|  |
| --- |
| Door de Graaf Simon, Janseger Eli, Van Severen Seppe, Willaert Rob 6Ia  22-11-2019 |

|  |
| --- |
| BROEDERS |
| Chatprogramma |
| Maken van een cliënt |

Inhoud

[1 TCP & UDP 3](#_Toc24118703)

[1.2 TCP 3](#_Toc24118704)

[1.3 UDP 3](#_Toc24118705)

[1.4 Waarom gebruiken we TCP 3](#_Toc24118706)

[1.5 Voorbeeld TCP 3](#_Toc24118707)

[2 Poorten 4](#_Toc24118708)

[2.2 Wat zijn poorten 4](#_Toc24118709)

[2.3 Welke poorten gaan we gebruiken 4](#_Toc24118710)

[2.4 Voorbeeld firewall error 4](#_Toc24118711)

[3 Mutlithreading 5](#_Toc24118712)

[3.2 Wat is multithreading 5](#_Toc24118713)

[3.3 Wat zijn threads 5](#_Toc24118714)

[3.4 Waarvoor gebruiken we threads 5](#_Toc24118715)

[3.5 Voorbeeld multithreading 5](#_Toc24118716)

[4 Sockets 6](#_Toc24118717)

[4.2 Wat zijn sockets 6](#_Toc24118718)

[Een *socket* is een communicatie-eindpunt in een netwerk. Deze worden geïdentificeerd door de combinatie van het IP-adres en de poortnummer gescheiden door een deelteken bv: 127.0.0.1:80 6](#_Toc24118719)

[4.3 Waarom gebruiken we sockets 6](#_Toc24118720)

[We gebruiken in de server de stream socket dit is een type socket waarmee een betrouwbare bytestroom tussen *hosts* kan worden opgezet. Dit is standaard bij TCP. 6](#_Toc24118721)

[4.4 Voorbeeld van sockets 6](#_Toc24118722)

[5 Oefening Chatprogram 7](#_Toc24118723)

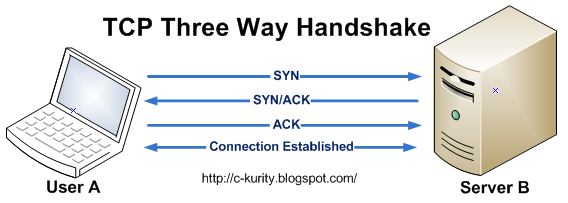
[5.1 Client: solution 7](#_Toc24118724)

[6 Bronnen: 8](#_Toc24118725)

# TCP & UDP

## TCP &

TCP of transmission control protocol is een connectie-georiënteerd protocol dat veel gebruikt wordt voor het maken van netwerkverbindingen.

TCP maakt ook gebruik van *Quality of service (QoS)* dit betekent dat pakketten voorrang op andere kunnen krijgen maar het maakt ook gebruik van de *Three-Way Handshake,* dit is om de connectie te controleren of de data die je verstuurt zeker aankomt.

UDP of *User Datagram Protocol* is sneller dan TCP maar gaat de connectie tussen *sender* en *reciever* niet controleren waardoor er data verloren kan gaan. Dit word gebruikt bij Livestreans, fps

## Waarom gebruiken we TCP

We gaan TCP gebruiken voor de zekerheid dat onze berichten zeker aankomen. Als je

## Voorbeeld TCP

# Poorten

# Wat zijn poorten

Een poort of een gate zien we vooral als een toegang tot iets. Er zijn 2 soorten poorten een *hardwarepoort* en een *netwerkpoort*.

Altijd als je ergens naar connecteert maak je eigenlijk een connectie met een eind-poort. Dit gebeurt bij sockets ook als je bijvoorbeeld *FTP* probeert te connecteren is dit via poort 21.

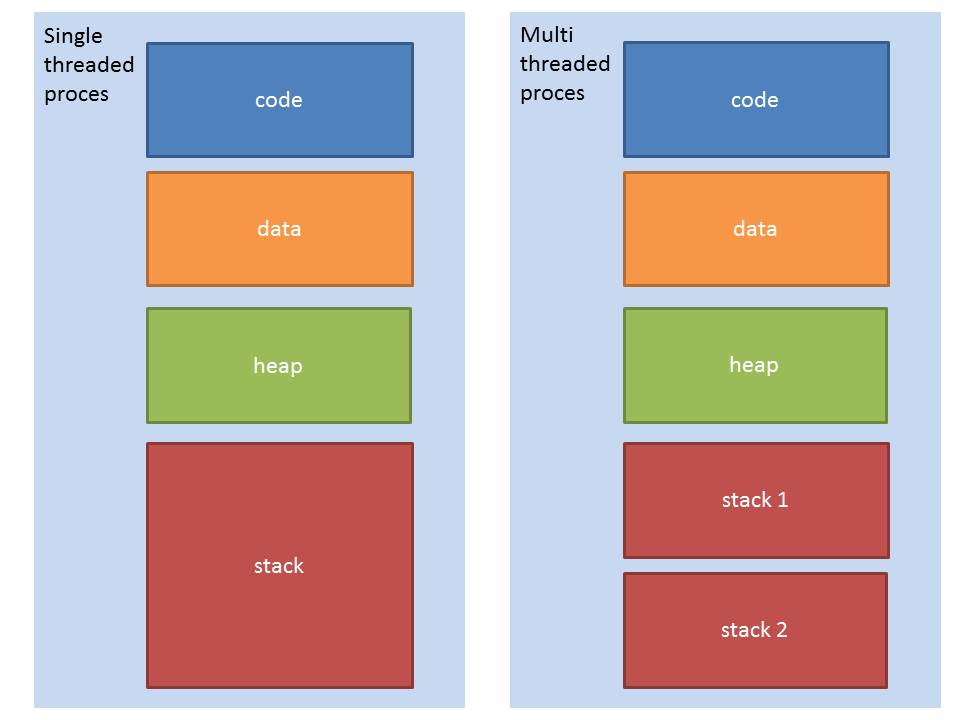
## Welke poorten gaan we gebruiken

We gaan netwerkpoorten gebruiken voor de verbinding tussen de server en de client. We gaan poort 443 gebruiken voor de connectie naar het internet.

## Voorbeeld firewall error

# 3. Multithreading

## Wat is multithreading

Bij multithreaden wordt de taak verdeeld in kleinere delen, die kleinere delen noemen we threads. Deze worden dan allemaal verstuurd en op het einde terug in hun originele vorm geplaatst. Het doel van multithreading is om meerdere taken tegelijk uit te kennen voeren bv: het luisteren naar wat de cliënts sturen en ondertussen berichten kunnen *displayen*.

## Wat zijn threads

We verdelen onze taken in *threads*, maar wat zijn *threads* juist? Threads zijn kleine onderdelen van een grotere taak. Een voorbeeld ervan gaan we duidelijk zien in de oefening als je als cliënt een bericht wilt sturen naar de server gebeurt dit met multithreading in de server. De server zal luisteren en ondertussen berichten versturen naar al de andere cliënts. We gebruiken een thread voor te kunnen luisteren en een thread voor de berichten te kunnen versturen.

## Waarvoor gebruiken we threads

Wij gaan threads gebruiken in de server voor het luisteren naar inkomende berichten van de cliënts en het doorsturen van de berichten van de cliënt naar alle andere verbonden cliënts.

## Voorbeeld multithreading

# Sockets

## Wat zijn sockets

## Afbeeldingsresultaat voor network socketEen socket ziet er zo uit: 192.168.1.1:1024 . Alles voor de dubbele punt staat voor het IP-adres van de server of cliënt. Alles na de dubbele punt staat voor de poort waarover het gaat verlopen. De Verbinding zal altijd verschillend zijn voor elke pc. Maar dit is niet zo voor elke server hierbij kunnen sommige poorten hetzelfde blijven.

## Waarom gebruiken we sockets

## De Socket die we al zeker gebruiken is een *listening* socket, deze gaan we gerbuiken om te luisteren naar sockets die binnenkomen. Het poortnummer van deze socket gaat nooit veranderen omdat het alle *requests* van de cliënts altijd via deze socket zal accepteren.

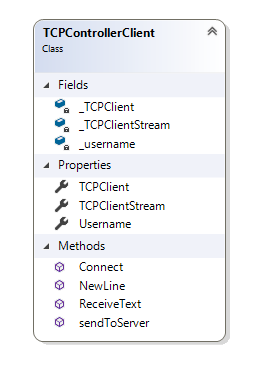
## Voorbeeld van sockets

# Oefening Chatprogram

Met deze oefening gaan we een chatprogramma maken zodat cliënts met elkaar kunnen communiceren. Wat we voor deze oefening nodig hebben zijn 1 project: Client Jullie gaan de client maken, de server hebben wij al voor jullie gemaakt en kan je vinden op onze github namelijk(moet nog gemaakt worden).

## 5.1 Client: solution

Maak een *Windows Forms Application* aan met de naam ***ChatClient.***

**Client: Domeinlaag**

* Maak 3 eigenschappen aan waarin de TCPClient, TCPCLientStream en Username kunnen bewaard worden. Declareer \_TCPClient als TcpClient en \_TCPClientStream als Networkstream.

## Bronnen:

<https://nl.wikipedia.org/wiki/TCP-_en_UDP-poorten>

<http://howtostartprogramming.com/vb-net/vb-net-tutorial-53-multithreading/>

<http://www.handel.broeders.be/wiki/index.php?title=Transport_protocol#Transmission_Control_Protocol>

<https://www.geeksforgeeks.org/tcp-and-udp-in-transport-layer/>