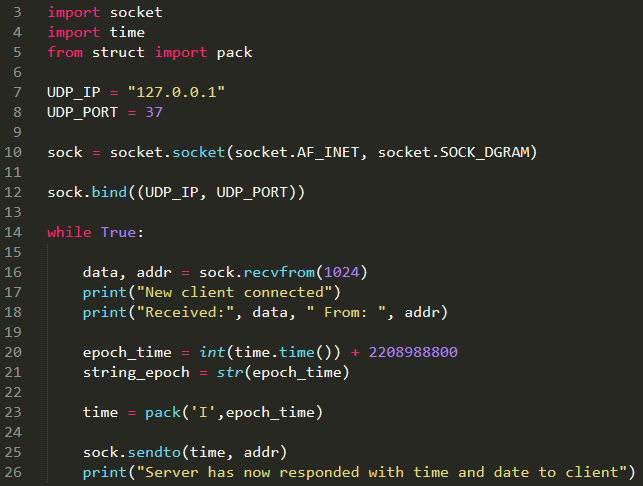
Workshop 2: Network programming

I denne workshop lød opgaven på, at vi skulle implementere en klient og en server for Time Protocol ved at bruge application program interface (API). I og med at der ikke var nogen krav om hvilket program sprog vi skulle skrive i, valgte vi at skrive i phyton. Det er et krav at serveren og klienten kan køre på en Linux maskine, og skal både være implementeret som UDP og TCP.

# UDP Server

På figur 1 nedenfor ses kodeudsnittet af vores UDP server.

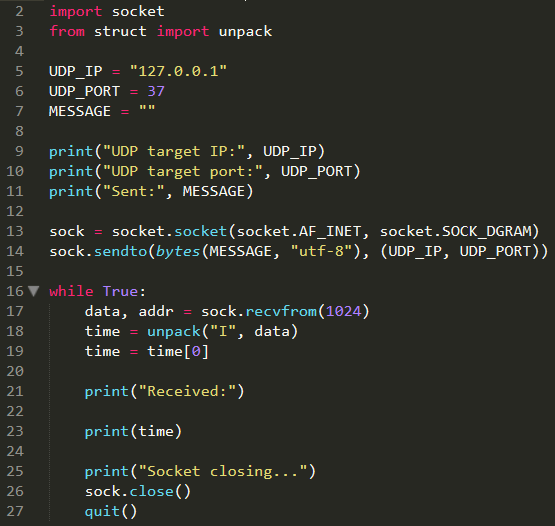


Figur 1 - Kodeudsnit af UDP server

På linje 10 opretter vi en socket ved hjælp af socket.socket() funktionen i python. På linje 12 binder vi så vores socket til vores IP addresse samt vores port.   
På linje 16 benytter vi os af funktionen sock.recvfrom(), til at modtage data fra klienten, hvor 1024 er den buffer størrelse vi besluttet os for at benytte.  
På linje 20 sammenlægger vi tiden i sekunder siden 1900, hvor 2208988800 er antallet af sekunder fra 1900-1970, og time.time() giver antallet af sekunder fra 1970 til nutid.   
På linje 23 indpakker vi så den data vi ønsker at sende til klienten.  
På linje 25 sender vi så pakken til klienten ved hjælp af funktionen sock.sendto().

# UDP Client

På figur 2 nedenfor ses kodeudsnit af UDP klienten.

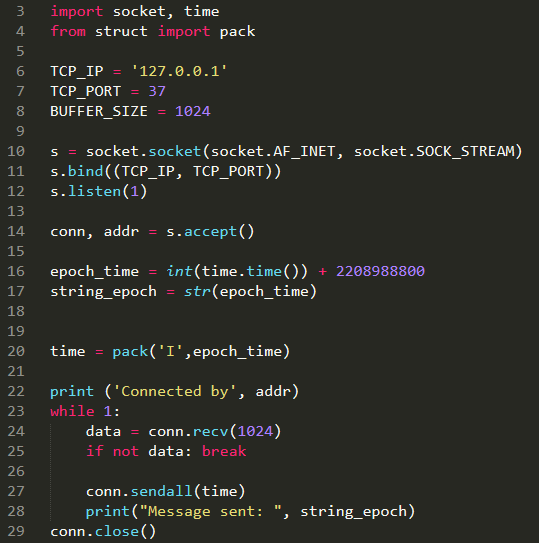


Figur 2 - kodeudsnit af UDP klienten

Vi beskriver ikke de samme funktioner som også blev benyttet i UDP serveren.   
På linje 14 sender vi ”This is a message” til UDP serveren, dette kunne lige så godt havde været en tom streng.   
Herefter modtager vi så pakken fra serveren, som vi bruger unpack() funktionen til at udpakke pakken.

# TCP Server

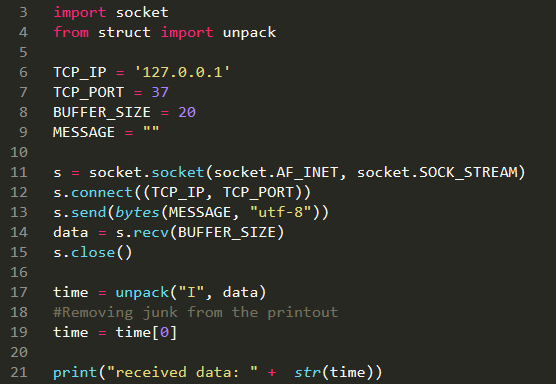
På figur 3 kan man se TCP serverens source kode. Funktionerne er mangen til UDP udgaven og er beskrevet ovenfor.  
Serveren opsættes til at lytte efter en klient. Når en klient tilsluttes bliver tiden beregnet og sendt tilbage binært.



Figur 3 - kodeudsnit af TCP serveren

# TCP Client

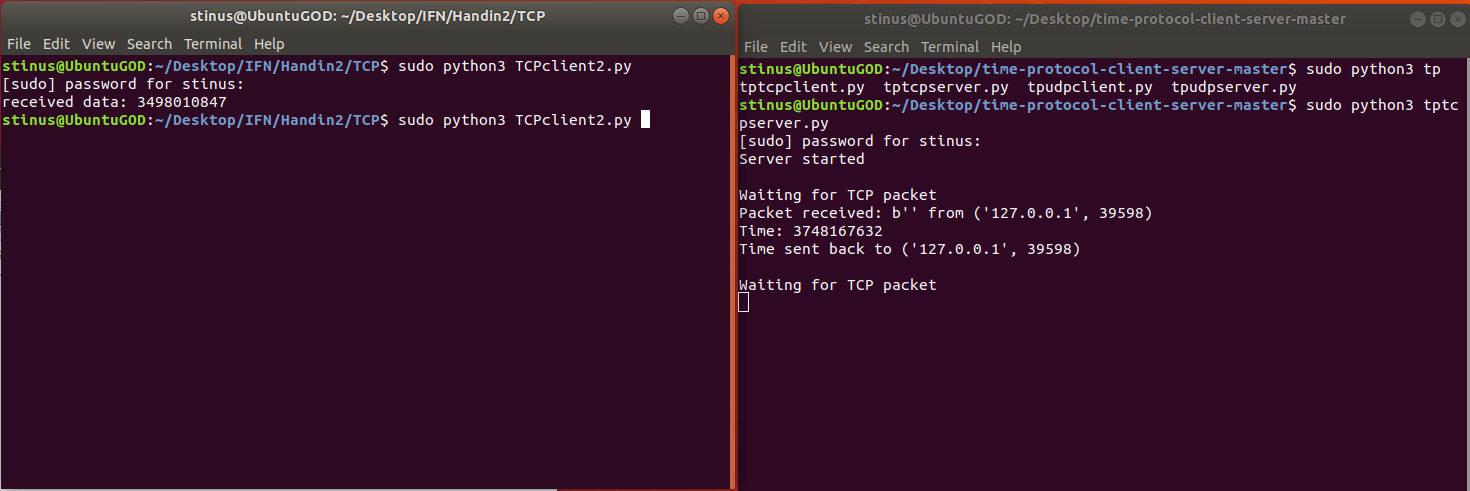
På figur 4 ses TCP klienten.  
Den er sat op til at tilslutte serverens ip-adresse, som i dette tilfælde er 127.0.0.1, hvilket også er localhost. Klienten tilslutter serveren og modtager med det samme tiden som en binær værdi. Dette formateres til en string og udskrives i terminalen.

v

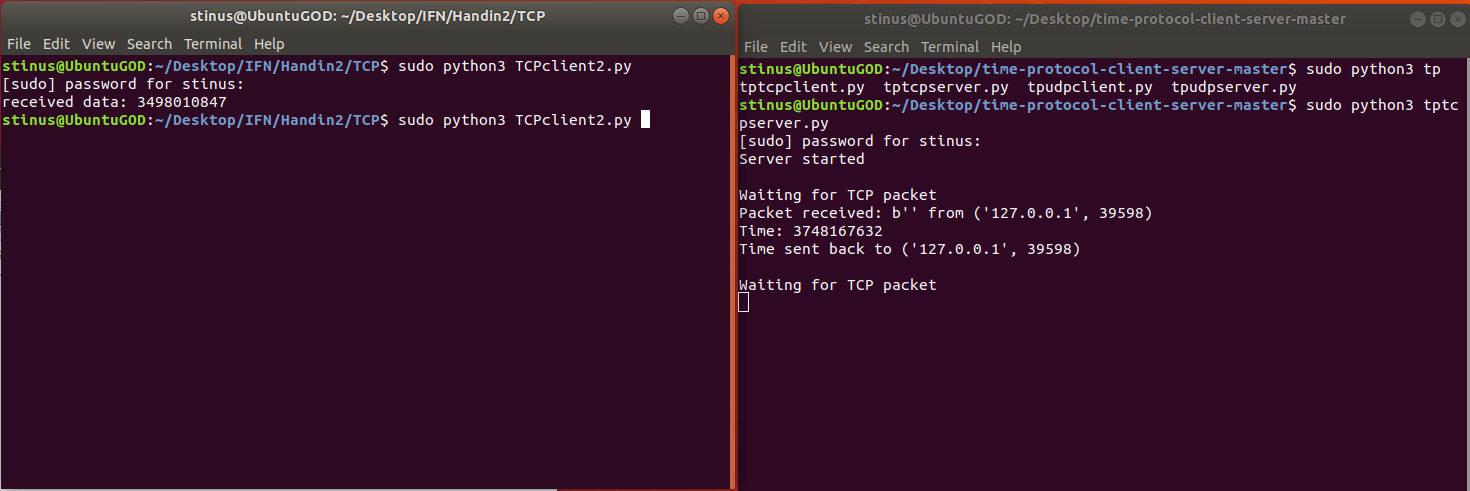
Figur 4 - kodeudsnit af TCP klienten

# Test med gruppe 6

Nedenfor ses vores test med gruppe 6. Det er skærmudklip af terminalerne hvor client til venstre og server til højre. I denne test er det vores klient og gruppe 6 server der er benyttet.



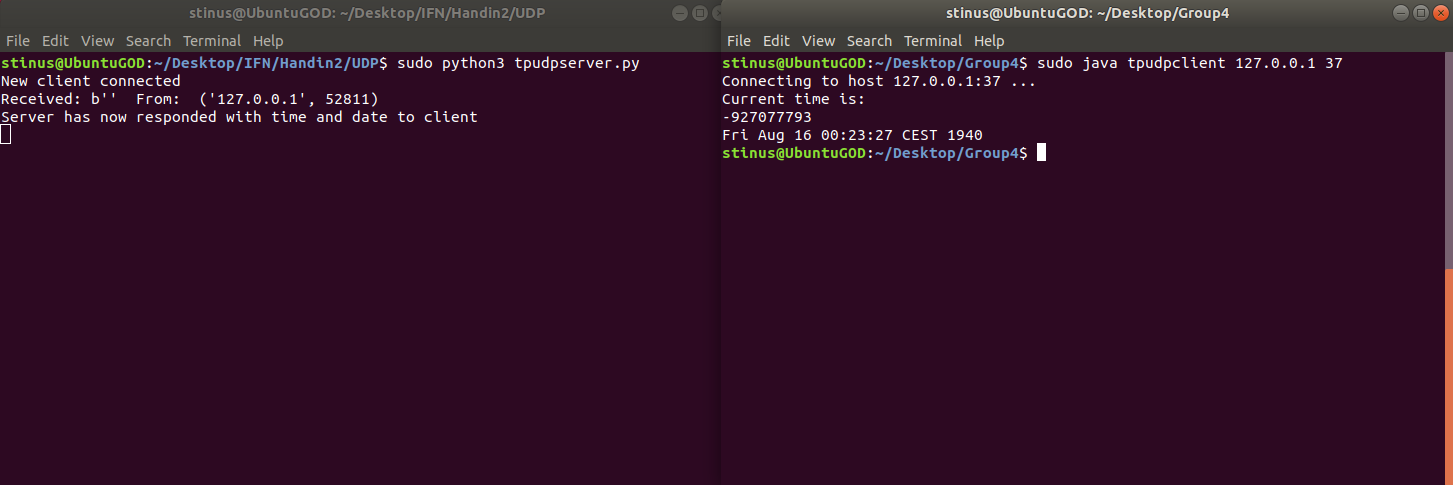
Figur 5 - TCP test med gruppe 6



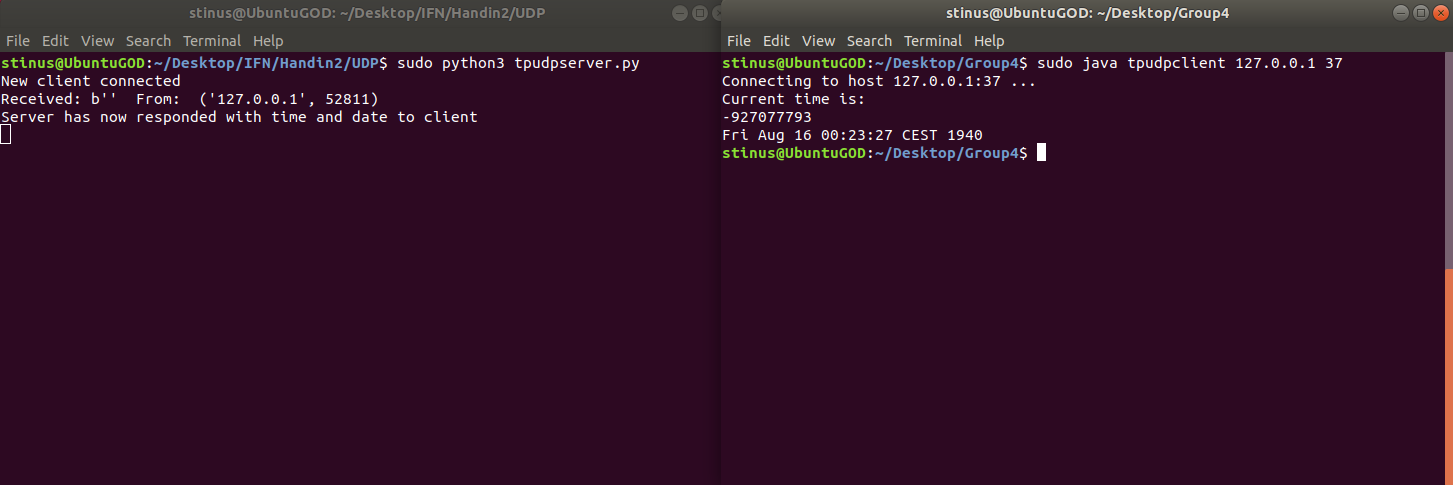
Figur 6 - UDP test med gruppe 6

# Test med gruppe 4

Nedenfor ses vores test med gruppe 4. Det er skærmudklip af terminalerne hvor client til højre og server til venstre. I denne test er det vores server og gruppe 4 klient der er benyttet.   
Som det kan ses, opstår der en fejl når vi prøver at få forbindelse over TCP, som vi ikke kan forklare.   
UDP virker som det skal.



Figur 7 - TCP test med gruppe 4



Figur 8 - UDP test med gruppe 4