Inleiding Computersystemen en Netwerken

Opdracht 2: Een-op-een chatprogramma in Python

Deze opdracht mag individueel worden gemaakt of in een team van **ten hoogste twee personen**. De opdracht moet worden ingestuurd volgens de instructies aan het einde van dit document. Ingestuurde opdrachten zullen worden beoordeeld volgens de rubric die beschikbaar is op Brightspace.

Voorzie je code van commentaar.

Plagiaat en het gebruik van generatieve AI zullen niet worden getolereerd. Vermoedens van fraude en plagiaat zullen worden gemeld bij de Examencommissie en de commissie beslist over het vervolg. Het is niet toegestaan om tekst of code te genereren met ChatGPT of andere (generatieve) AI-hulpmiddelen. Je bent verplicht om de gevraagde functionaliteiten zelf te implementeren. Automatische plagiaat checks zullen worden uitgevoerd op alle ingestuurde programma's. We behouden ons het recht voor om je uit te nodigen om je ingestuurde code nader toe te lichten. Zie ook de pagina Practicum > Algemene informatie op Brightspace.

Deadline: vrijdag 12 december 23:59 uur (einde week 7)

Introductie

In dit project ga je in de programmeertaal Python een klein programma schrijven dat gebruik maakt van het netwerk. Dit programma zal een één-op-één chatprogramma zijn. Wanneer we dit programma uitvoeren op twee verschillende computers ("hosts") kunnen we eenvoudige tekstberichten uitwisselen tussen die twee computers. Naast het chatten wordt ook een functionaliteit geïmplementeerd om bestanden te versturen. Het overbrengen van het bestand zal worden gerealiseerd met een (betrouwbare) TCP-verbinding.

Er is wel één belangrijke kanttekening: je bent **verplicht** om UDP te gebruiken voor het versturen van de tekstberichten. Zoals we in de hoorcolleges hebben gezien geeft UDP geen enkele garantie of een packet wel aankomt. Het is daarom jouw taak om een methode of protocol bovenop UDP te ontwikkelen welke zorgt voor betrouwbare overkomst van de tekstberichten. Deze methode moet functioneren als volgt:

- Wanneer een bericht wordt verzonden krijgt het een uniek en oplopend volgnummer ("sequence number"). Het bericht wordt samen met dit volgnummer naar de andere partij gestuurd. Daarnaast wordt het bericht ook opgeslagen in een lokale datastructuur (bijv. een Python-list).
- Wanneer een bericht wordt ontvangen, wordt een bevestiging ("acknowledgment") voor dit volgnummer naar de zender gestuurd.
- Aan de kant van de ontvanger moeten berichten worden getoond in de volgorde waarin deze oorspronkelijk zijn verzonden. Wanneer een bericht wordt ontvangen, moet dus worden gecontroleerd dat het volgnummer precies één hoger is dan het vorige volgnummer. Als dat zo is, dan kan het bericht direct aan de gebruiker worden getoond. En anders moet het bericht worden opgeslagen in een lokale datastructuur **én** wordt aan de andere partij een verzoek gestuurd om het ontbrekende bericht nogmaals te sturen.
- Als een bevestiging wordt ontvangen voor een bericht, dan kan dat bericht worden verwijderd uit de lokale datastructuur.
- Indien een verzoek wordt ontvangen om een bericht (volgnummer) nogmaals te versturen, dan wordt dit bericht opgezocht in de lokale datastructuur middels het volgnummer en dan nog een keer verstuurd.
- De lokale datastructuren moeten regelmatig worden gecontroleerd: alle berichten waarvoor geen bevestiging is ontvangen worden nogmaals verstuurd; voor alle ontbrekende berichten wordt een verzoek gestuurd om dat bericht nogmaals te versturen.

Het programma moet vier command-line arguments ondersteunen:

```
python mijnprogramma.py <nickname> <IP of hostnaam> <port uitgaand> <port inkomend>
```

Toelichting:

- `nickname`: jouw nickname, welke de andere gebruiker te zien krijgt. De nickname wordt toegevoegd aan ieder bericht dat wordt verstuurd; je hoeft dus niet een protocol-message te implementeren om voorafgaand aan de chat de nickname te communiceren aan de andere partij.
- `IP of hostnaam`: een IP-adres of hostnaam van de computer waarmee moet worden gechat.
- `port uitgaand`: de UDP-poort waarnaartoe berichten zullen worden gestuurd, en waarop de ontvanger dus aan het luisteren is. Bijv. poort 5678.
- `port inkomend`: de UDP-poort waarop berichten zullen worden ontvangen (en waarnaartoe de andere partij dus berichten stuurt). Bijv. poort 5679.

Verwachte functionaliteiten

Samenvattend verwachten wij dat jouw programma de volgende functionaliteiten bevat:

1. De command-line arguments, zoals hierboven omschreven, worden correct afgehandeld.
2. Het programma heeft de mogelijkheid om een bericht in te voeren. Een bericht wordt afgesloten met een newline (Enter), waarna het wordt verzonden (zie punt 3). De lengte van de berichten kan worden gelimiteerd tot 300 karakters.
3. Voordat een bericht wordt verstuurd, wordt de nickname eraan toegevoegd. Een bericht dat wordt verstuurd wordt genummerd met een uniek en oplopend volgnummer. Berichten moeten naar de andere partij worden verstuurd met UDP en het volgnummer moet worden meegestuurd. Hoe je de berichten met nickname en volgnummer vormgeeft is een vrije keuze. De chatprogramma's van verschillende teams hoeven onderling **niet** compatible met elkaar te zijn.
4. Iedere keer wanneer een bericht wordt verstuurd, wordt er ook gekeken of er een bericht kan worden ontvangen.
5. Indien de gebruiker bij het invoeren van een bericht direct op Enter drukt, en het bericht dus leeg is, dan wordt er geen leeg bericht gestuurd maar wordt er gekeken of er een bericht kan worden ontvangen (net als punt 4).
6. Voor ieder ontvangen bericht wordt een bevestiging ("acknowledgment") verstuurd naar de verzender (middels UDP).
7. Berichten worden bewaard door de verzender totdat een bevestiging wordt ontvangen.
8. Ontvangen berichten worden aan de gebruiker getoond in de volgorde waarin deze oorspronkelijk zijn verzonden. Dit wordt bereikt door gebruik te maken van de volgnummers. Voor ontbrekende berichten wordt (middels UDP) een verzoek gestuurd om dat bericht nogmaals te sturen.
9. Indien een verzoek wordt ontvangen om een bericht nogmaals te sturen, dan wordt het corresponderende bericht nogmaals verstuurd.
10. Het invoeren van "quit" als bericht zal leiden tot het afsluiten van het programma. Dit bericht wordt ook naar de tegenpartij gestuurd. Bij ontvangst van "quit" zal de tegenpartij ook het programma afsluiten. Er hoeft **niet** te worden gewacht op een acknowledgment voordat het programma daadwerkelijk wordt afgesloten.
11. Maak het mogelijk dat berichten ook kunnen worden ontvangen en getoond terwijl er wordt gewacht op invoer van de gebruiker. Je hebt hiervoor een oplossing nodig om tegelijkertijd te kunnen wachten op invoer van de gebruiker (stdin) en binnenkomende netwerkberichten (socket).
12. Iedere 5 seconden (en dus **niet** alleen wanneer er een bericht wordt verstuurd) moeten beide kanten een scan uitvoeren van de lokale datastructuren. Berichten waarvoor geen bevestiging is ontvangen worden nogmaals verstuurd. Voor alle ontbrekende berichten wordt een verzoek gestuurd om dat bericht nogmaals te sturen.
13. Het invoeren van "file" als bericht heeft ook een speciale betekenis. Het keyword "file" moet worden gevolgd door een pad naar een lokaal bestand. Een TCP-verbinding moet worden opgezet naar de andere partij om het bestand te versturen. De andere partij moet de gebruiker informeren over het binnenkomende bestand en vragen naar een pad naar een lokaal bestand waar de ontvangen data moet worden opgeslagen. Het bestand wordt overbracht over de TCP-verbinding, waarna de verbinding wordt gesloten.

Indiening en beoordeling

De opdracht mag individueel worden gemaakt of in een team bestaande **uit ten hoogste twee personen**. De opdracht moet worden ingediend volgens onderstaande instructies. De deadline is vrijdag 12 december 2025, 23:59 uur. Het maximaal behaalbare cijfer is 10. De rubric die zal worden gebruikt om de opdracht te beoordelen is beschikbaar op Brightspace. Het behaalde aantal punten (maximaal 100) zal worden gedeeld door 10 om het cijfer te berekenen.

Hetgeen dat moet worden ingestuurd is de Python source code voor jouw project. Zorg ervoor dat alle bestanden die je gaat insturen zijn voorzien van naam en studentnummer!

Indien je een enkel Python-bestand instuurt, geef het dan de naam:

```
assignment2-sXXXXXXX-sYYYYYYY.py
```

waar voor `XXXXXXX` en `YYYYYYY` jullie studentnummers worden ingevuld. Als jouw project bestaat uit meerdere bestanden, maak dan een ZIP-bestand of tar.gz-bestand met naam:

```
assignment2-sXXXXXXX-sYYYYYYY.zip
```

```
assignment2-sXXXXXXX-sYYYYYYY.tar.gz
```

Dien je bestand in via de Brightspace assignment submission site. Geef bij het indienen van je opdracht in Brightspace in het tekstvak jullie namen en studentnummers aan. Indien je werkt in een team van twee, dan hoeft maar één teamlid de opdracht in te dienen, zodat er één submissie is per team.

Appendix

In deze appendix geven we enige tips voor het werken aan deze opdracht.

Twee chats op dezelfde computer

Voor het testen en ontwikkelen is de meest eenvoudige optie om beide chats op dezelfde computer te draaien. Je kunt met je lokale computer verbinden via de hostnaam "localhost" of IP-adres 127.0.0.1. Om te testen, open je twee terminals of command-vensters. In het ene voer je uit:

```
python chatprogramma.py naamA localhost 5678 5679
```

en in de andere:

```
python chatprogramma.py naamB localhost 5679 5678
```

IP-adres

Het is natuurlijk nog leuker als de twee chat-programma's op twee verschillende computers worden uitgevoerd. Dit kan werken als de twee computers zich op hetzelfde lokale netwerk (bijv. een WiFi-netwerk thuis) bevinden. Theoretisch gezien werkt het programma voor iedere twee computers die via Internet met elkaar zijn verbonden, maar praktisch gezien zal dat helaas tegenvallen door firewalls e.d.

Via de user interface van jouw besturingssysteem kun je het IP-adres van jouw computer opzoeken bij de netwerkinstellingen. Dit adres vul je dan in op de plaats van "localhost". Je kunt het adres ook via een command-line tool achterhalen: "ipconfig" op Windows, "ip addr" op Linux, en "ifconfig" op macOS.

Op Eduroam is het waarschijnlijk niet mogelijk om het chat-programma tussen twee verschillende computers te gebruiken, dit komt door een firewall. Een WiFi-netwerk thuis werkt wel. En ander alternatief is om een hotspot te maken met je smartphone en twee laptops daarmee te laten verbinden. Door een hotspot te maken wordt er in feite een lokaal netwerk gemaakt.

Hoe verstuur je berichten via UDP in Python?

Voor het ontvangen en versturen van berichten met UDP in Python moet je gebruik maken van de `socket` module. Je moet hierbij eerst een socket object aanmaken, met *address family* AF_INET (je gebruikt IPv4 adressen of iets wat ertoe kan worden herleid) en type SOCK_DGRAM (een UDP socket).

Zodra je een socket object hebt, kun je berichten gaan versturen met behulp van de `sendto` functie op de socket. Zo stuur je met `socket.sendto(b"Hello, world!", ("localhost", 5678))` het berichtje "Hello, world!" naar localhost op port 5678.

Om berichtjes te ontvangen moet je iets meer doen. Eerst moet je eenmalig aan het operating system vertellen dat je op een bepaalde port wil gaan luisteren, met behulp van de bind functie, dus `socket.bind(("localhost", 5678))` om op port 5678 te gaan luisteren op localhost. Vervolgens kun je een berichtje ontvangen met behulp van de `recv` methode, die je de maximum hoeveelheid data moet meegeven. Dus als je `msg = socket.recv(1000)` doet, dan ontvang je maximaal 1000 bytes aan data in msg.

Onbetrouwbaar netwerk simuleren

Zoals is omschreven in de opdracht moet je zelf een protocol implementeren om te zorgen voor betrouwbare overkomst van de berichten. Een eerste interesse test is het volgende:

- Start één chat-programma op, we noemen dit A, maar het chat-programma B voor de andere kant nog niet!
- Verstuur al een aantal berichten met A. Dus zullen dus nog niet aankomen, omdat chat-programma B nog niet draait.
- Start nu chat-programma B op en verstuur vanuit B een bericht naar A.
- Wanneer chat-programma A nu dat bericht ontvangt, zou het moeten opmerken dat het nooit acknowledgments heeft ontvangen voor de berichten die het eerder naar B heeft verstuurd. Het programma A verstuurt deze berichten nogmaals.

Om dit uitgebreider te testen, is het nuttig als je zelf een onbetrouwbaar netwerk kunt simuleren. Dit kun je bereiken met de Python `random` module. De `random.random()` functie geeft een willekeurige waarde tussen 0 en 1 terug. Je kunt dan bijvoorbeeld alleen een pakketje versturen als de waarde groter is dan 0.1, om te simuleren dat 10% van je pakketjes niet aankomen. Dit is een eenvoudige manier om te testen of je applicatie bestand is tegen het verliezen van pakketjes op verschillende momenten.