Beispiel für TCP Kommunikation

> Warum TCP?
>
> - HTTP baut darauf auf
> - wichtig für verbindungsorientierte Kommunikation (Wechselsprechen, wie Telefon...)
>
> Und UDP?
>
> - verbindungslose Kommunikation
> - wie Versenden von Postkarten, wartet nicht auf Rückantwort...
> - auch wichtig aber nicht jetzt

Die üblichen Betriebssysteme übernehmen heutzutage die Verarbeitung der ein- und ausgehenden Netzwerkdaten bis zur TCP-Ebene. Eine Software, die so einen TCP-Port nutzen möchte, muss eine vom Betriebssystem bereitgestellte interne Programmierschnittstelle verwenden, die sogenannten *Sockets*.

Beispiel für **Client** und **Server** in der Programmiersprache Python:

Der Server schnappt sich einen TCP Port und wartet darauf, dass ein Client ihm etwas sendet. Dazu muss der Client die IP-Adresse und die Port Nummer wissen.

Zuerst der Python Code des *Servers*:

```
import socket

PORT = 65432        # Port zum Lauschen

# wir erklären hier, was wir benutzen wollen:
#   1) Das Internet Protokoll (also IP)
#   2) TCP (das erkennt man an dem SOCK_STREAM)
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:

    s.bind(('0.0.0.0', PORT))    # das ist eine Schreibweise für 'alle
Netzwerkschnittstellen'
    s.listen()
    conn, addr = s.accept()
    with conn:
        print('Verbindung von (IP, Port)', addr)
        while True:
            data = conn.recv(1024)
            if not data:
                break
            print('empfangen: ', data.decode("utf-8") )
            conn.sendall(b'Hallo Client!')
```

Das kann jetzt ausgeführt werden: Mit Rechtsklick in neuem Fenster: Konsole

```
cd Schulung; python3 tcp_server.py
```

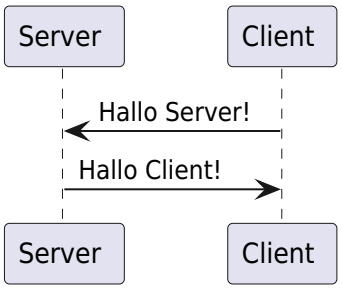
Und hier der Code des *Clients*:

```
import socket

HOST = '195.201.41.116'  # Auf diesem Rechner läuft die gewünschte Anwendung
PORT = 65432        # Dies Port Nummer muss zur Server Anwendung passen

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b'Hallo Server!')
    data = s.recv(1024)

print('empfangen: ', data.decode("utf-8") )
```

Das kann jetzt ebenfalls ausgeführt werden: Mit Rechtsklick in neuem Fenster: Konsole

```
cd Schulung; python3 tcp_client.py
```

Und das Ganze läuft dann so ab:



Merke:

> TCP ist verbindungsorientiert, d.h. nachdem der Client eine Verbindung aufgebaut hat, können Server und Client bidirektional miteinander kommunizieren. Einen kompletten Kommunikationsverlauf von Beginn durch den Client bis zum Abbau des Kommunikationskanals nennt man eine **Session**.