Lab 10: Komunikacja sieciowa

Moduł `socket` w Pythonie udostępnia interfejs do Berkeley sockets API i jest wykorzystywany do komunikacji sieciowej.

Podstawowe funkcje i metody w tym module to:

- `socket()`, `bind()`, `listen()`, `accept()`,
- `connect()`, `connect_ex()`, `send()`, `recv()`, `close()`.

Python zapewnia spójne API, które odwzorowuje systemowe wywołania w C. Istnieją także klasy ułatwiające korzystanie z tych funkcji, np. moduł `socketserver`, który upraszcza tworzenie serwerów sieciowych. Python obsługuje również wyższe protokoły, takie jak HTTP i SMTP.

Dlaczego TCP? Protokół ten:

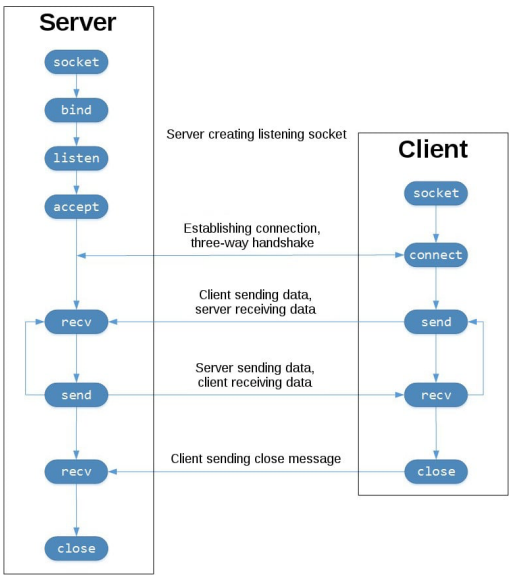
- **Jest niezawodny** – wykrywa i retransmituje zgubione pakiety.
- **Gwarantuje kolejność danych** – odbiorca otrzymuje dane w takiej samej kolejności, w jakiej zostały wysłane.

W przeciwieństwie do tego, gniazda UDP (`socket.SOCK_DGRAM`) nie zapewniają niezawodności ani kolejności danych – jednak zapewniają większą wydajność dzięki mniejszym narzutom czasowym związanym z obsługą komunikacji.

Dlaczego to ważne?

Sieci nie gwarantują dostarczenia danych – mogą wystąpić opóźnienia, zgubione pakiety czy ograniczenia sprzętowe. TCP automatycznie zarządza tymi problemami, zapewniając stabilną komunikację.

Schemat typowej komunikacji klient – serwer przedstawiono na rysunku poniżej:



Lewa kolumna reprezentuje serwer, a prawa klienta.

W górnej części po lewej stronie znajdują się wywołania API, które serwer wykonuje, aby utworzyć gniazdo „nasłuchujące”:

- `socket()`
- `.bind()`
- `.listen()`
- `.accept()`

Gniazdo nasłuchujące działa zgodnie z nazwą – oczekuje na połączenia od klientów. Gdy klient się połączy, serwer wywołuje `.accept()`, aby zaakceptować połączenie.

Klient wywołuje `.connect()`, aby nawiązać połączenie z serwerem i rozpocząć tzw. **handshake** (trójetapowe uzgadnianie połączenia). Ten proces zapewnia, że każda strona połączenia jest osiągalna w sieci.

W środkowej części następuje wymiana danych między klientem a serwerem przy użyciu `.send()` i `.recv()`.

Na końcu zarówno klient, jak i serwer zamykają swoje gniazda.

Zadanie: Uruchom przykład z: https://docs.python.org/3/library/socket.html#example i zaobserwuj działanie. Zmień port, na którym komunikują się programy. Czy każdy dowolny port jest dostępny do komunikacji?

Zadania do samodzielnego wykonania:

Zadanie 1 - Prosty klient-serwer (powitanie)

Stwórz aplikację składającą się z dwóch programów: serwera i klienta.

- **Serwer:**
  - Czeka na jedno połączenie.
  - Po nawiązaniu połączenia czeka na wiadomość od klienta.
  - Jeśli klient wyśle wiadomość `"Hello"`, serwer odpowiada wiadomością `"Hello from server"` i kończy połączenie.
  - Jeśli klient wyśle wiadomość `"Time"`, to serwer odpowiada wysyłając klientowi aktualny czas.
  - Jeśli to inna wiadomość, serwer odpowiada „`Unknown command`”.
- **Klient:**
  - Łączy się z serwerem.
  - Wysyła wiadomość `"Hello"`.
  - Odbiera odpowiedź z serwera i wyświetla ją na ekranie.

Zadanie 2 - Gra „Papier, Nożyce, Kamień” (PvP do 3 wygranych)

Zaimplementuj serwer TCP umożliwiający rozgrywkę pomiędzy dwoma użytkownikami.

- Serwer czeka na połączenie dokładnie dwóch graczy.
- Każdy gracz wybiera jedną z opcji: „papier”, „nożyce” lub „kamień” i wysyła ją na serwer.
- Po otrzymaniu wyboru od obu graczy serwer:
  - Oblicza wynik rundy.
  - Wysyła do obu graczy informację o wyborze przeciwnika oraz wynik rundy.
- Rozgrywka trwa do momentu, gdy jeden z graczy osiągnie 3 zwycięstwa.
- Po zakończeniu gry serwer informuje graczy o wyniku końcowym i zamyka połączenia.

Zadanie 3 - Chat wieloosobowy z administracją

Stwórz prostą aplikację czatu TCP umożliwiającą komunikację wielu użytkowników jednocześnie:

- Każdy klient po podłączeniu musi podać swój nick.
- Każda wiadomość wysłana przez klienta jest rozsyłana do wszystkich podłączonych użytkowników.
- Osoba korzystająca z aplikacji serwera występuje jako `admin` i także może pisać wiadomości.
- Admin posiada dodatkowo specjalną komendę `/kick <nick>` umożliwiającą wyrzucenie wybranego użytkownika z czatu.
- Każde zdarzenie (dołączenie, opuszczenie czatu, wyrzucenie użytkownika) powinno być komunikowane wszystkim uczestnikom.