**Socket programmering**

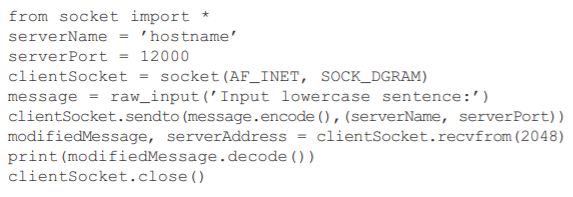
Sockets er måden applikationslagprotokoller er opbygget. Client-server modellen går ud på client processen og server processen kommunikere over netværket. Dette sker genenm sockets hvor der bliver skrevet og aflæst kommunikation. Når der programmeres en netværksapplikation skal det fungere for både clienten og serveren.

**UDP**

UDP-headeren indeholder destination IP og destinations port, samt afsenderens IP og port nummer. Source IP og port er typisk ikke gjort i selve implementationen, men i stedet af operativ systemet.

Bogen giver et eksempel i Python 3, hvor client sender et lille bogstav til serveren og serveren returnere det store bogstav. x 🡪 X

**UDP\_client:**



ServerName = skal være server IP, eller hostnavnet.

ServerPort = Skal være en fri port (som ikke er defineret som velkendt)

clientSocket initializere en socket

clientSocket.sendto = typen af beskden skal være en ”byte”. Dette gør man i python med .encode.

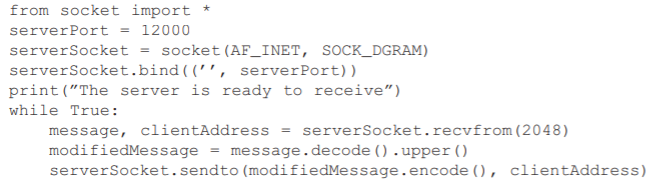
Her bliver beskeden sendt til destination: ServerName, ServerPort.

Modified message, serverAddress = gemmer beskeden i modified message og gemmer IP+port

i server address variablet. Beskeden kommer fra clientSocket

som bruger .recvfrom metoden som bruger en buffer størrelse.

**UDP\_Server**



serverSocket.bind = Assigner porten til serverSocketen.

While loopet betyder serveren venter på besked.

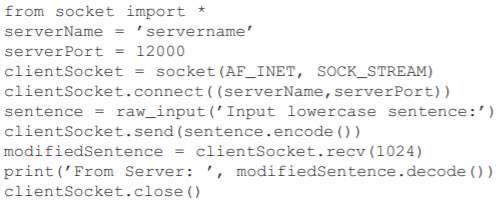
Når den for en besked bruger en message, clientAddress til at gennem clientens IP og port og besked. Herefter decoder den byte typen og sender modified message tilbage til client. (encoder den også får afsending, da det skal være en byte type der sendes).

**TCP**

TCP er anderledes end UDP. På grund af at der først skal oprettes en forbindelse med et 3-vejs handshake. Her bruger man client addressen (IP og port) og server addressen (IP og port) i oprettelsen af TCP forbindelsen. Efter forbindelsen er oprettet kan en side bare ”droppe” data igennem TCP forbindelsen (i modsætning til UDP-serveren der hele tiden skulle huske en adresse).

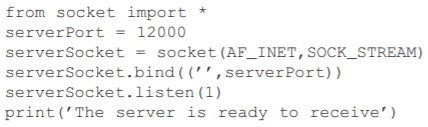
I TCP er det altid clienten som tager initiativ til at starte en forbindelse. Det kræver kun at serveren allerede har en process kørende. Serveren venter på at clienten vil oprette en forbindelse, når dette sker opretter serveren en ny forbindelse som er dedikeret til clienten. Python 3 eksempel fra bogen:

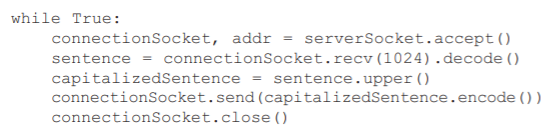
**TCP\_Client**



Der er ikke meget forskel på UDP eksemplet og TCP eksemplet her. Udover clientSocket er initializeret på en anden måde (med TCP i stedet for UDP). Der er også forskel hvordan man sender og modtager, da man ikke behøves at bruge client/server adresser.

**TCP\_Server**



¨

Der er lidt anderledes fra TCP\_client og UDP eksemplet.

Igen bliver serverSocket initializeret som en TCP forbindelse.

Server.Socket(1) – Serveren venter på TCP requests, parametren betyder max antal forbindelser.

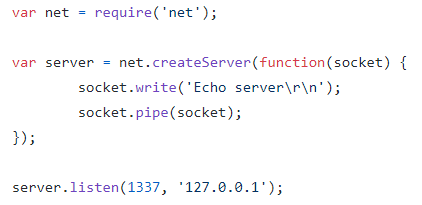
Forbindelsen bliver accepteret med serverSocket.accept() hvor serveren opretter connectionSocket, som er en TCP forbindelse. Her sendes beskeden gennem connectionsocket, uden at bruge dest. adressen.

**Socket programmering i JavaScript**

JavaScript implementering fra: https://cs.lmu.edu/~ray/notes/jsnetexamples/

Eksemplet er ligesom i Python 3 fra bogen. Client sender en besked og serveren returnere beskeden i all caps.

**Eksempel på en simpel TCP server**



Her venter server på request fra clienten. Her sender den bare ”echo server” til client.

**Eksempel på TCP client**



Her laves der en client socket og der oprettes forbindelse med port, ip og en navnløs function.

Client.on(data) og (close) modtager beskeder, i dette tilfælde printer clienten dataen fra serveren

**Links**

<https://cs.lmu.edu/~ray/notes/jsnetexamples/> - fra pensum

<https://gist.github.com/tedmiston/5935757> - eksempel fra billeder

<https://riptutorial.com/node-js/example/22406/a-simple-tcp-client> - simpelt eksempel

<https://nodejs.org/api/net.html> - docs til net

<https://www.hacksparrow.com/nodejs/tcp-socket-programming-in-node-js.html> - meget detaljeret

websocket:

<https://javascript.info/websocket>

<https://html.spec.whatwg.org/multipage/web-sockets.html#network>

<https://www.youtube.com/watch?v=YaJbc7s1ROg>