

AAP12: Netzwerkprogrammierung mit python

Fabio Plunser
Betreuer : Walter Mueller

May 29, 2020

Inhaltsverzeichnis

1	Verwendet das netcat Programm	1
2	Programmiert einen Klientenprogrammcode, der abwechselnd eine Zeile vom Benutzer einliest, diese an den Server (ncat mit Euren Antworten) sendet und dann das Ergebnis vom Server abholt und wieder auf dem Bildschirm ausgibt.	1
3	Programmiert einen einfachen Server, der eine Verbindung entgegen nimmt und dann die erhaltenen Daten eures bereits geschriebenen Netzwerkklienten jeweils wieder zurück sendet.	2
4	Baut den Code mit Hilfe von Threads in einen multithreaded Server um:	3
5	Baut den Code mit Hilfe von Threads in einen multithreaded Server um:	3
6	Lasst den Server mit mehreren Klienten laufen und dokumentiert das wiederum mit einem großen personalisierten screenshot des Bildschirms.	4
7	Dokumentiert die aktiven Verbindungen und den listening-Socket mit dem netstat-Kommando des Betriebssystems.	4
8	Was ist unklar geblieben, wo braucht Ihr Hilfe?	4

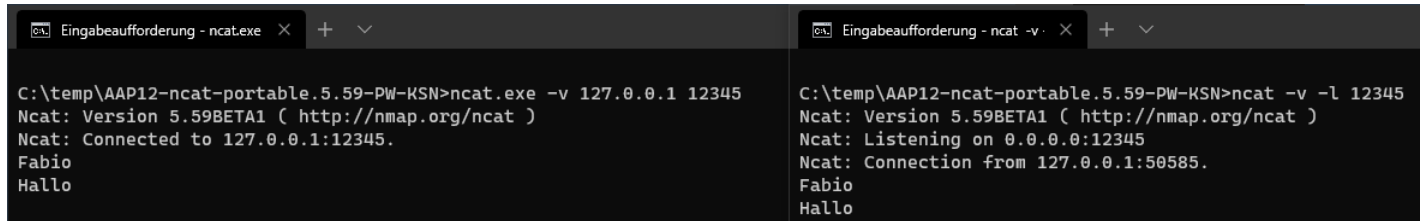
Abbildungsverzeichnis

1	CMD-ncat	1
2	Client-Programm	1
3	Client-Programm-Ausgabe	1
4	Server-Programm und Ausgabe	2
5	Multithreaded-Server-Programm	3
6	Viele Clients auf einem Server	4
7	Netstat	4

1 Verwendet das netcat Programm

1. Verwendet das netcat Programm um auf eurem PC eine Client-Server Kommunikation zu testen. Mit dem ZIP-Passwort KSN entpacken, im Windows-Defender als ungefährlich markieren und das Löschen rückgängig machen. ncat ist in zwei cmd-Fenstern einmal als Client und einmal als Server zu starten. Wie kann das Programm als Server, wie als Klient verwendet werden?

(dokumentiert eine personalisierte Kommunikation zw. Klient und Server mit screenshots)

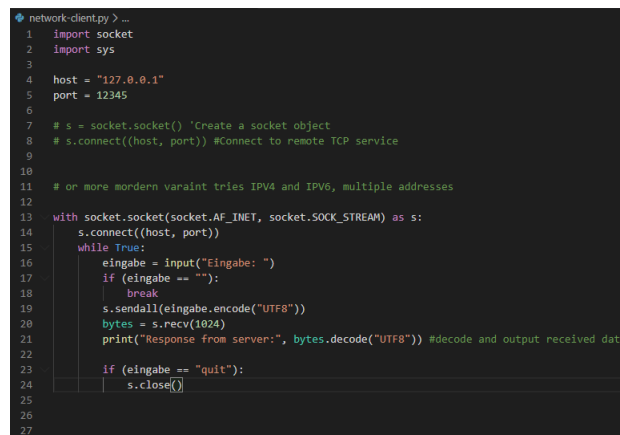


```
C:\temp\AAP12-ncat-portable.5.59-PW-KSN>ncat.exe -v 127.0.0.1 12345
Ncat: Version 5.59BETA1 ( http://nmap.org/ncat )
Ncat: Connected to 127.0.0.1:12345.
Fabio
Hallo
```

```
C:\temp\AAP12-ncat-portable.5.59-PW-KSN>ncat -v -l 12345
Ncat: Version 5.59BETA1 ( http://nmap.org/ncat )
Ncat: Listening on 0.0.0.0:12345
Ncat: Connection from 127.0.0.1:50585.
Fabio
Hallo
```

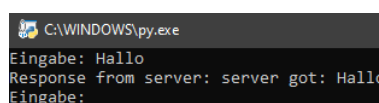
Figure 1: CMD-ncat

2 Programmiert einen Klientenprogrammcode, der abwechselnd eine Zeile vom Benutzer einliest, diese an den Server (ncat mit Euren Antworten) sendet und dann das Ergebnis vom Server abholt und wieder auf dem Bildschirm ausgibt.



```
1 import socket
2 import sys
3
4 host = "127.0.0.1"
5 port = 12345
6
7 # s = socket.socket() #Create a socket object
8 # s.connect((host, port)) #Connect to remote TCP service
9
10
11 # or more modern variant tries IPV4 and IPV6, multiple addresses
12
13 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
14     s.connect((host, port))
15     while True:
16         eingabe = input("Eingabe: ")
17         if (eingabe == ""):
18             break
19         s.sendall(eingabe.encode("UTF8"))
20         bytes = s.recv(1024)
21         print("Response from server:", bytes.decode("UTF8")) #decode and output received data
22
23         if (eingabe == "quit"):
24             s.close()
25
26
27
```

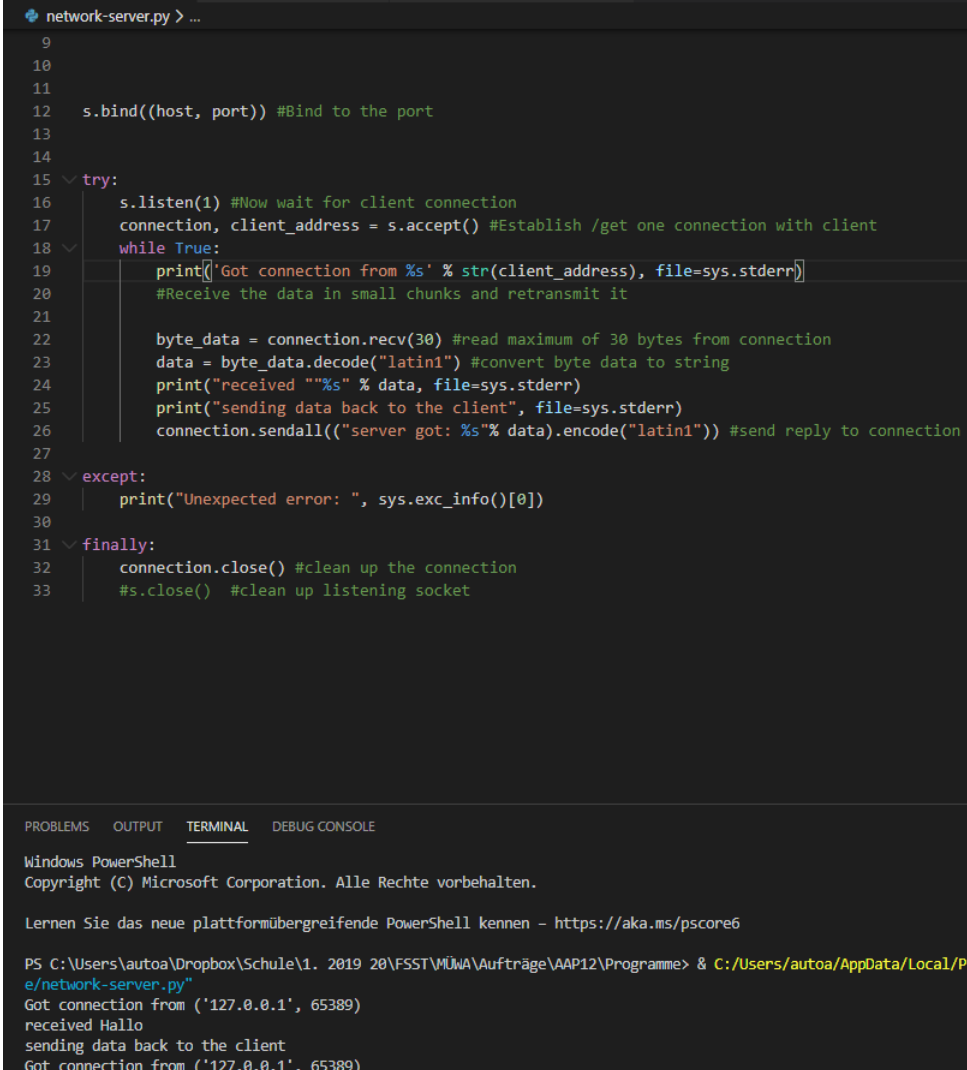
Figure 2: Client-Programm



```
C:\WINDOWS\py.exe
Eingabe: Hallo
Response from server: server got: Hallo
Eingabe:
```

Figure 3: Client-Programm-Ausgabe

3 Programmiert einen einfachen Server, der eine Verbindung entgegen nimmt und dann die erhaltenen Daten eures bereits geschriebenen Netzwerkklienten jeweils wieder zurück sendet.



```
network-server.py > ...
9
10
11
12 s.bind((host, port)) #bind to the port
13
14
15 try:
16     s.listen(1) #Now wait for client connection
17     connection, client_address = s.accept() #Establish /get one connection with client
18     while True:
19         print('Got connection from %s' % str(client_address), file=sys.stderr)
20         #Receive the data in small chunks and retransmit it
21
22         byte_data = connection.recv(30) #read maximum of 30 bytes from connection
23         data = byte_data.decode("latin1") #convert byte data to string
24         print("received \"%s\" % data, file=sys.stderr)
25         print("sending data back to the client", file=sys.stderr)
26         connection.sendall(("server got: %s" % data).encode("latin1")) #send reply to connection
27
28 except:
29     print("Unexpected error: ", sys.exc_info()[0])
30
31 finally:
32     connection.close() #clean up the connection
33     #s.close() #clean up listening socket
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

Lernen Sie das neue plattformübergreifende PowerShell kennen - <https://aka.ms/pscore6>

PS C:\Users\autoa\Dropbox\Schule\1. 2019 20\FSST\MÜNA\Aufträge\AAP12\Programme> & C:/Users/autoa/AppData/Local/Programs/Python/Python38-64/Python.exe network-server.py

Got connection from ('127.0.0.1', 65389)
received Hallo
sending data back to the client
Got connection from ('127.0.0.1', 65389)

Figure 4: Server-Programm und Ausgabe

4 Baut den Code mit Hilfe von Threads in einen multithreaded Server um:

Was sind Threads?

Ein Thread ist Teil eines Prozesses.

Es wird zwischen zwei Arten von Threads unterschieden: Threads im engeren Sinne, die sogenannten Kernel-Threads, laufen ab unter Steuerung durch das Betriebssystem.

Im Gegensatz dazu stehen die sogenannten User-Threads, die das Computerprogramm des Anwenders komplett selbst verwalten muss.

Was ist dazu in Python notwendig?

Thread "library"

siehe Untenstehendes Programm

Wann/Warum benötigt man einen multithreaded Server?

Sobald man mehr wie einen User erwartet benötigt man einen multithreaded Server

```
import socket #Import socket module
import threading

s = socket.socket() #Create a socket Object
host = '' #unspecified ip - all interfaces on host
port = 12345 #Reserve a port for your service
s.bind((host, port)) #Bind to the port
s.listen(5)

def new_client(client, connection):
    ip = connection[0]
    port = connection[1]
    print(f"the new connection was made form IP : {ip}, and port: {port}!")
    while True:
        msg = client.recv(1024)
        print(f"The client with Port:{port} said: {msg.decode()}")
        reply = f"You told me: {msg.decode()}"
        client.sendall(reply.encode("UTF8"))

while True:
    client, ip = s.accept()
    threading._start_new_thread(new_client, (client, ip))
```

Figure 5: Multithreaded-Server-Programm

5 Baut den Code mit Hilfe von Threads in einen multithreaded Server um:

Ich habe für so ein einfaches Programm einfach Programmiert dass immer ein Thread hinzugefügt wird. Mir ist bewusst, dass noch eine Funktion fehlt, die nach Schließung eines Clients auch wieder diesen Thread schließt und den Port wieder freigibt.

6 Lasst den Server mit mehreren Klienten laufen und dokumentiert das wiederum mit einem großen personalisierten screenshot des Bildschirms.

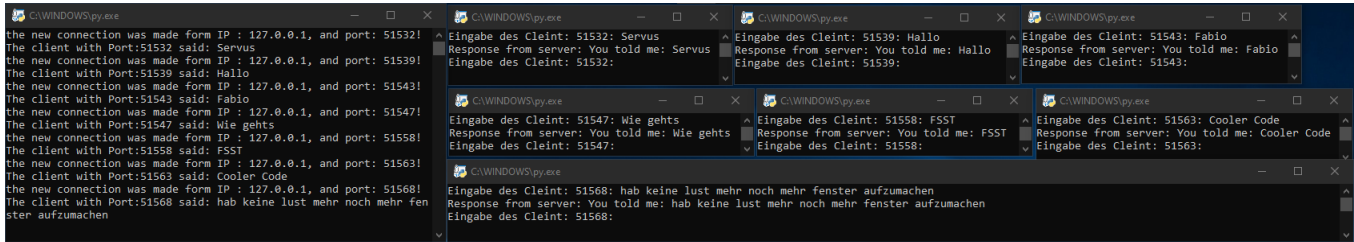


Figure 6: Viele Clients auf einem Server

7 Dokumentiert die aktiven Verbindungen und den listening-Socket mit dem netstat-Kommando des Betriebssystems.

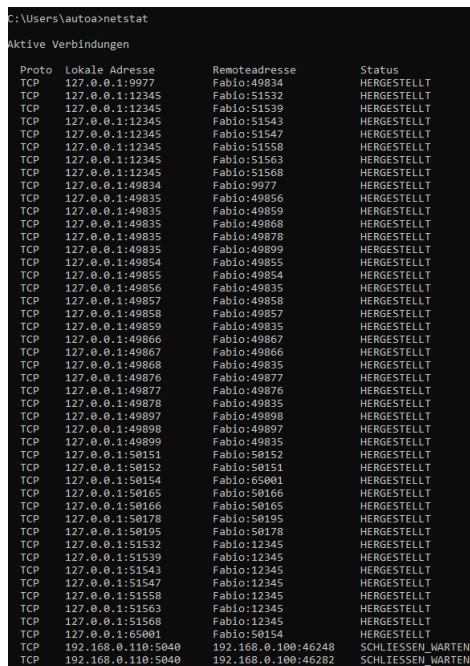


Figure 7: Netstat

8 Was ist unklar geblieben, wo braucht Ihr Hilfe?

Reicht es einfach eine Firewall Einstellung zu machen um innerhalb des Netzwerkes über Mehrere PCs miteinander zu kommunizieren? Und wenn dies dann reicht Port-Forwarding des Servers sodass alle mit meinem Code auf dem Server zugreifen können?