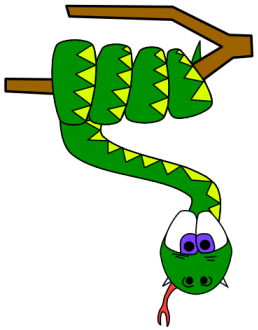


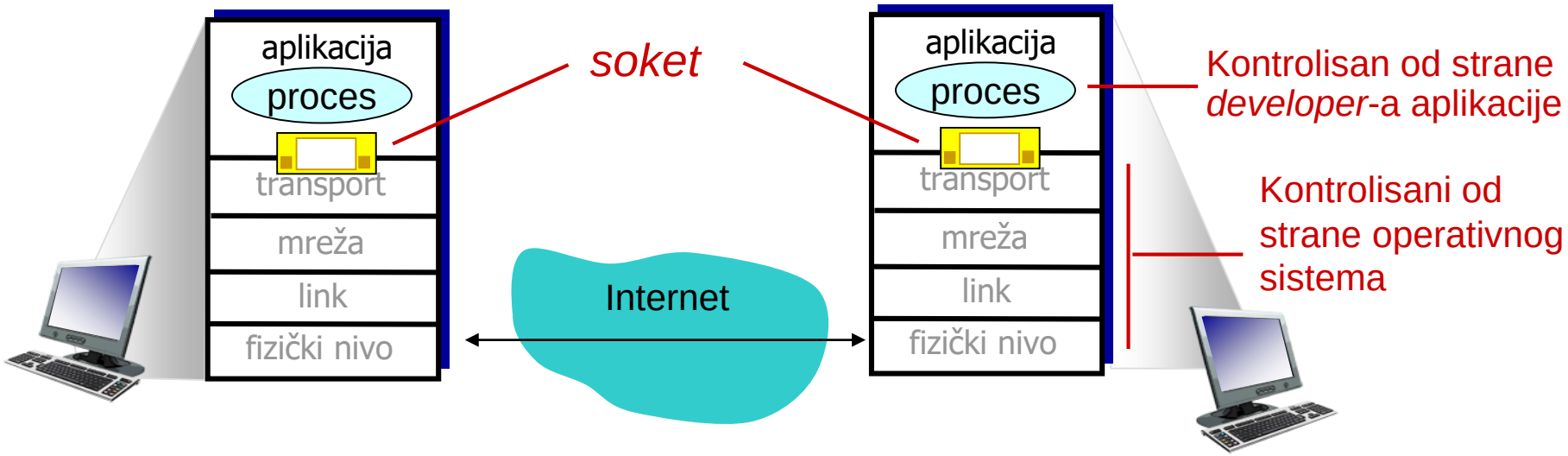
Uvod u mrežno programiranje

Slavica Tomović (slavicat@ucg.ac.me)
Elektrotehnički fakultet, Podgorica
Univerzitet Crne Gore

Programiranje soketa

cilj: naučiti kako se razvijaju klijent/server aplikacije koje komuniciraju preko soketa

soket: krajnje tačke *end-to-end* transportne konekcije između dva aplikacijska procesa



Programiranje soketa

Dva tipa soketa za dva tipa transportnih servisa:

UDP: nepouzdan prenos

TCP: pouzdan prenos, *stream* bajtova

Primjer Aplikacije:

1. Klijent čita liniju podataka unesenih putem tastature i šalje je serveru.
2. Server prihvata podatke i kovertuje karaktere u velika slova (*uppercase*).
3. Server šalje modifikovane podatke klijentu.
4. Klijent prihvata modifikovane podatke i štampa ih na ekranu.

Programiranje UDP soketa

UDP: nema "konekcije" između klijenta i servera

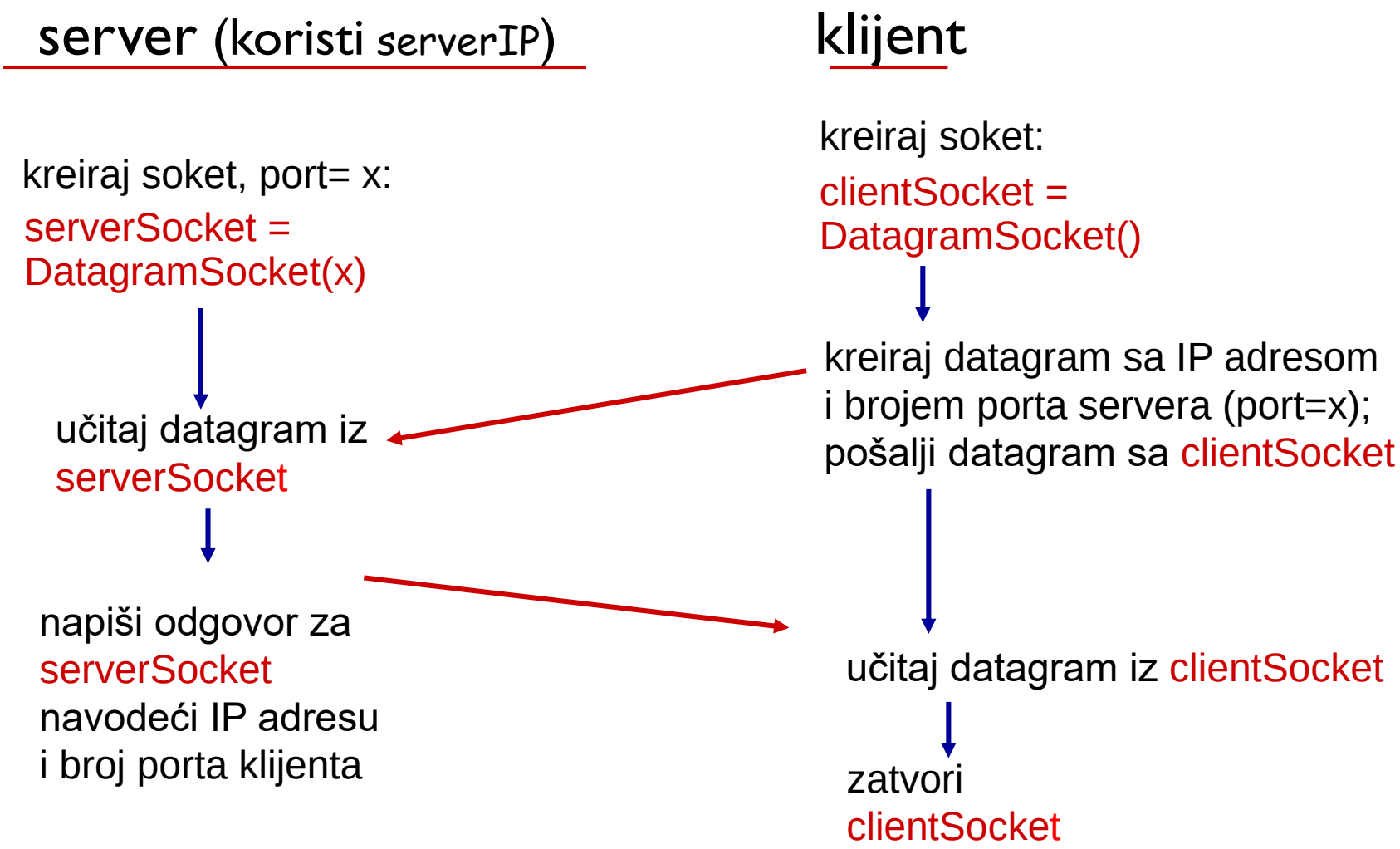
- nema kontrolne komunikacije prije slanja podataka
- pošiljalac eksplicitno dodaje destinacionu IP adresu i broj porta svakom paketu
- primalac ekstraktuje IP adresu i broj porta pošiljaoca iz primljenog paketa

UDP: poslati podaci se mogu izgubiti ili primiti neredosledno

Sa aspekta aplikacije:

- UDP pruža *nepouzdan* prenos grupe datagrama između klijenta i servera

Klijent-server interakcija: UDP



Primjer aplikacije: UDP klijent

Python UDPClient

Python biblioteka za sokete

kreira UDP soket za server

učitavanje unosa sa tastature

dodavanje imena server i destinacinog porta u poruku koja se šalje soketom

učitavanje modifikovanih karaktera u string

štampa primljeni string i zatvara soket

```
import socket
serverName = 'alias ili IP adresa servera'
serverPort = 12000
clientSocket = socket.socket(socket.AF_INET,
                             socket.SOCK_DGRAM)
message = input('Unesite rečenicu malim slovima:').encode()
clientSocket.sendto(message,(serverName, serverPort))
modifiedMessage, serverAddress =
                             clientSocket.recvfrom(2048)
print(modifiedMessage.decode())
clientSocket.close()
```

Primjer aplikacije: UDP server

Python UDPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print("Server je spreman za prijem podataka")
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

kreira UDP soket

pridružuje soket lokalnom broju porta 12000

beskonačna petlja

učitava podatke iz UDP soketa u varijablu *message* i eskraktuje klijentovu IP adresu i broj porta

Šalje string sa velikim slovima klijentu

Programiranje TCP soketa

klijent mora kontaktirati servera prije slanja podataka

•server mora biti već pokrenut

•server mora imati soket za prihvatanje klijentskih konekcija

klijent se povezuje sa serverom:

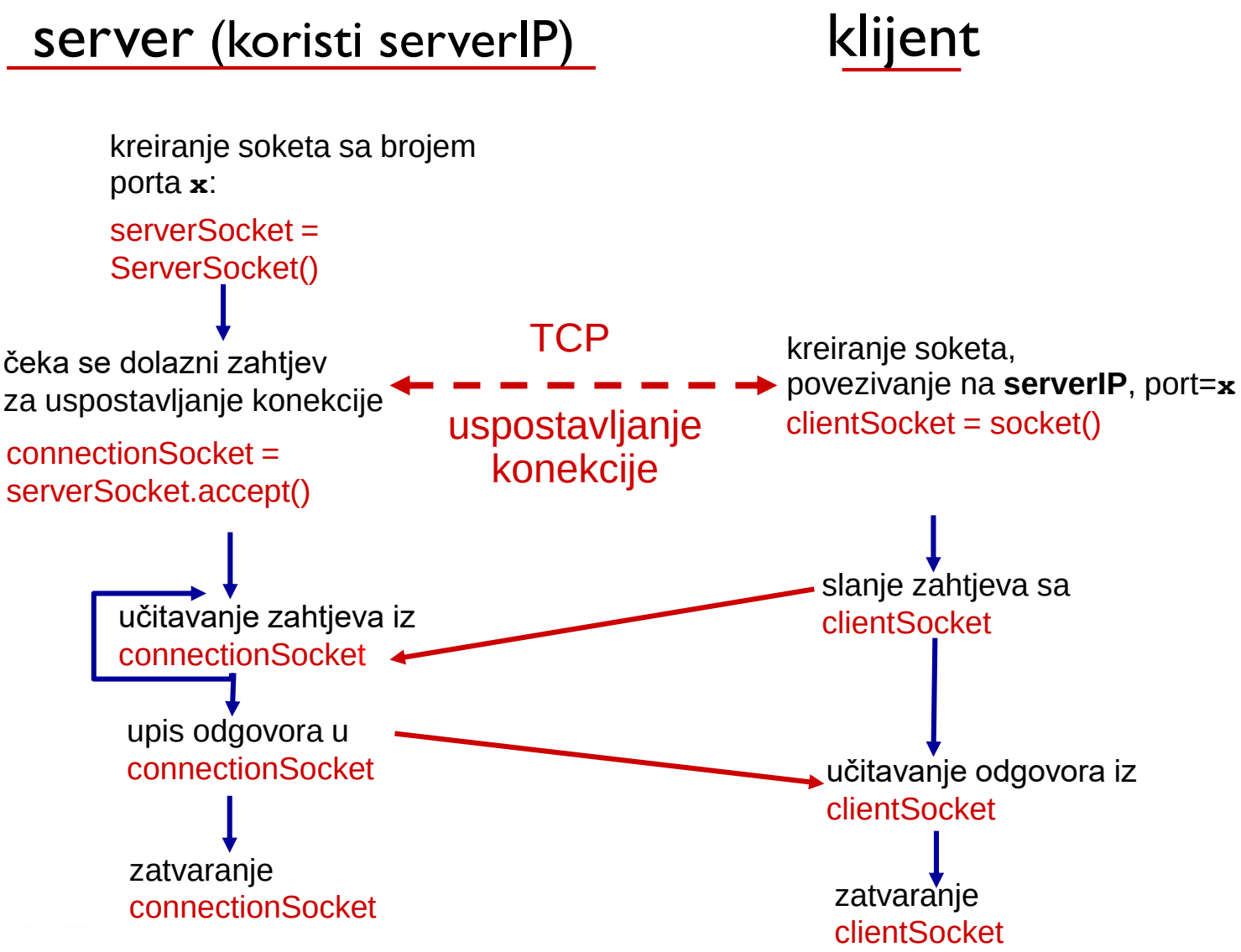
•kreiranjem TCP soketa, navodeći IP adresu i broj porta serverskog procesa

server prihvata konekciju:

- kreiranjem novog za konekciju rezervisanog soketa
- server može simultatno komunicirati sa više klijenata

Sa aspekta aplikacije:
TCP pruža pouzdan, redosledan prenos toka bajtova ("*pipe*") između klijenta i servera

Klijent-server interakcija: TCP



Primjer aplikacije: TCP klijent

Python TCPClient

```
import socket
serverName = ‘alias ili IP adresa servera’
serverPort = 12000
clientSocket = socket.socket(socket.AF_INET,
                             socket.SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
message = input(‘Unesite rečenicu malim slovima:’).encode()
clientSocket.send(message)
modifiedMessage = clientSocket.recv(1024)
print(“Od servera:”, modifiedMessage.decode())
clientSocket.close()
```

kreirati TCP soket za serverIP, udaljeni port 12000

nema potrebe za navođenjem IP adrese servera i destinacionag broja porta

Primjer aplikacije: TCP server

Python TCPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((‘’,serverPort))
serverSocket.listen(1)
print(‘Server je spreman za prijem podataka’)
while 1:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024)

    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

kreiranje TCP soketa za prijem zahtjeva

server osluškuje dolazne TCP zahtjeve

beskonačna petlja

server čeka na accept(), kreira se novi soket preko kojeg prihvata poruku

učitavanje bajtova iz soketa (ali ne i adresa kao kod UDP-a)

zatvaranje konekcije ali ne i soketa za prijem TCP zahtjeva

Dodatna dokumentacija

- http://python.org/
  - dokumentacija, tutorijali ...
- Knjige:
  - *Learning Python*, Mark Lutz
  - *Python Essential Reference*, David Beazley
  - *Python Cookbook*, Martelli, Ravenscroft and Ascher
  - (online http://code.activestate.com/recipes/langs/python/)
  - http://wiki.python.org/moin/PythonBooks