Osnove klijent-server komunikacije (u Javi)

Klijent-server model

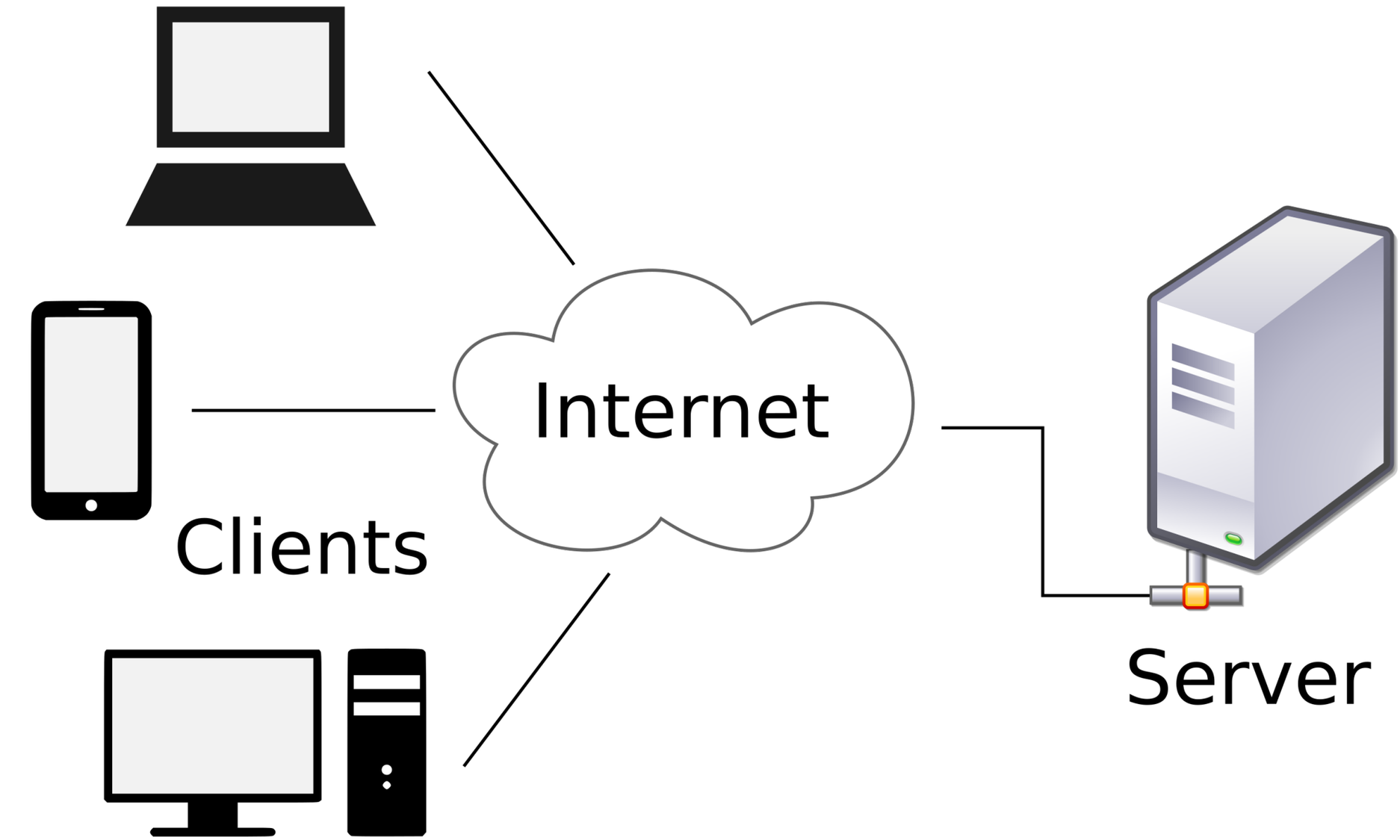
- Predstavlja jedan model arhitekture softverskog sistema koja se koristi u distribuiranim sistemima

- Različite aktivnosti pri izvršavanju distribuiranog programa se obavljaju u različitim komponentama distribuiranog sistema

- Server čini dostupnim preko mreže određene resurse ili servise

- Klijent ostvaruje vezu i koristi servise ili resurse koje server nudi

Klijent / server model



Server

- Iako se naziv koristi i za sam hardver, obično se pri pomenu servera misli na **aplikaciju** koja se izvršava na određenom **hostu** i dostupna je klijentima preko mreže

- Tipično može da opsluži **veliki broj klijenata** istovremeno

- Postoje različite implementacije (web server, file server, mail server, streaming server), u zavisnosti od toga kakve resurse čini dostupnim

- Određeni računar može biti samo server, ali može biti istovremeno i klijent za neki drugi servis koji nudi neki drugi server

Klijent

- Predstavlja aplikaciju koja se povezuje na određenu serversku aplikaciju i koristi resurse koje taj server čini dostupnim

- Klijent obično inicira komunikaciju tako što šalje zahtev serveru

- Format zahteva je definisan prirodom aplikacija i **protokolom** koji se koristi za komunikaciju

- Web aplikacije koriste **HTTP(S)** protokol i poruke moraju biti u skladu sa tim standardom

Komunikacija između klijenta i servera

- Server mora biti startovan i spreman da prihvati dolazne konekcije

- Klijent mora znati kako da se poveže na server.
  Klijent i server komunikaciju obsavljaju preko mreže, samim tim klijent mora znati mrežnu adresu servera.

- U slučaju Http-a klijent za uspostavljanje konekcije ka željenom serveru mora znati url servera i port na kome serverska aplikacija “sluša”

- Po pokretanju server tipično upada u beskonačnu tzv. **serversku petlju**, u kojoj se čeka dolazna konekcija

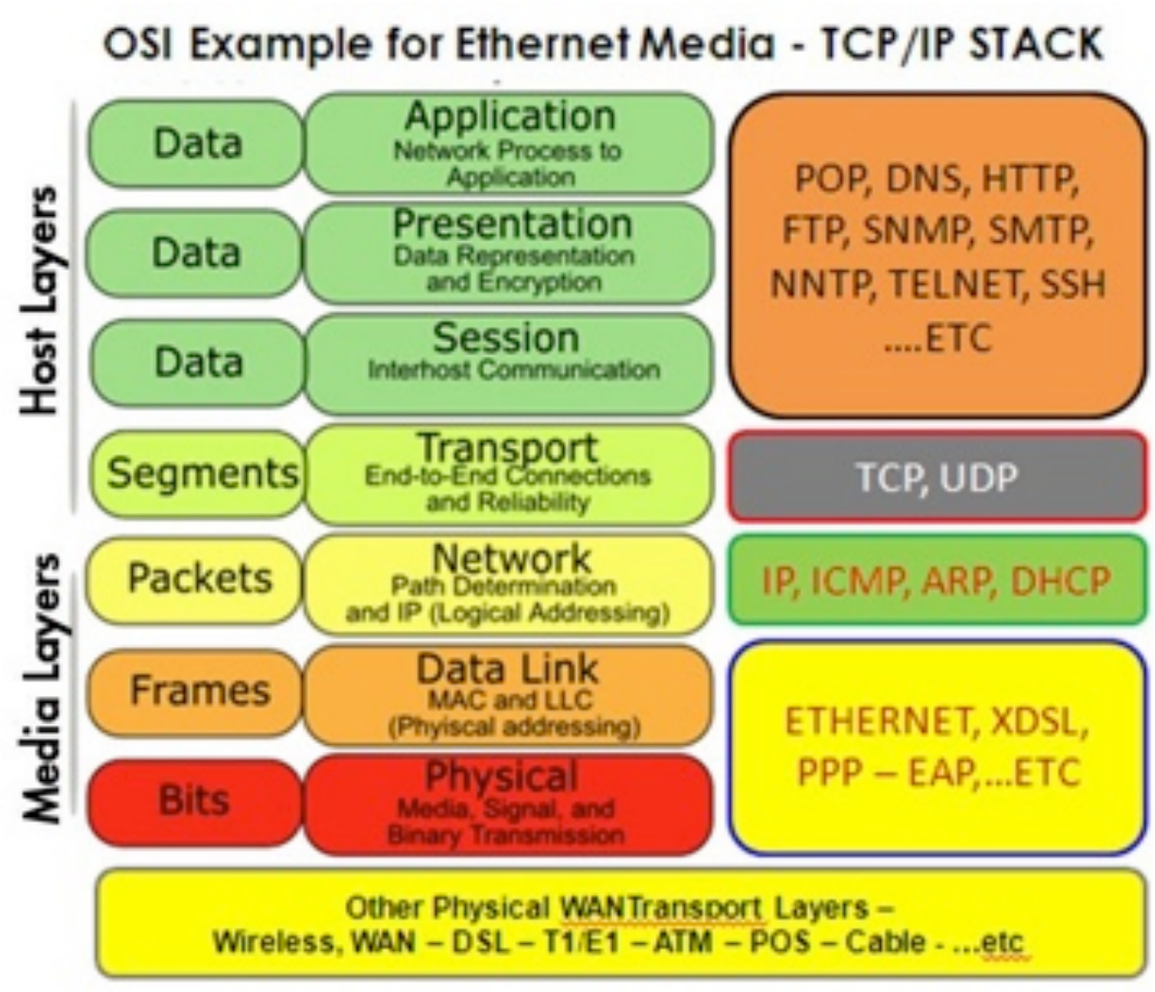
Osnove mrežne komunikacije

- Kada god se obavlja komunikacija preko mreže, neophodno je imati dogovorene sve tehničke detalje komunikacije (na najnižem nivou sam oblik signala koji se šalju, na višim nivoima kako ti signali formiraju poruke, kako se poruke formatiraju…)

- Ovi detalji su specificirani protokolima - da njih nema komunikacija bi bila nemoguća jer se niko međusobno ne bi razumeo

Osnove mrežne komunikacije - OSI nivoi

- Na različitom nivou komunikacije koriste se različiti protokoli



Osnove mrežne komunikacije

- Kada računari komuniciraju preko Interneta one to čine oslanjajući se na TCP (Transmission Control Protocol) ili UDP (User Datagram Protocol)

- Koji od ova dva protokola se koristi od strane protokola viših nivoa zavisi od prirode aplikacija i od toga šta one očekuju od mrežne komunikacije

Osnove mrežne komunikacije - TCP

- TCP obezbeđuje tačka-do-tačke (point-to-point) komunikacioni kanal za aplikacije koje zahtevaju pouzdanu komunikaciju (komunikaciju kod koji garantuju da su poslani podaci isporučeni primaoci ili će se prijaviti greška).
  Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), i Telnet su sve aplikativni protokoli koji zahtevaju pouzdan komunikacioni kanal - koriste TCP. TCP se zasniva na uspostavljanju (logičke) veze između dve aplikacije (slično kao uspostavljanje telefonske veze - preko koje se zatim razmenjuju podaci).
  Redosled kojim se isporučuju paketi je bitan za funkcionisanje aplikacija.

Osnove mrežne komunikacije - UDP

- UDP protokol omogućava komunikaciju koja ne garantuje pouzdanost.
UDP nije baziran na principu uspostavljanja konekcije. Ovde se šalju nezavisni paketi podataka (datagrami) od jedne ka drugoj aplikaciji (slično kao kod pošte - redosled isporuke nije garantovan, a svaka pošiljka je nezavisna od svih ostalih).

Osnove mrežne komunikacije - portovi

- U nekom opštem slučaju računari na mreži imaju samo jednu mrežnu konekciju

- Svi podaci koji su poslani ka aplikacijama na tom računaru pristižu kroz tu jednu konekciju

- Da bi računar znao koja aplikacija (od svih koje su na njemu pokrenute) treba da primi određeni podatak koristi se koncept porta

Osnove mrežne komunikacije - portovi

- Analogija - vaš računar je kao zgrada (ili jedan ulaz u većoj zgradi)

  - cela zgrada (ili ulaz) ima samo jednu adresu - vaš računar ima (najčešće) samo jednu mrežnu adresu

  - u zgradi u različitim stanovima žive različiti ljudi - mrežna aplikacija na vašem računaru za sebe vezuje broj socket-a na kome “sluša” mrežnu komunikaciju.
    Na ovaj način svi paketi koji u adresi sadrže navedeni port biće automatski prosleđeni datoj aplikaciji baš kao što pošta stiže samo vama u sanduče.

Osnove mrežne komunikacije - u Javi

- Danas se web aplikacije pišu po pravilu korišćenjem raznih *framework-a,* ali ipak je dobro znati šta se dešava u osnovi svega

- Kada se pišu Java programi koji treba da omoguće komunikaciju preko mreže, tipično se ne bavimo detaljima na nižim slojevima (Transportni i niži)

- Osnovne stvari koje su neophodne za mrežnu komunikaciju imamo obezbeđene u java.net paketu. Ove klase obezbeđuju sistemski nezavisan sloj za mrežnu komunikaciju

- Kada pišemo mrežnu aplikaciju u Javi mi najčešće pišemo kod koji rešava samo kako naša aplikacija obrađuje razmenu podataka preko mreže -> naš kod najčešće rešava komunikaciju na aplikativnom sloju, komunikacija na nižim slojevima je već rešena!

java.net

- Klase iz paketa java.net obezbeđuju podršku za korišćenje TCP ili UDP za komuniciranje vašeg programa preko Interneta.

- Klase **URL**, **URLConnection**, **Socket**, i **ServerSocket** koriste TCP za mrežnu komunikaciju.

- Klase **DatagramPacket**, **DatagramSocket**, i **MulticastSocket** koriste UDP.

Korišćenje URL-ova

- URL (Uniform Resource Locator) - je jedinstvena adresa nekog resursa na Internetu
- Java programi koji ostvaruju komunikaciju preko Interneta koriste **java.net.URL** klasu za predstavljanje URL adresa.
    - kreira se URL reprezentacija,
    - parsira se adresa (na ovaj način može se izvući informacija o protokolu, hostu, lokaciji resursa na serveru…),
    - otvara konekcija
    - otvaraju streamovi za čitanje i pisanje

Korišćenje URL-ova

- Primer sa https://docs.oracle.com/javase/tutorial/networking/urls/readingWriting.html

```
import java.net.*;
import java.io.*;

public class URLConnectionReader {
    public static void main(String[] args) throws Exception {
        URL oracle = new URL("http://www.oracle.com/");
        URLConnection yc = oracle.openConnection();
        BufferedReader in = new BufferedReader(new InputStreamReader(
                                yc.getInputStream()));
        String inputLine;
        while ((inputLine = in.readLine()) != null)
            System.out.println(inputLine);
        in.close();
    }
}
```

Socket-i

- Korišćenje URL i URLConnectiona obezbeđuje podršku za komunikaciju na relativno visokom nivou apstrakcije same komunikacije

- Ostvarivanje komunikacije korišćenjem nešto nižim nivoima apstrakcije ostvaruje se putem **Socket**-a

Socket-i

- Socketi predstavljaju softversku tačku prtistupa (***endpoint***) putem koje se uspostavlja dvosmerna (***bidirekciona***) komunikacija između servera i klijenata

- Obezbeđuju mehanizam za komunikaciju između dva računarska sistema preko TCPa

- Socket povezuje vaš program na određeni port na mašini (hostu) na kom je pokrenut
  - Kaže se da program „sluša“ mrežnu komunikaciju na određenom portu
  - Svaki klijent može da se poveže sa vašim serverskim programom ako zna adresu hosta i port na kome program čeka konekcije

Jednostavni klijent-server u Javi

Serverska aplikacija

- Serverski program kreira serverski socket (**ServerSocket**)
  - on obezbeđuje mehanizam kojim se serverski program povezuje na određeni **port**, čeka dolazne konekcije od strane klijenta na tom portu i uspostavlja konekcije sa klijentom

- Kada je ServerSocket kreiran serverski program poziva njegovu ***accept*** metodu kojom ulazi u **stanje čekanja** dok se ne pojavi dolazna konekcija

Jednostavni klijent-server u Javi

Klijentska aplikacija

- Klijentski program kreira svoj klijentski socket (**Socket**) tako što specificira adresu servera i port na koji želi da se poveže.

- Koristeći ovaj Socket klijentski program pokušava da se „poveže“ sa socketom na serveru.

Socket komunikacija u Javi

- Klijentski program kreira svoj klijentski socket (Socket) tako što specificira adresu servera i port na koji želi da se poveže.

- Koristeći ovaj Socket klijentski program pokušava da se „poveže“ sa socketom na serveru.

- Kada se konekcija uspostavi, **accept** metod vraća referencu na novi Socket objekat, koji je povezan sa klijentskim Socketom i opslužuje komunikaciju sa tim klijentom.
- Sada klijent i server mogu razmenjivati poruke tako što čitaju i pišu iz/na ulazno izlazne streamove koji su kreirani na socketima

Opsluživanje više klijenata

- Ako bi server mogao da opsluži samo jednu konekciju u svojoj serverskoj petlji (čitanje i pisanje samo preko jedne konekcije) ne bi bio od velike koristi

- Server bi trebalo da bude u stanju da opsluži više klijenata istovremeno

- Tipično se serverska aplikacija realizuje kao:
  - multiprocesna
  - multithreaded ili
  - hibridna aplikacija

- U svakoj od ovih implementacija server po uspostavljanju konekcije predaje obradu zahteva nekom ***workeru,*** dok se sam glavni serverski proces u petlji vraća da **čeka** sledeću konekciju

Primer klijent-servera u Javi

- kod se nalazi u SimpleClientServer projektu