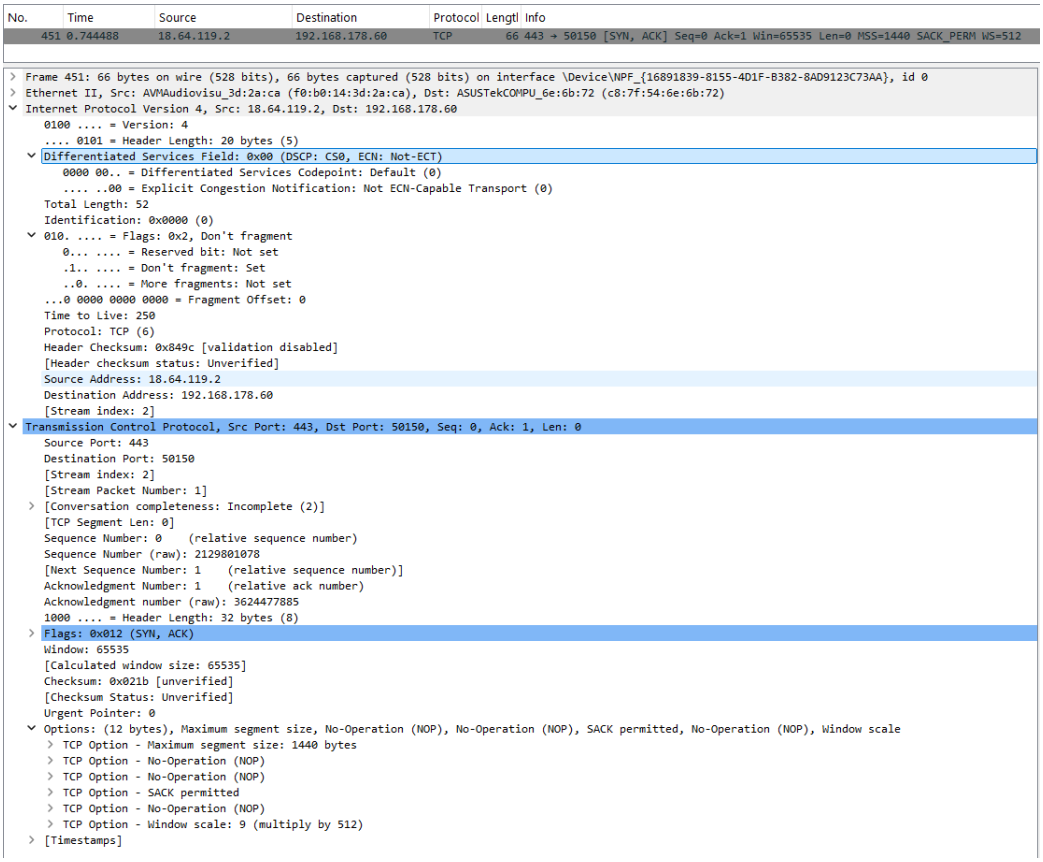
Erweitern Sie das Chat-Programm um folgende Methoden:

a) Sende Nachricht an alle bekannten Clients (für UDP-Implementierung & TCP-Server)
b) Anfrage, Nachricht an alle bekannten Client zu schicken (für TCP-Client)
c) Sendung der Liste der bekannten Clients (für TCP-Server)
d) Anfrage, Liste der bekannten Clients zu schicken (für TCP-Client)
e) Wenn bestimmte, vordefinierte Fragen gestellt wird, soll automatisch eine bestimmte, vordefinierte Antwort an den Fragesteller gesendet werden. z.B. „Was ist deine MAC-Adresse?“, „Sind Kartoffeln eine richtige Mahlzeit?“, etc. (UDP- Implementierung & TCP-Client)

Wie können Sie sicherstellen, dass diese Funktionen auch bei der Kommunikation mit Clients/Servern von anderen Implementierungen funktionieren? Nutzen Sie dazu dieses Google Docs-Dokument. Einigen Sie sich auf bestimmte Spezifikationen, so dass Ihre Implementierungen mit denen Ihrer KommilitonInnen funktionieren werden.

zu Aufgabe 1

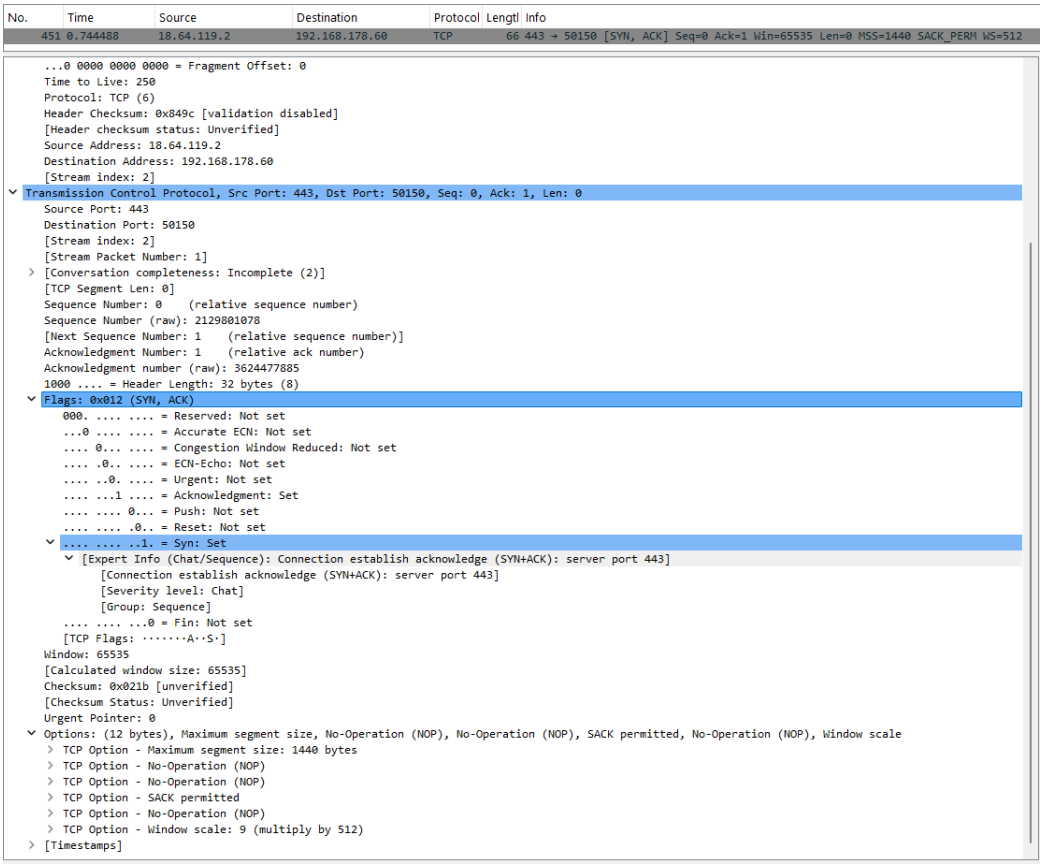
IPv4 Header



Version	0100 (Version: 4)
Header Length	0101 (20 bytes)
Type of Service	0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length	52
Identification	0x0000 (0)
ID Flags	010. .... (0x2, Don't Fragment)
	0... .... =Reserved bit: Not set
	.1.. .... =Don't fragment: set
	..0. .... =More fragments: Not set
Fragment Offset	...0 0000 0000 0000 = Fragment Offset: 0

Time to Live	250
Protocol	TCP (6)
Header Checksum	0x849 [validation disabled]
Source Address	18.64.119.2
Destination Address	192.168.178.60

TCP Header

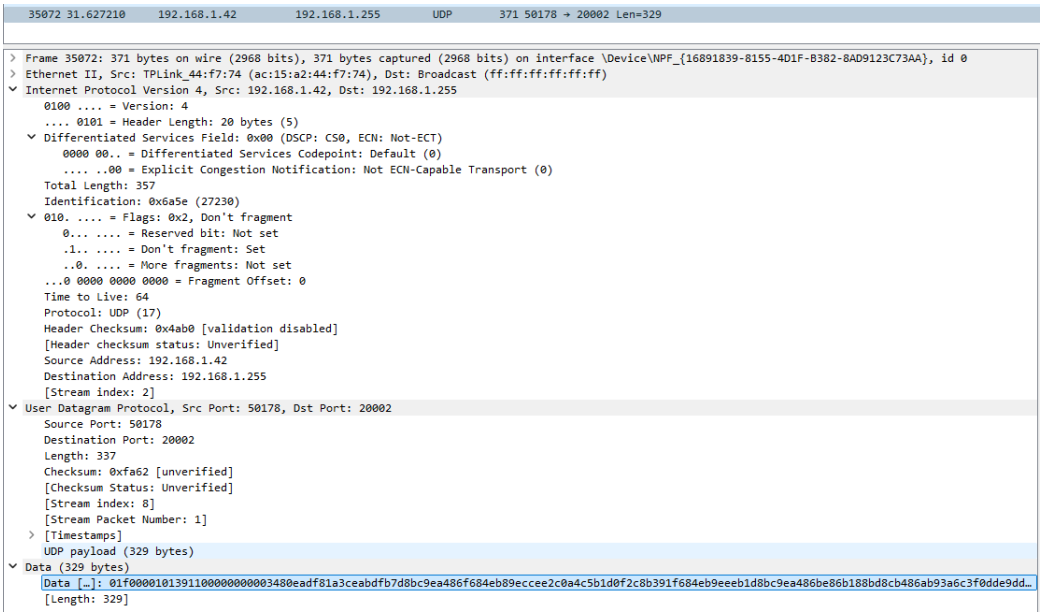


Source Port	443
Destination Port	65535
Sequence Number	0 (relative sequence number)
Acknowledgement Number	1 (relative ACK number)
Header length	32 bytes (8) Data Offset

Reserved	000. .... Not set
Cingestion Window Reduced	.... 0... .... Not set
ECN-Echo	.... .0.. .... Not set
Urgent	.... ..0. .... Not set

Acknowledgement	....1 .... Set
Push	.... 0... Not set
Reset	.... .0.. Not set
Syn	....1. Not set
Fin	....0 Set
Window	65535
Checksum	0x021b [unverified]
Urgent Pointer	0
Options	(12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted, No-Operation (NOP), Window scale

UDP Header



Source Port	50178
Destination Port	20002
Length	337
Checksum	0xfa62 [unverified]

zu Aufgabe 2

103.161.122.83/18 ist classless inter-domain routing Notation, um die IPv4 Adresse 103.161.122.83 mit ihrer zugehörigen Netzmaske darzustellen. Dabei stellen die ersten 18 Bits der IPv4 Adresse die Netzadresse und die letzten 14 Bits die Host-Adresse dar.
Die Netzadresse erhält man, indem man alle Bits der Host-Adresse auf 0 setzt. Die Netadresse für das Beispiel ist 103.161.64.0, d.i. in binär

01100111.10100001.01000000.00000000

Entsprechend umfasst der Hostbereich für das Beispiel die Host-Adressen 103.161.64.1 bis 103.161.127.254, d.i. in binär

01100111.10100001.01**000000.00000001**
bis
01100111.10100001.01**111111.11111110**

Die höchste Host-Adresse im Netzwerk ist die Broadcastadresse. Diese erhält man, indem man alle Bits der Host-Adresse auf 1 setzt. Für das Beispiel ist das die Adresse 103.161.127.255, d.i. in binär

01100111.10100001.01**111111.11111111**

Die Subnetzmaske erhält man, indem man alle Bits der Netzwerkadresse auf 1 setzt. Für das Beispiel ist die Subnetzmaske somit 255.255.192.0, d.i. in binär

11111111.11111111.11000000.00000000

Die beiden Adressen 103.161.122.83/18 und 103.161.193.83/18 liegen nicht im selben Netzwerk, da 103.161.122.83/18 im Netzwerk 103.161.64.0 und 103.161.193.83/18 im Netzwerk 103.161.192.0, d.i. in binär

01100111.10100001.01000000.00000000 (103.161.64.0)
im Unterschied zu
01100111.10100001.11000000.00000000 (103.161.192.0)

zu Aufgabe 3

UDP- Probleme

- Es kam nur eine einseitige Verbindung zustande, da nur ein client die Registrierung des anderen parsen konnte
- Es kam keine Verbindung zustande, da keiner der clients die Registrierung des anderen parsen konnte
- Es konnten von einem client keine gesendeten Nachrichten empfangen werden, da die send Nachrichten nicht geparst werden konnten
- Es traten Fragmente wie *send username* in übermittelten Nachrichten auf, da die Nachrichten vom Empfänger fehlerhaft geparst wurden

TCP-Probleme

- Es kam keine Verbindung zum Server zustande, da der Server die Nachrichten nicht parsen konnte
- Es kam zu einem Deadlock, da Client und Server nach dem Verbindungsaufbau in einen blockierenden read übergegangen sind
- Einzelne Befehle funktionierten nicht, da diese vom Server / einem anderen Client nicht geparst werden konnten
- Es traten Fragmente wie *send username* in übermittelten Nachrichten auf, da die Nachrichten vom Empfänger fehlerhaft geparst wurden

Lösungsansätze

- UDP & TCP: eine einheitliche Syntax für den Aufbau der Nachrichten festlegen
- TCP: Die Kommunikation zwischen Server und Client vereinheitlichen, um zu verhindern das client und server beide in einem *blockierenden read* stecken bleiben
- UDP & TCP: die Anzahl der Annahmen, die über die Implementierung von client und server gemacht werden, reduzieren