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 3, 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 28.05.2025 um 23:59 Uhr geschehen.

Aufgabe 1: Wireshark

Machen Sie sich mit der Implementation der UDP- und TCP-Programme vertraut, die in der ersten Vorlesung vorgestellt wurden (die Klassen nc_udp und nc_tcp, auch zu finden im Repository https://github.com/syssoft-ds/netcat). Nutzen Sie Generische KI, um sich Codeabschnitte, die Ihnen unklar sind, erklären zu lassen. Implementieren Sie die beiden Programme und probieren Sie sie aus. Überlegen Sie sich, wie Sie damit Nachrichten schicken und empfangen können.

Nutzen Sie die Capture-Funktion von Wireshark, um Ihren Traffic aufzuzeichnen, während die Programme laufen (auch im Rahmen der nächsten beiden Übungen). Vergleichen Sie die Aufzeichnung des TCP-Programms, mit der Aufzeichnung des UDP-Programms.

Erläutern Sie die Unterschiede und Gemeinsamkeiten der Funktionsweisen der beiden Programme. Nehmen Sie dabei Bezug auf konkrete Pakete in Ihren beiden PCAP-Dateien.

Aufgabe 2: UDP

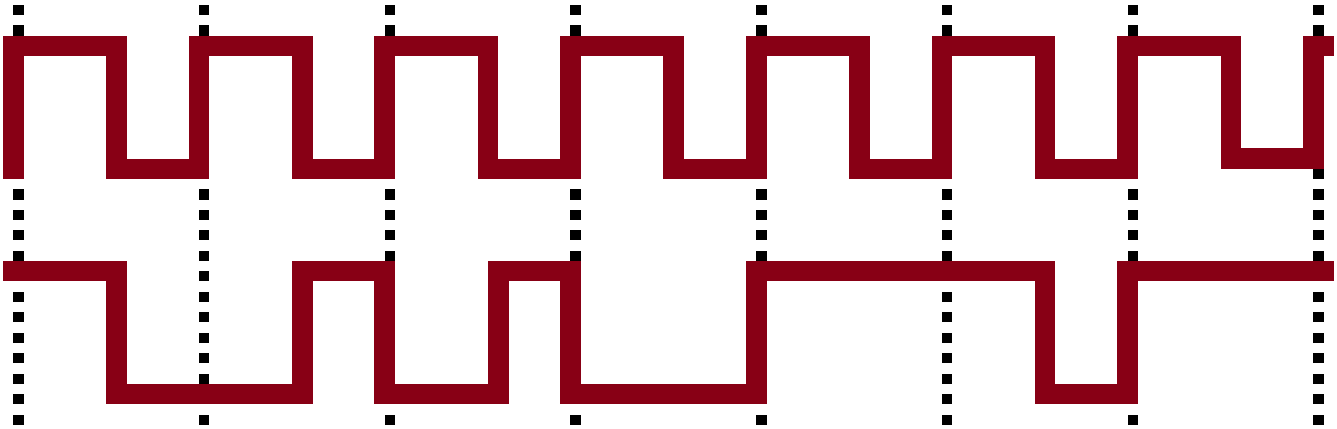
Nehmen sie nc_udp als Ausgangspunkt, und bauen Sie es zu einem Chatprogramm um. Jede Instanz des Programms soll einen Namen haben und sich bei einer anderen Instanz registrieren können (also so etwas wie „Hallo, hier ist Marvin, meine IP-Adresse ist die 192.168.0.42 und du kannst mich unter Port-Nummer 31337 erreichen.“). Anschließend sollen die Instanzen, die sich kennen, über einen Befehl „send name message“ sich gegenseitig Nachrichten senden können (name für den Ansprechpartner, message für die Nachricht).

Aufgabe 3: TCP

Verändern Sie nc_tcp ebenfalls zu einem Chatprogramm. Allerdings soll die Registrierung der Instanzen hier über den Server ablaufen. Nach der Registrierung soll es den Instanzen jedoch auch hier möglich sein, sich gegenseitig mit „send name message“ Nachrichten zu senden. Beachten Sie hier, dass bei TCP über die gesamte Dauer des Sendens und Empfangens eine Verbindung bestehen muss.

Aufgabe 4: Manchester-Code

Machen Sie sich mit der Manchester-Codierung vertraut. Zeichnen Sie den daraus resultierenden Manchester-Code. Welche der beiden Definitionen des Codes Sie verwenden, ist Ihnen überlasse.



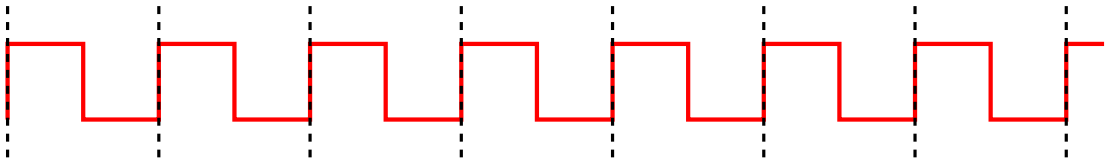
Oben ist die Clock, unten die zu codierende Bitfolge.

Aufgabe 1

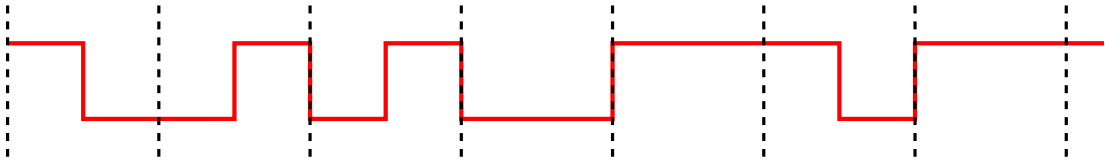
Beide Programme können als entweder Serverinstanz oder als Clientinstanz gestartet werden. Wobei die Clientinstanz jeweils eine Verbindung zu einer Serverinstanz aufbaut, indem beim Programmstart IP Adresse und Port des Servers übergeben werden. Anschließend schickt die Clientinstanz Klartext Nachrichten an die Serverinstanz und Beendet die Verbindung durch Senden des Wortes *stop*. In beiden Programmen kann die Serverinstanz lediglich Nachrichten empfangen und die Clientinstanz lediglich Nachrichten senden.
Bei der TCP Version von netcat wird für die Kommunikation zwischen dem Client und dem Server eine Verbindung aufgebaut (vgl. [SYN, ACK] und [ACK] Pakete am Beginn des Paket captures). Die Verbindung besteht während der gesamten Kommunikation (vgl. [ACK] Pakete im packet capture zwischen den [PSH] Paketen) und wird erst nach dem Senden von "stop" beendet (vgl.[FIN, ACK] Paket im packet capture).
Dagegen ist die UDP Version von netcat verbindungslos. Das heißt, es werden zwischen client und server lediglich die Nachrichten als UDP Pakete übertragen.
In der TCP Version von netcat wird für die Serverinstanz ein Serversocket verwendet und für jede Verbindung zu einer Clientinstanz ein neuer Thread gestartet, an den die Verbindung übergeben wird. Beendet ein Client die Verbindung durch Senden von "stop", wird stets nur der Socket zu diesem Client geschlossen und der entsprechende Thread beendet, während der main Thread und somit der Server weiterhin über den Serversocket erreichbar bleibt.
Dagegen verwendet die UDP Version sowohl für die Server als auch für die Clientinstanz einen DatagramSocket, der eine Peer to Peer Verbindung zwischen Client und Server aufbaut. Schließt hier die Clientinstanz durch Senden von "stop" die Verbindung wird auch der DatagramSocket der Serverinstanz geschlossen und der der Server ist nicht mehr erreichbar.
Die TCP Version hat den Vorteil, dass sie eine Verbindung zum Server aufbaut und somit sicherstellen kann, dass alle Pakete der Clientinstanz beim Server ankommen, indem sie ggf. verlorene Pakete erneut sendet. Da UDP verbindungslos ist, kann UDP nicht garantieren, dass alle Pakete beim Empfänger ankommen.

Aufgabe 4

clock:



data:



Manchester Kodierung: clock $\oplus$ data