Distributed Systems, 3rd edition, MvS&AST, *Architectures*

Przykład: dwie strony komunikacji

Opracował Maksymilian Zawartko

Przykład ma na celu uwydatnienie różnic między usługą (service), międzymordziem (interface) i protokołem.

Serwer:

```
from socket import *
s = socket(AF_INET, SOCK_STREAM)
(conn, addr) = s.accept()  # returns new socket and addr. client
while True:  # forever
        data = conn.recv(1024)  # receive data from client
        if not data:
                break  # stop if client stopped
        conn.send(str(data)+"*")  # return sent data plus an "*"
conn.close()  # close the connection
```

Klient:

```
from socket import *
s = socket(AF_INET, SOCK_STREAM)
s.connect((HOST, PORT))  # connect to server (blocking)
s.send('Hello, world')  # send some data
data = s.recv(1024)  # receive the response
print data  # print the result
s.close()  # close the connection
```

Serwer używa zorientowanych na komunikację (communication-oriented) funkcjonalności pythonowej biblioteki socket, co umożliwia obu stronom niezawodną komunikację, wysyłanie i odbieranie danych.

Najważniejsze funkcje dostępne w bibliotece socket

- socket() - tworzy obiekt reprezentujący połączenie
- accept() - blokująca funkcja oczekująca na połączenie; jeśli takowe nadejdzie, zwraca nowe gniazdo (socket) dla tego połączenia
- connect() - inicjalizuje połączenie z zadanym odbiorcą / nadawcą
- close() - zamyka połączenie
- send(), recv() - służą odpowiednio do wysyłania i odbierania danych

Stałe AF_INET i SOCK_STREAM

są używane, by komunikacja nastąpiła za pomocą protokołu TCP. Jednak to, jak ten protokół działa, lub jakiekolwiek inne szczegóły implementacyjne, są ukryte przed aplikacjami wykorzystującymi bibliotekę socket.