HoGent

BEDRIJF
EN
ORGANISATIE

Hoofdstuk 28
Netwerkprogrammeren

Leerdoelen

- Netwerkprogrammeren in Java met
  - URL's
  - Sockets
  - Datagrammen
  - Clients
  - Servers

1. Inleiding

- Fundamentele netwerkmogelijkheden:
  - package java.net
  - Stream-based communicatie: netwerking als een stream van gegevens
  - Packet-based communicatie: verzenden van individuele pakketten met informatie zoals beelden, audio en video over het Internet
- Client/server:
  - De client vraagt een actie uit te voeren
  - De server voert de actie uit
  - De server antwoordt aan de client

  → request-response model

  → vb: interactie tussen webbrowsers en webservers

3

- Met socket-based communicatie is netwerking zoals file I/O
  - Een socket is stukje software dat één eindpunt van een verbinding voorstelt
- Met stream sockets brengt een proces een connectie tot stand met een ander proces
- Zodra de connectie er is, stromen de gegevens tussen de processen in continue streams
- Stream sockets leveren een connectie-geörienteerde service
- Het gebruikte transmissieprotocol is TCP (Transmission Control Protocol)

HoGent

4

- Met datagram sockets worden individuele pakketten met informatie verzonden
- UDP—User Datagram Protocol—is een connectieloze service, en zodoende wordt niet gegarandeerd dat de pakketten in een bepaalde volgorde toekomen
  - Pakketten kunnen verloren gaan of zelfs dubbel aankomen
- UDP is het meest geschikt voor netwerktoepassingen die geen error checking en betrouwbaarheid van TCP vereisen

HoGent 5

TCP versus UDP

TCP

- Transmission control protocol
- Connectie-georiënteerd
- 3-way handshaking
- Betrouwbaar transport

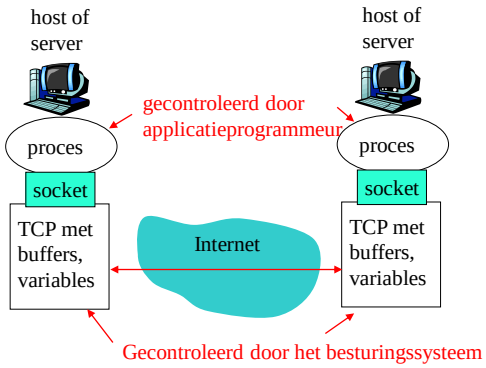
UDP

- User datagram protocol
- Connectieloos (no handshaking)
- Onbetrouwbare data tranfert
- Snelheid is belangrijk
- Realtime applicaties (skype)

HoGent 6

2. Socket-programmeren met TCP

- Schema:



HoGent

Een eenvoudige **Server** met Stream Sockets opzetten

Een eenvoudige server opzetten in 5 stappen:

Stap 1: creëer een ServerSocket object

- ServerSocket constructor
  ServerSocket server = new ServerSocket (*portNumber, queueLength);*
- De constructor legt het poortnummer vast waarop de server wacht op connecties van de client
  → binding van de server aan de poort
- Een client vraagt een connectie aan de server op deze poort
- Een geldig poortnummer ligt tussen 0 en 65565.
  De meeste besturingssystemen reserveren de poortnummers kleiner dan 1024 voor systeem services
- Een poort vragen die al in gebruik is of geen geldig nummer heeft, leidt tot een BindException

HoGent

- Programma's beheren elke connectie van een client met een Socket object.
- Sockets verbergen de complexiteit van netwerk-programmeren

***Stap 2:* de server luistert onafgebroken naar een poging van een client om een connectie te maken (blocks)**

- Het programma roept de methode accept aan om te luisteren naar een connectie van een client

```
Socket connection = server.accept();
```

  - Deze methode levert een Socket af wanneer een connectie met een client tot stand gekomen is
- Door de Socket kan de server interageren met de client

HoGent 9

Stap 3: de OutputStream- en InputStream-objecten worden opgehaald zodat de server kan communiceren met de client door het verzenden en ontvangen van bytes.

  - De server roept de methode getOutputStream aan op de Socket en krijgt een referentie naar de Socket's OutputStream. Dan wordt de methode getInputStream aangeroepen op de Socket om een referentie te krijgen naar de the Socket's InputStream

Stap 4: tijdens de verwerkingsfase communiceren de server en de client via de OutputStream- en InputStream-objecten

Stap 5: wanneer de transmissie afgehandeld is, sluit de server de connectie door de methode close aan te roepen op de streams en op de Socket

HoGent 10

Een eenvoudige **Client** met Stream Sockets opzetten

Een eenvoudige client opzetten in 4 stappen:

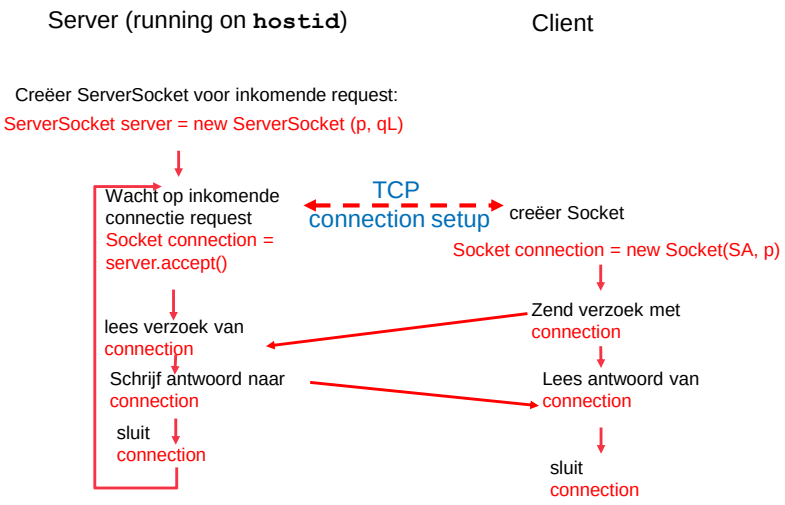
***Stap 1**:* de Socket constructor legt een connectie met de server

```
Socket connection = new Socket (serverAddress, port);
```

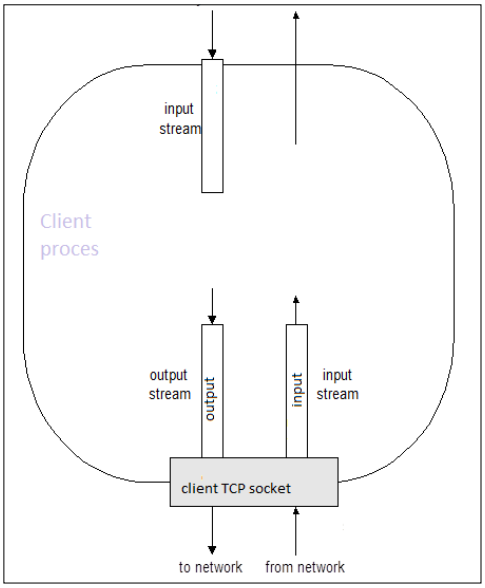
- Als de connectie tot stand gebracht is, dan wordt een Socket afgeleverd
- Als de connectie niet tot stand kan gebracht worden, dan wordt een IOException geworpen
- Een onjuiste servernaam heeft een UnknownHostException tot gevolg

- ***Stap 2:*** de client gebruikt de methoden getInputStream en getOutputStream om referenties naar InputStream and OutputStream te verkrijgen
- ***Stap 3:*** tijdens de verwerkingsfase communiceren de server en de client via de OutputStream en InputStream objecten
- ***Stap 4:*** wanneer de transmissie afgehandeld is, sluit de client de connectie door de methode close aan te roepen op de streams en op de Socket

Client/server socket interaction: TCP

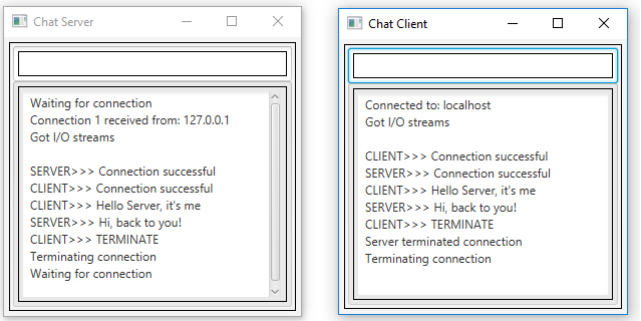


Streams bij TCP



Client/Server Interactie met Stream Socket Connecties

- Voorbeeld: een eenvoudige client-/server chat applicatie (**zie ECLIPSE projecten** OOPIII_NET_TCP_FX_server, OOPIII_NET_TCP_FX_client en OOPIII_NET_TCP_FX_common)



TCP Chat Voorbeeld

- De methode runServer zet de server klaar om een connectie te ontvangen en één connectie per keer te verwerken
- De methode getInet-Address levert een InetAddress (package java.net) af dat informatie bevat over de client
- De methode getHostName levert de hostname af van de client
- Het IP adres (127.0.0.1) en de hostname (localhost) zijn bruikbaar om netwerkapplicaties te testen op lokale computer
  - Heet het loopback adres
- De client kan uitgevoerd worden vanaf elke computer op het internet en geeft het IP adres of de hostname van de server mee als command-line argument

TCP Chat Voorbeeld

- Aanroep van de methode flush van OutputStream
  - Zorgt ervoor dat de ObjectOutputStream op de server de stream header zendt naar de ObjectInputStream van de client
  - De stream header bevat informatie van object serialization dat gebruikt wordt om objecten te verzenden
  - Deze informatie is nodig voor ObjectInputStream zodat voorbereidingen getroffen worden om die objecten correct te ontvangen
    - Creëer eerst altijd ObjectOutputStream
    - Roep dan flush aan

TCP Chat Voorbeeld

- De methode getByName levert een object af dat het IP adres bevat
  - De methode getByName heeft een String als parameter die ofwel het IP adres of de hostname bevat
- De localhost:
  - InetAddress.getByName(“127.0.0.1”)
  - InetAddress.getByName("localhost")
  - InetAddress.getLocalHost()

3. Connectieloze transmissie met datagrammen

- Connectie geörienteerde transmissie vertoont gelijkenis met het telefoonverkeer
  - Je draait een nummer en je krijgt een verbinding met de telefoon van de persoon met wie je wil communiceren
  - De verbinding blijft in stand zelfs al praat je niet
- Connectieloze transmissie met datagrammen vertoont gelijkenis met de bezorging van post via de klassieke weg (postbode)
  - Een grote boodschap die niet in één omslag past, wordt verdeeld en elk deel wordt in een genummerde briefomslag verstuurd
  - Alle omslagen worden op hetzelfde tijdstip gepost
  - De omslagen kunnen toekomen in juiste volgorde, in een andere volgorde of totaal niet toekomen (uitzonderlijk)

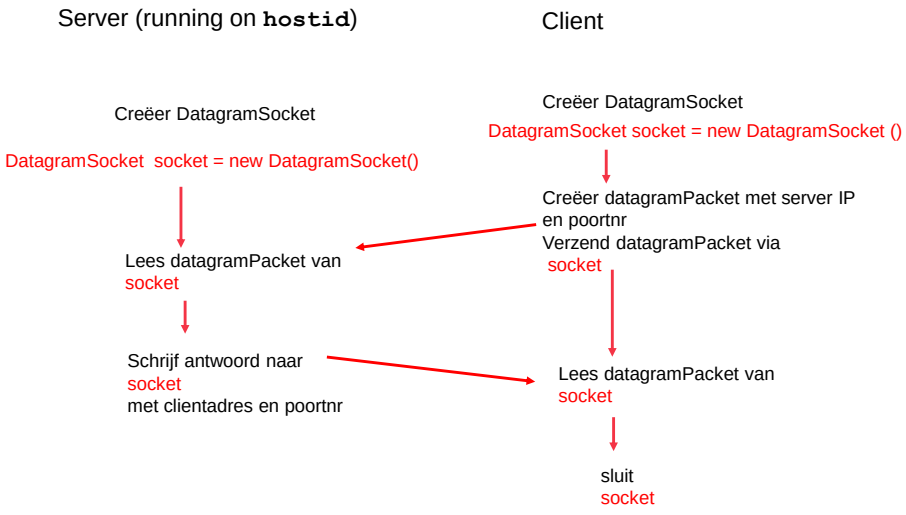
19

- De klasse Server heeft twee DatagramPakketten die de server gebruikt om informatie te verzenden en te ontvangen en één DatagramSocket die de pakketten verzendt en ontvangt
- De DatagramSocket constructor kent aan de server een poort toe waarop de server de pakketten van de client kan ontvangen
  - De client geeft het poortnummer mee in de pakketten die hij verzendt naar de server
  - Een Socket-Exception wordt gegooid wanneer de DatagramSocket constructor het DatagramSocket niet kan binden aan het poortnummer (ongeldig poortnummer, poortnummer reeds in gebruik)

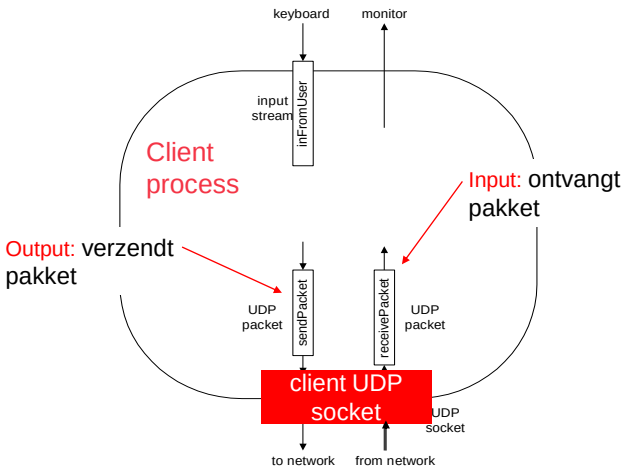
HoGent

20

Client/server socket interaction: UDP



Client bij UDP



Voorbeeld: UDP Chat

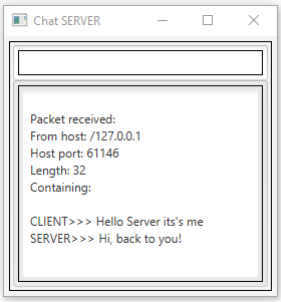
• Voorbeeld: een eenvoudige UPD versie van de client-/server chat applicatie (zie **ECLIPSE projecten** OOPIII_NET_UDP_FX) De server en de client versie zitten in dezelfde applicatie. Het verschil is door een run argument mee te geven (Unnamed parameter)

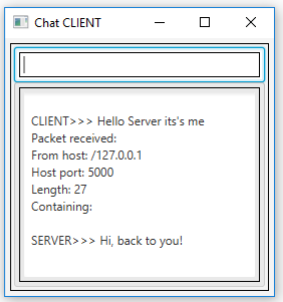
- •Als client --> arg1 = CLIENT en optioneel arg2 = hostname
- •Als server --> geen arg nodig (default status = SERVER)

•Voor server run:

- • doe build, kopieer executable jar (zit in map dist\) naar andere map (eventueel ander computer) en run de executable jar (dubbelklik).
- • Dan kan je één of meerdere clients runnen (opgelet unnamed run parameter CLIENT instellen)

Voorbeeld: UDP Chat





- De methode receive wacht op een pakket dat toekomt op de Server
  - Slaat het verzonden pakket op in het DatagramPacket
  - Werpt een IOException als een error optreedt bij het ontvangen van een pakket
- De methode getAddress levert het IP adres van de computer waar het pakket werd verzonden
- De methode getPort levert het poortnummer van de client die het pakket verzond
- De methode getLength levert het aantal bytes van de ontvangen gegevens af
- De methode getData levert een byte array met de gegevens
- De methode send gooit een IOException wanneer een error plaatsvindt bij het verzenden van een pakket

25