

Mrežno programiranje

Laboratorijska vježba br.3



U ovom laboratorijskoj vježbi ćemo detaljnije proći pojam priključnica, tj. *socketa*.

NAPOMENA: Kod je pisan u Pythonu 2, ukoliko koristite Python 3 potrebno je koristiti `encode()` funkciju kod nekih funkcija (error byte to string conversion kada se desi)

Osnove socketa

Do sada, kroz kolegij Računalne mreže i mrežne usluge smo prošli osnovne koncepte računalnih mreža, a s prvom laboratorijskom vježbom smo prošli osnovne karakteristike Python programskog jezika.

Da bi razumjeli što je socket, počet ćemo s Internet konekcijom. Internet konekcija spaja dvije točke, tj. dva čvora preko Interneta u svrhu razmjene podataka.

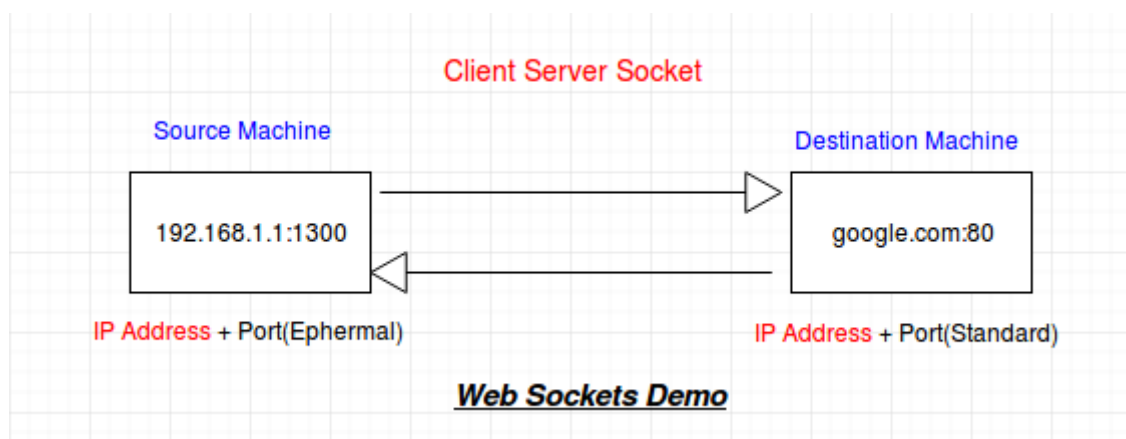
Jedan proces s računala PC1 može komunicirati s procesom na računalu PC2 preko takve konekcije.

Svojstva takve komunikacije su:

- Pouzdanost
- Point-to-point
- Full-duplex

Priključnice ili **socketi** su krajnje točke bidirekcionalnog, point-to-point komunikacijskog kanala. Ako za primjer uzmemo konekciju između klijenta (web preglednik) i servera (npr. www.google.com), imat ćemo dvije priključnice. Klijentski socket i Serverski socket.

Socket se sastoji od dva dijela: **IP adresa + Port**



Generalno gledajući, naprimjer, svaki put kad odete na www.google.com, browser napravi sljedeće:

```
>>> import socket
>>> #socket objekt je kreiran za komunikaciju
...
>>> socket_na_klijentu = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
>>> #spajanje na web server na port 80
...
>>> socket_na_klijentu.connect(("www.google.com", 80))
```

Ovo je napravljeno na klijentskoj strani. Kada se klijent pokuša spojiti na server, toj konekciji operacijski sustav dodijeli random port. Taj random port se naziva **Ephemeral Port**. Na gornjoj ilustraciji, 1300 je ephemeralni port na klijentu. Socket na klijentu je kratkog „životnog“ vijeka, naprimjer, čim su podaci razmijenjeni on se zatvori.

Na serveru se događa sličan, ali ipak malo različit proces u odnosu na klijenta.

```
>>> import socket
>>> socket_na_serveru = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
>>> #povezivanje ili bindanje socketa na javni host i poznati port
...
>>> socket_na_serveru.bind((socket.gethostname(), 8080))
>>> #postani serverski socket i slušaj dolazne konekcije
...
>>> socket_na_serveru.listen(5)
>>>
```

Navedeno možemo vidjeti i u izlistu TCP konekcija:

python.exe	14216	TCP	wmat1750.mshom... 8080	WMAT1750	0	LISTENING
------------	-------	-----	------------------------	----------	---	-----------

Za sada ćemo se fokusirati samo na *connect* i *bind* metode.

Connect koristi klijentski socket za uspostavljanje konekcije sa serverom. Taj zahtjev ispunjava *bind* na serverskom socketu.

Razlike između klijentskih i serverskih socketa:

- Za razliku od klijentskih socketa serverski nisu kratkotrajni. Naprimjer; možda vi trebate jednokratno ili s vremena na vrijeme pristup Aspirinoj web stranici, ali ta stranica mora biti dostupna cijelo vrijeme za bilo kojih zahtjev koji stigne od bilo kojeg korisnika

- Za razliku od klijentskih socketa koji koriste ephemeral port za ostvarivanje konekcije, serverski socket zahtijeva standardni ili dobro definirani port za konekciju kao što je naprimjer port 80 za normalnu HTTP konekciju.

Socketi u Pythonu

Razlikujemo dva tipa socketa: `SOCK_STREAM` i `SOCK_DGRAM`.

SOCK_STREAM	SOCK_DGRAM
ZA TCP PROTOKOL	ZA UDP PROTOKOL
POUZDANA ISPORUKA	NEPOUZDANA ISPORUKA
GARANTIRA ISPRAVAN REDOSLIJED PAKETA	NE GARANTIRA REDOSLIJED
KONEKCIJSKI ORIJENTIRAN	NIJE KONEKCIJSKI ORIJENTIRAN
BIDIREKCIONALAN	NIJE BIDIREKCIONALAN

Za kreiranje konekcije u Pytonu moramo koristiti `socket.socket()` funkciju koja je dostupna u Python socket modulu koji ima generalnu sintaksu sljedećeg oblika:

```
S = socket.socket(socket_family, socket_type, protocol=0)
```

- **socket_family:** može biti ili `AF_UNIX` ili `AF_INET`. Govoriti ćemo samo o `INET` socketima pošto oni zauzimaju više od 99% upotrebe.
- **socket_type:** može biti ili `SOCK_STREAM` ili `SOCK_DGRAM`.
- **protocol:** obično se izostavi što rezultira defaultiranjem na 0

Socket metode

connect()

Klijentska metoda koji se koristi za spajanje na udaljeni socket na nekoj adresi. Adresni format (host, port) se koristi za AF_INET adrese.

bind()

Serverska metoda koja binda socket na adresu. Format adrese također ovisi o AF_INET adresama.

listen(backlog)

Serverska metoda koja sluša konekcije koje se spajaju na socket. Backlog je maksimalan broj konekcija u redu čekanja koje moraju biti poslušane prije odbijanja konekcije.

accept()

Serverska metoda koja se koristi za prihvatanje konekcije. Socket mora biti bindan na adresu i mora slušati konekcije. Return vrijednost je pair(conn, address) gdje je *conn* novi socket objekt koji može biti iskorišten za slanje i primanje podataka na toj konekciji, a adresa je adresa bindana na socket na drugom kraju konekcije.

; za detaljan opis metoda pogledajte Python dokumentaciju:

<https://docs.python.org/2/library/socket.html>

RAD SA SOCKETIMA

Napraviti ćemo jednostavan klijent-server program.

Pripremite sljedeći program i pokrenite ga u Powershellu, cmd-u ili nekom drugom IDE-u. Po pokretanju program neće raditi u tom trenutku ništa osim što će osluškivati dolazne konekcije. Slično tome, svaka web stranica koju posjetite ima server na kojem je hostana, a koji uvijek čeka klijente da se spoje, tj. osluškuje konekcije.

```
#tcp_server.py

import socket

server_socket = socket.socket()
host = socket.gethostname()
port = 9999

server_socket.bind((host,port))
print "Waiting for connection..."
server_socket.listen(5)

while True:
    conn,addr = server_socket.accept()
    print 'Got Connection from', addr
    conn.send('Server Saying Hi')
    conn.close()
```

Niže se nalazi program za klijenta. Klijent se pokušava spojiti na serverski port (9999).

Pitanje: što radi linija koda `client_socket.connect((host,port))`?

Odgovor:

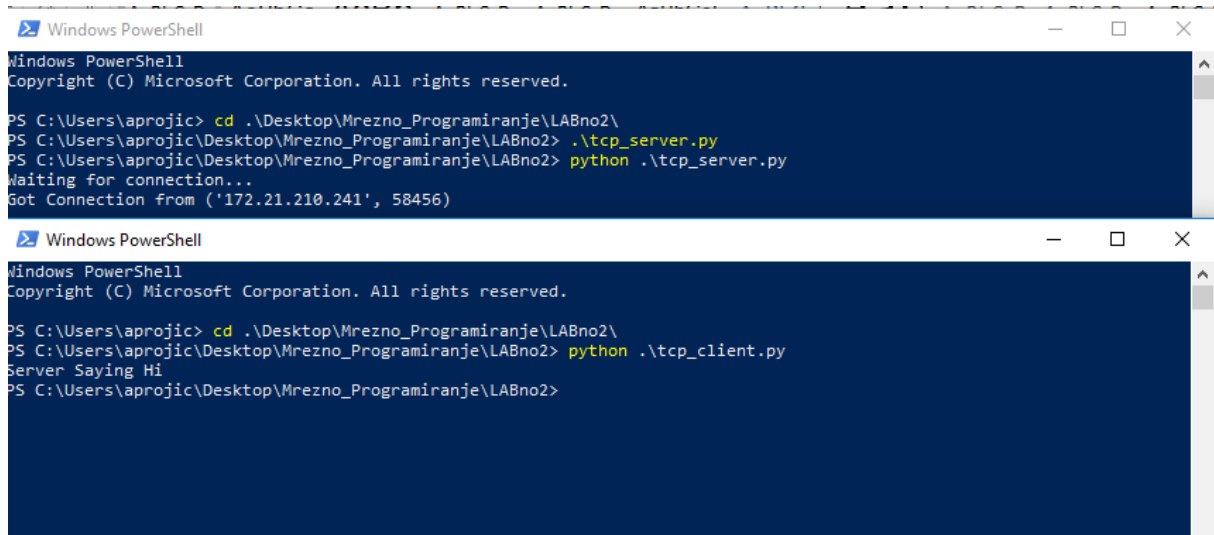
```
#tcp_client.py
import socket

client_socket = socket.socket()
host = socket.gethostname()
port = 9999

client_socket.connect((host,port))
print client_socket.recv(1024)

client_socket.close()
```

U dva odvojena Powershella pokrenite oba programa. Ako je sve u redu trebali bi dobiti sljedeće:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

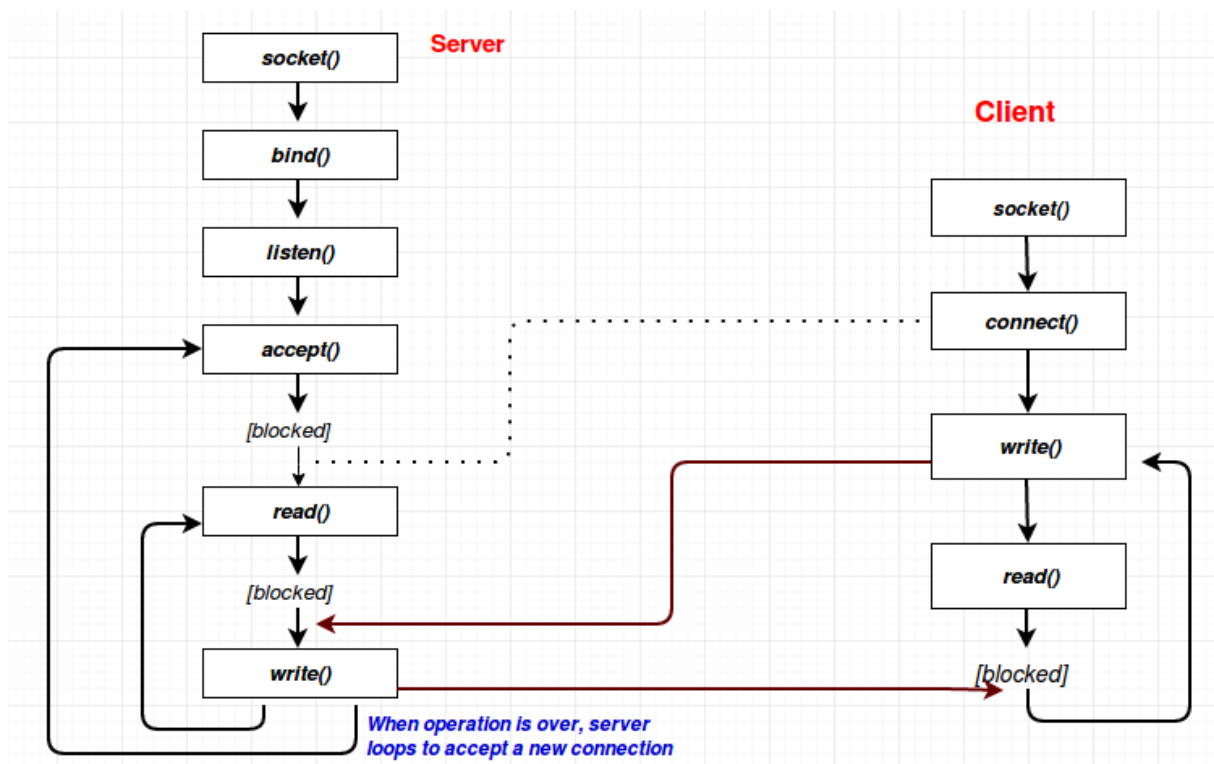
PS C:\Users\aprojic> cd .\Desktop\Mrežno_Programiranje\LABno2\
PS C:\Users\aprojic\Desktop\Mrežno_Programiranje\LABno2> .\tcp_server.py
PS C:\Users\aprojic\Desktop\Mrežno_Programiranje\LABno2> python .\tcp_server.py
Waiting for connection...
Got Connection from ('172.21.210.241', 58456)

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\aprojic> cd .\Desktop\Mrežno_Programiranje\LABno2\
PS C:\Users\aprojic\Desktop\Mrežno_Programiranje\LABno2> python .\tcp_client.py
Server Saying Hi
PS C:\Users\aprojic\Desktop\Mrežno_Programiranje\LABno2>
```

Pitanje: U serverskom dijelu vašeg programa, što znači drugi broj u zagradama kod ispisa „Got connection...”?

Odgovor:



ZAKLJUČNO

Kreirajte .py skripte s navedenim kodom te ih pokrenite (**OBAVEZNO IZABRATI NEKI DRUGI PORT U ODNOSU NA PRIMJER U VJEŽBI – između 1024 i 49151!!**). Dostavite kod i odgovore za pitanja unutar laboratorijske vježbe (2 su).

Priložite izlist aktivnih konekcija na koji god način hoćete (netstat, tcpconn, itd..) gdje se vidi na kojem portu vaš serverski program sluša.

Modificirajte tcp_client program na način da otvorite socket konekciju prema www.google.com i priložite kod. Ispis može biti sličan ovome:

```
PS C:\Users\aprojic\Desktop\Mrežno_Programiranje\LABno2> python .\tcp_client_google.py
IP for hostname www.google.com is: '216.58.209.196'
The socket has successfully connected to Google on port == 80, and IP address == 216.58.209.196
PS C:\Users\aprojic\Desktop\Mrežno_Programiranje\LABno2>
```

OBAVEZNO: sav kod mora biti pushan na vaš GIT repo iz vježbe 2.

NAPOMENA: Kod je pisan u Pythonu 2, ukoliko koristite Python 3 potrebno je koristiti encode() funkciju kod nekih funkcija (error byte to string conversion kada se desi)