Komunikace v síti - tvorba síťových aplikací, Berkley socket a jeho rozhraní

Komunikace v síti obecně
Umožňuje přenos dat mezi dvěma nebo více zařízeními přes síť. Klíčkové pro funkčnost webu, komunikace se serverem, databáze….

Základní pojmy
Server - naslouchá a přijímá připojení - responses, program běžící někde na počítači, který čeká na requesty, multitasking, stabilní ip a známý port
Klient - připojuje se k serveru a odesílá requesty na server - requests, program iniciující requesty, může být připojený k více serverům
Protokol - soubor pravidel, kterým se řídí přenos dat mezi zařízeními, definují formát, pořadí a význam zpráv, které si zařízení vyměňují

Nejdůležitější síťové protokly:
TCP (Transmision control protocol)
- garantuje spolehlivé doručení a správné pořadí
- používá handshake - něco jako garance připojení
- stavový protokol - udržuje informaci o spojení
- zajišťuje kontrolu toku dat a řízení přehlcení
- dělí data na segmenty a znovu je sestavuje
- pomalejší, ale spolehlivější
- ACK packety atd.

UDP (User datagram protocol)
- negarantuje doručení a neřeší pořadí
- nespojovaný protokol - nenavazuje formální spojení
- neudržuje stav připojení
- overhead
- nezajišťuje kontrolu toku ani zahlcení
- vhodnější pro aplikace, kde je rychlost důležitější než spolehlivost (hry, streamování, VoIP) - je rychlejší

Adresování v síti
IP adresa - identifikátor zařízení v síti, ipv4,6, každé zařízení má unikátní v síti
Port - určení konkrétní služby, číselná hodnota od 0-65535, známý porty 80, 443, 21, 22, 53….
Socket - koncový bod síťové komunikace, který umožňuje připojení v síti,

Tvorba síťových aplikací
V pythonu existují 2 přístupy způsoby tvorby síťových aplikací:

1. Low-level programming (Socketové programování) - pracuje se přímo se sokety, poskytuje větší kontrolu, ale je pracnější
2. High-level programming - využívají se knihovny, http.client, https.server, asyncio, requests - tyto knihovny abstrahují detaily síťové komunikace

Postup práce se sockety:

1. import knihovny socket
   import socket
2. vytvoření socketu pomocí funkce socket.socket()
   s = socket.socket(socket_family, socket_type, protocol=0)
   socket_family - AF_INET - IPv4, AF_INET6 - IPv6
   socket_type - SOCK_STREAM - TCP, SOCK_DGRAM - UDP
   protocol 0 - operační systém automaticky vybere správný protokol podle typu socketu
3. přiřazení IP adresy a portu pomocí funkce bind()
   server_inet_adress = ("127.0.0.1", 5)
   s.bind(server_inet_adress)
4. nastavení socketu pro naslouchání příchozím připojením pomocí funkce listen()
   s.listen()
5. příjmutí připojení pomocí funkce accept()
   client_socket, client_inet_address = s.accept()
   metoda vrací nový socket objekt a adresu klienta, původní socket nadále naslouchá, nový socket používá pro komunikaci
6. komunikace pro odesilátele a přijímání dat
   send() - poslání TCP zprávy, sendall() - bezpečnější, odešle vše nebo vyhodí výjimku
   sendto() - poslání UDP zprávy
   recv() - přijmutí TCP zprávy
   recvfrom - přijmutí UDP zprávy
   client_socket.send("Hello from Server!".encode()) data = client_socket.recv(1024).decode("utf-8").strip()
7. uzavření spojení, musí se zavřít klient i server
   client_socket.close()

Příklad - TCP echo server jednoduchej

```
import socket

def start():
   server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   server_adress = ("127.0.0.1", 8080)

   server_socket.bind(server_adress)
   server_socket.listen(5)

   while True:
       client_socket, client_address = server_socket.accept()
       try:

           while True:
               data = client_socket.recv(1024)
               print(f"Received {data}")

               if not data:
                   break
               else:
                   client_socket.send(data)
       finally:
           client_socket.close()

if __name__ == "__main__":
   start()
```

Nevim jestli to bude možný se u maturity na to připojit přes puttynu, docela by to uhlecilo, ale kdyztak tady je nejakej ez client jeste

```
import socket

def klient():
   klient = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   klient.connect(('127.0.0.1', 8080))

   while True:
       user_input = str(input())
       klient.send(user_input.encode())

       answer = klient.recv(1024).decode()
       print(answer)

   klient.close()


if __name__ == "__main__":
```

```
klient()
```

Barkley socket

BSD socket, standardizované programové rozhraní (API) pro síťovou komunikaci. Tento standard se stal základem pro síťové programování skoro ve všech moderních OS. Umožňuje programům vytvářet koncové body (sockety) a používat je pro přenos dat po síti. V pythonu je modul socket postaven na BDS API, python za pomocí barkley socketu poskytuje jednodušší přístup.

jsou to ty funkce jako listen(), bind(), accept(), send()..... + ty adresy AF_INET + ty socket SOCK_STREAM, SOCK_DGRAM….