mrežno programiranje

miša stefanović
mikmik1011@proton.me

uvod:

- socket programiranje uključuje komunikaciju između dva računara putem socketa
- klijent-server arhitektura gde jedna strana (server) pruža usluge, a druga strana (klijent) ih koristi

klijent-server arhitektura:

- klijent: inicira zahteve za uslugama ili resursima
- server: sluša zahteve klijenata, obrađuje ih i pruža odgovore
- omogućava distribuirano računarstvo, efikasno iskorišćavanje resursa i modularni dizajn

primer:

- aplikacija = kafić
- klijenti = mušterije
- server = konobar

socketi:

- endpointi za slanje i primanje podataka preko mreže
- mogu biti razni, kao sto su ipv4, bluetooth, unix...
- ipv4 socketi su identifikovani prema ip adresi i broju porta
- tipovi ipv4 socketa su tcp i udp

tcp (transmission control protocol):

- pouzdan protokol sa konekcijama
- osigurava isporuku podataka i sekvenciranje
- prikladan za aplikacije koje zahtevaju integritet podataka (npr. transfer fajlova, email)
- syn i ack paketi za uspostavljanje konekcije

udp (user datagram protocol):

- lightweight protokol bez konekcija
- brži, ali može dovesti do gubitka podataka ili isporuke van redosleda
- koristi se za aplikacije u realnom vremenu (npr. video strimovanje, online igrice)

socket api:

- programski interfejs za socket komunikaciju
- funkcije za kreiranje, vezivanje, slušanje, prihvatanje, povezivanje, slanje i primanje podataka
- različiti jezici pružaju svoje sopstvene biblioteke za sockete (npr. python-ov modul `socket`)

workflow rada sa socketima:

- server:
  - kreiranje socketa korišćenjem socket api-ja
  - vezivanje socketa za ip adresu i port
  - slušanje dolaznih klijentskih veza
  - prihvatanje klijentskih veza
  - primanje i obrada zahteva klijenata, slanje odgovora nazad
- klijent:
  - kreiranje socketa korišćenjem socket api-ja
  - povezivanje sa ip adresom i portom servera
  - slanje zahteva serveru
  - primanje i obrada odgovora servera

prednosti socketa:

- efikasna razmena podataka između procesa na različitim mašinama
- fleksibilnost u dizajniranju mrežnih aplikacija
- omogućava komunikaciju u različitim domenima (web, igrice, iot itd.)

izazovi:

- postupanje sa greškama u mreži i osiguravanje pouzdanosti podataka
- skaliranje kako broj klijenata i obim podataka raste
- bezbednosni problemi, uključujući enkripciju podataka i autentifikaciju

primeri iz stvarnog sveta:

- surfovanje web-om (http), email (smtp), transfer fajlova (ftp) koriste sockete
- online multiplayer igre koriste sockete za interakciju u realnom vremenu
- aplikacije za video konferencije se oslanjaju na sockete za audio-video strimovanje
- u suštini, ceo internet ispod haube koristi sockete

dosta dosadne teorije, ajmo sada malo da kuckamo.

first things first:

python virtual environments (venv)

prednosti:

- izolovano python okruženje
- specifični dependency-ji
- reproduktivnost
- kompatibilnost verzija modula

python venv

kreiranje:

komanda u terminalu: `python -m venv venv`

aktivacija:

- linux/macos: `source venv/bin/activate`
- windows: `.\venv\scripts\activate.bat` (ili `.ps1` na kraju za powershell)

pip

- pythonov package manager
- služi za upravljanje paketima/modulima

usage:

- `pip install ime_paketa` - za instaliranje pojedinačnog paketa
- `pip install -r requirements.txt` - za instaliranje paketa iz fajla

zadatak 1:

napraviti udp klijent/server arhitekturu, gde klijent šalje poruku serveru i server ga pozdravlja.

bitne stvari za ovaj zadatak:

- af_inet = internet address family
- sock_stream = tcp
- sock_dgram = udp
- setsockopt(level/grupa, opcija, value)
- so_reuseaddr - bypassuje cooldown nakon korišćenja porta, pogodno za restartovanje

zadatak 2:

pomoću udp protokola napraviti osnovnu chat klijent/server aplikaciju.

bitne stvari za ovaj zadatak pt.2:

threading (jako tl;dr eli5 objašnjenje)

- način da se više nekih funkcija izvršava istovremeno
- mi ćemo za svakog korisnika pokrenuti funkciju koja handluje njegove poruke
- u pythonu za to koristimo biblioteku `threading`

zadatak 3:

pomoću tcp protokola napraviti klijent/server aplikaciju, gde server vraća cowsay-ovan tekst koji mu je klijent poslao.

bitne stvari pt.3

- cowsay - komanda koja ubacuje kravu i oblačić na tekst
- želimo da omogućimo prenos većih tekstova
- udp dropuje višak teksta, tcp čita kao novu poruku
- hoćemo samo jedan veliki cowsay umesto više malih
- možemo da pošaljemo serveru prvo veličinu teksta (header), pa onda tekst (body)
- python socket ima buffer limit 64kb
- koristićemo chunking da to sve lepo učitamo

zadatak 4

pomoću tcp protokola napraviti klijent/server aplikaciju, gde server cowsayu-je tekst ili deepfry-je sliku, u zavisnosti od sadržaja

bitne stvari pt. 4

- želimo da u headeru pošaljemo više različitih podataka, kao što su ime, tip slike, tip životinje ako je tekst, itd.
- pravljenje našeg headera je veoma komplikovan i bespotreban proces
- idealno želimo da pošaljemo sve to kao python dictionary

neke od mogućih alternativa

- pythonov struct
- json
- pickle

najbolja alternativa za ovaj use case

msgpack

- možemo da lako spakujemo dict u bajtove, koji će zadržati imena, omogućiti kakve god tipove podataka, neće biti vulnerable pri deserijalizaciji i zauzimati mnogo malo prostora

bitne stvari pt. 4.2

async/asyncio (opet tl;dr i eli5)

- neblokirajuć način za rad sa fajlovima
- slično kao threadovi, ali optimizovano za i/o stvari

wireshark

- program koji služi za snimanje i analizu socket saobraćaja
- dobar za debugovanje ili security forenziku, reverse engineerovanje itd