LINKgroup

Distance Learning System

Advanced Java

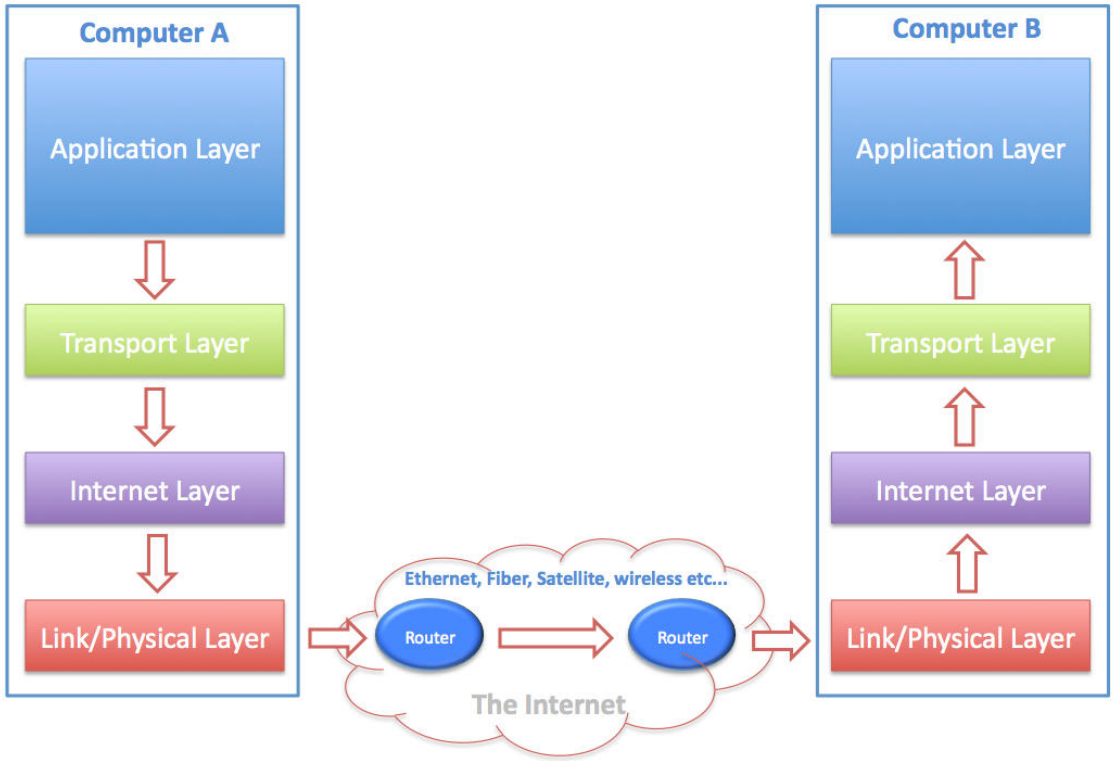
Tokovi podataka i rad sa mrežom

Internet mrežni protokoli

- Mrežni protokol je set pravila koji omogućava komunikaciju između dva povezana računara
- U računarskoj mreži, komunikacija između računara podrazumeva nekoliko vertikalnih slojeva
- Količina ovih slojeva zavisi od koncepta raslojavanja, ali obično se uzima u obzir jedan od dva modela: OSI model ili Internet model
- **OSI (Open System Interconnection)** je podela na sedam slojeva (Physical, Data Link, Network, Transport, Session, Presentation i Application).
- **Internet (tcp/ip)** podrazumeva četiri sloja. Link, Internet, Transport i Application (Ovo je ujedno i model koji ćemo obrađivati)

Internet model komunikacije

- **Link** je najniži sloj u internet komunikaciji. Ovaj sloj podrazumeva modulaciju i demodulaciju signala, fizičku konekciju i slično.
- U **Internet (IP)** sloju se obrađuju paketi, proveravaju greške, vrši kompresija i dekompresija...
- U **Transport** sloju se vrši konekcija između krajnjih tačaka komunikacije (End Points), šalju i primaju podaci i proverava njihov integritet. Ovo predstavlja sistem po kome će funkcionisati dve strane koje komuniciraju (za nas su najvažniji TCP i UDP transportni protokoli).
- U **Application** sloju, aplikacije razmenjuju informacije među sobom, na osnovu tehnologija na kojima počivaju. Način na koji će se informacije tretirati predstavljen je odgovarajućim protokolom (HTTP, FTP, SMTP...)

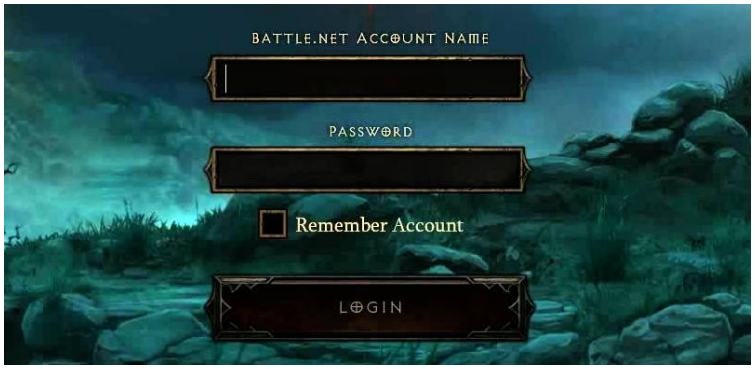


Transportni protokoli u tcp/ip modelu

- Postoje tri bitna transportno-komunikaciona protokola u razmeni informacija između računara putem interneta.
- To su:
  - **Transmission Control Protocol (TCP)**
  - **User Datagram Protocol (UDP)**
  - **Internet Control Message Protocol (ICMP)**.

TCP Protokol

- Kada se ostvari konekcija između pošiljaoca i primaoca u TCP protokolu, ona biva održavana sve do trenutka dok paket ne bude prihvaćen. To omogućava dvosmernu komunikaciju između korespondenata (računara)
- Njegove karakteristike su:
  - Visok nivo pouzdanosti i zato njega srećemo u većini internet protokola višeg nivoa (FTP, HTTP, SMTP...).
  - Slabija brzina (nije pogodan za aplikacije koje zahtevaju veliku brzinu komunikacije (realno vreme)



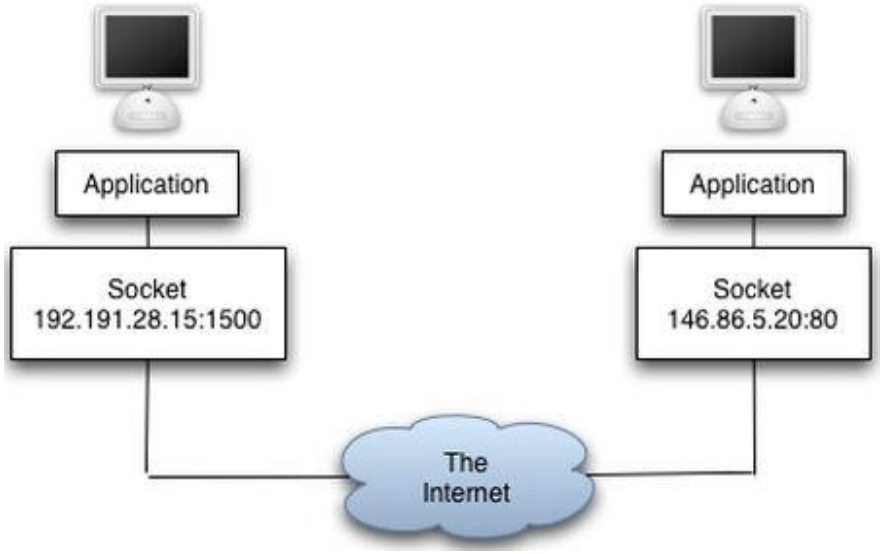
UDP Protokol

- UDP ne zahteva perzistentnu konekciju između krajnjih tačaka komunikacije

- Njegove karakteristike su:
  - Brzina i efikasnost
  - Slab integritet podataka

Socket
(http://beej.us/guide/bgnet/)

- Socket predstavlja jedan kraj u komunikaciji između dva procesa. Određen je IP adresom, koja identifikuje host, i brojem porta koji identifikuje proces koji se izvršava na mrežnom čvoru
- Postoji nekoliko tipova Socketa u zavisnosti od tipa konekcije:
  - ***Datagram socket***, socketi koji ne zahtevaju otvorenu konekciju, koriste se sa UDP protokolom
  - ***Stream socket***, socketi za rad u konektovanom okruženju sa TCP protokolima
  - ***Raw socket***, uglavnom implementiran u ruterima i drugoj mrežnoj opremi



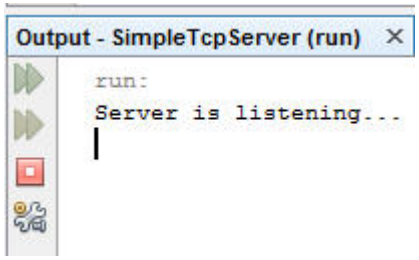
TCP Socket programiranje

- Socket predstavlja IP adresu i port jednog sagovornika u konekciji. On počinje da postoji onog trenutka kada se konekcija ostvari i prekida svoje postojanje nakon što se konekcija prekine. Pri tome je, kod TCP protokola, ova konekcija je **dvosmerna** i **sinhronizovana**.
- Da bi postojala neka konekcija, potrebne su dve strane: **server** i **klijent**.
  - **Server** ima ulogu slušaoca. On sluša sve aktivnosti na zadatom portu. Nakon što se konekcija dogodi, server inicijalizuje Socket i izvršava definisane aktivnosti. Dakle, pasivan je sve dok klijent ne inicira promenu njegovog statusa.
  - **Klijent** je taj koji inicira komunikaciju sa serverom. Nakon što prosledi svoj zahtev, on očekuje od servera odgovor čime se uspostavlja uzajamna razmena podataka. Sve dok je klijent „nakačen" na server, razmena podataka traje.
- Razmena podataka biva prekinuta kada klijent prekine komunikaciju sa serverom.
- Java ima već ugrađen sistem za rukovanje Socketima. On se, kao i ostale klase vezane za mrežu, nalazi u paketu **java.net**. *(https://docs.oracle.com/javase/7/docs/api/java/net/package-summary.html)*

Kreiranje TCP slušača (Klasa ServerSocket)
(jaex032014 SimpleTcpServer)

- Iako je u Javi moguće raditi direktno na nivou soketa (pomoću klase Socket), ona sadrži i specijalizovane klase za kreiranje serverskog i klijentskog socket-a.
- Za kreiranje serverskog tcp socket-a, može se koristiti klasa **ServerSocket**

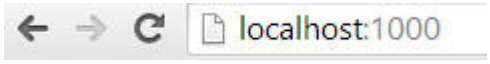
```
ServerSocket serverSocket = new ServerSocket(1000);
System.out.println("Server is listening...");
serverSocket.accept();
System.out.println("Connection accepted");
```



- Za razliku od prethodnih programa, ovaj program će blokirati konzolu
- Program će nastaviti sa radom tek nakon što se neko konektuje na socket:
- Na primer, kucanjem sledećeg u konzoli

```
ab -n1 -c1 http://localhost:1000/
```

- Ili unosom sledeće adrese u browser



- Konačan izlaz programa

```
run:
Server is listening...
Connection accepted
BUILD SUCCESSFUL (total time: 1 minute 2 seconds)
```

Preuzimanje socket-a iz konekcije
(jaex032014 SimpleHttpServer)

- O jednom nakačenom klijentu, možemo saznati dosta na osnovu podataka iz socket-a
- Ono što nas najčešće interesuje, jeste njegova ip adresa. Ovaj podatak (kao i mnoge druge), obezbeđuje klasa **Socket**, čiju instancu kreira metod **accept**.

```
ServerSocket server = new ServerSocket(1000);
System.out.println("Server is listening...");

Socket socket = server.accept();
StringBuilder cOut = new StringBuilder();
    cOut.append("Remote address: " + socket.getRemoteSocketAddress()+"\n");
    cOut.append("Local address: " + socket.getLocalAddress()+"\n");
    cOut.append("Local port: " + socket.getLocalPort()+"\n");
System.out.println(cOut);
```

Preuzimanje sadržaja od klijenta
(jaex032014 SimpleHttpServer)

- Komunikacija između klijenta i servera se vrši pomoću toka
- Ovaj tok dostupan je kroz metod **getInputStream**
- Sledeći kod preuzima ulazni tok od socket-a, čita ga i ispisuje na izlaz:

```
InputStreamReader sreader = new InputStreamReader(stream);
BufferedReader rdr = new BufferedReader(sreader);
while(!(line=rdr.readLine()).isEmpty()){
    System.out.println(line);
}
```

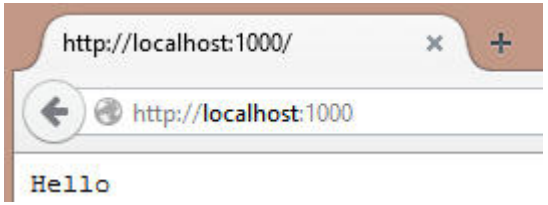
Izlaz

```
GET / HTTP/1.0
Host: 127.0.0.1:1000
User-Agent: ApacheBench/2.3
Accept: */*
```

Slanje sadržaja klijentu
(jaex032014 SimpleHttpServer)

- Kada je klijent „rekao" serveru šta je imao, server mora da mu odgovori, kako bi komunikacija bila po http standardu:

```
BufferedWriter wr = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
wr.write("HTTP/1.1 200 OK\r\n\r\n");
wr.write("Hello");
wr.flush();
socket.close();
```



Vežba 1
(jaex032014 SimpleWebServer)

- Potrebno je kreirati jednostavan web server
- Web server prilikom startovanja čita konfiguracioni fajl u kome se nalazi port na kome će slušati i direktorijum u kome će se nalaziti html strane

Datagram (UDP Socket)

https://docs.oracle.com/javase/tutorial/networking/datagrams/

- UDP komunikacija podrazumeva slanje poruka bez korišćenja toka i direktne konekcije
- Posledica toga su paketi koji nisu sigurni u smislu konzistentnosti
- Ne postoji garancija isporuke
- Ne postoji automatska provera isporuke

DatagramSocket
(jaex032014 SimpleUdp)

- Za rukovanje UDP soketom, zadužena je klasa **java.net.DatagramSocket**
- Jedan UDP paket, predstavlja se pomoću klase **DatagramPacket**
- Klasa DatagramSocket ima mogućnost slušanja

```
DatagramSocket udpServer = new DatagramSocket(1000);
byte[] data = new byte[32];
DatagramPacket dPacket = new DatagramPacket(data, data.length);
System.out.println("Server is listening...");
while(true){
    udpServer.receive(dPacket);
    byte[] recData = dPacket.getData();
    System.out.println(new String(recData,"UTF-8"));
    dPacket.setData(new byte[32]);
}
```

DatagramSocket
(jaex032014 SimpleUdp)

- Da bi poslali poruku preko udp porta, koristimo istu tehnologiju kao i za slušanje, samo što umesto metode receive, koristimo metod **send**.
- Ovaj sistem je odličan za brzu jednosmernu komunikaciju (na primer, slanje pozicije na mapi, slanje strima, slanje statusa neke mašine...)

```
InetAddress host = InetAddress.getByName("localhost");
DatagramSocket dSocket = new DatagramSocket();
byte[] buffer = "Hello".getBytes();
DatagramPacket message = new DatagramPacket(buffer, buffer.length,host,1000);
dSocket.send(message);
```

Dvosmerna UDP komunikacija
(jaex032014 Udp2Way)

- Za dvosmernu udp komunikaciju treba koristiti istu tehnologiju i na klijentu i na serveru

Zadatak

- Pokušajte da modifikujete igru iz prvog kursa (kamen, papir, makaze), tako da bude za dva igrača povezana preko mreže

Rukovanje elektronskom poštom (Java mail API)

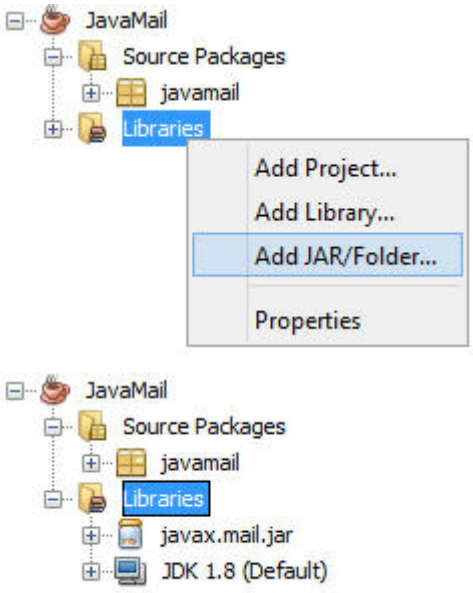
https://java.net/projects/javamail/pages/Home

- JavaMail nije deo standardne Java SE distribucije, već se mora posebno preuzeti sa gore navedene adrese

Download JavaMail 1.5.2 Release

The following table provides easy access to the latest release. Most people will only need the main JavaMail reference im

Item	Description
javax.mail.jar	The JavaMail reference implementation, including the SMTP, IMAP, and POP3 protocol providers
README.txt	Overview of the release
NOTES.txt	Additional notes about using JavaMail
SSLNOTES.txt	Notes on using SSL/TLS with JavaMail
NTLMNOTES.txt	Notes on using NTLM authentication with JavaMail
CHANGES.txt	Changes since the previous release
COMPAT.txt	Important notes about compatibility



Korišćenje sigurnog smtp servera (jaex032014 SimpleMailClient)

- Slanje standardnog maila preko sigurnog smtp servera:

```
Properties props = new Properties();
props.setProperty("mail.smtp.host", "smtp.gmail.com");
props.setProperty("mail.smtp.socketFactory.port", "465");
props.setProperty("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
props.setProperty("mail.smtp.auth", "true");
props.setProperty("mail.smtp.port", "465");
Session session = Session.getInstance(props, new Authenticator() {
        protected PasswordAuthentication getPasswordAuthentication(){
            return new PasswordAuthentication("myemail@google.com", "mypassword");
        }
});
Message message = new MimeMessage(session);
message.setFrom(new InternetAddress("myemail@google.com"));
message.setRecipients(Message.RecipientType.TO, InternetAddress.parse("targetemail@targethost.com"));
message.setSubject("Test Java Mail");
message.setText("Hello from Java!");
Transport.send(message);
```

Upravljanje mailbox-om pomoću imap protokola (jaex032014 ImapMailClient)

```
Properties props = new Properties();
props.setProperty("mail.store.protocol", "imaps");
Session session = Session.getInstance(props);
Store store = session.getStore();
store.connect("imap.gmail.com", "mygmailaddress@gmail.com", "mygmailpassword");
Folder inbox = store.getFolder("INBOX");
inbox.open(Folder.READ_ONLY);
int lastmsg = inbox.getMessageCount();
while(lastmsg>0){
    Message msg = inbox.getMessage(lastmsg--);
    System.out.println("From: " + msg.getFrom()[0]);
    System.out.println("Subject: " + msg.getSubject());
    System.out.println("Body: " + msg.getContent());
}
```

Zadatak

- Pokušajte da kreirate jednostavan mail klijent koji uzima od korisnika adresu **primaoca**, zatim **subject** i **poruku**, a zatim to šalje kao mail sa neke fiksne adrese

```
To: sally.jones@sallysma.il
Subject: Hello!!!
Message: Hello Sally, how are you today?
Sending mail...
Mail sent successfully
```