УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

Трг Доситеја Обрадовића 6, 21000 Нови Сад, Југославија
Деканат: 021 350-413; 021 450-810; Централа: 021 350-122
Рачуноводство: 021 58-220; Студентска служба: 021 350-763
Телефакс: 021 58-133; e-mail: ftndean@uns.ns.ac.yu



**Geodezija i Geomatika**

# SEMINARSKI RAD

**Predmet:**

**Integrisani IT sistemi premera**

## Tema:

## TCP – Slanje i prijem poruka

**Profesor:** Dr. Vladimir Bulatović

**Student:** Milan Veselinović GG 68-2011

# Sadržaj

# 1.UVOD

TCP/IP je set protokola razvijen da omogući umreženim računarima da dele resurse putem mreže. Nastaje iz eksperimentalne „packet-switching“ mreže agencije ARPA, Advanced Research Projects Agency, nazvane ARPAnet, napravljene radi proučavanja tehnika slanja robusnih i pouzdanih podataka. ARPAnet postaje veoma uspešna mreža, toliko da su mnogobrojne organizacije povezane na nju počele da je koriste u svakodnevnoj komunikaciji.

Mreža iz eksperimentalne postaje operativna 1975. godine i tada kontrolu admnistracije mreže preuzima DCA, Defense Communications Agency (kasnije DISA), odnosno Američko ministarstvo odbrane. Posle toga TCP/IP postaje vojni standard i svi „hostovi“ spojeni na mrežu morali su preći na novi protokol. Daljim razvojem mreže u ono što danas znamo kao Internet, TCP/IP ostaje jedan od najrasprostranjenijih protokola za komunikaciju u mreži i između mreža.

Komunikacija u mreži i između njih se obavlja preko čvorova, TCP/IP hostova, gde je svakom čvoru u mreži dodeljena jedinstvena IP adresa. TCP/IP gateway čvorovi povezuju jedan tip mreže sa drugim tipom mreže, pomoću hardvera i softvera za konverziju sa iz jednog formata u drugi.

# 2. OSI REFERENTNI MODEL

ISO ( International Standards Organization ) razvija arhitektonski model koji je konstantno korišćen za opisivanje strukture i funkcije podatkovnih komunikacijskih protokola. Ovaj model, nazvan Open System Interconnect Reference Model ( OSI ), predstavlja osnovnu referencu za promatranje komunikacija. Nastaje 1979. godine, a pet godina kasnije izvršena je njegova revizija.

OSI referentni model sadrži sedam slojeva (leyers) koji definišu funkcije komunikacijskih protokola. Raspoređeni su od najnižeg koji opisuje fizičke i električne osobine mreže, pa do najvišeg gde se nalazi korisnik podataka koji se prenose.
OSI model:

7. Sloj aplikacije ( Application ) – mrežne primene poput SMTP, HTTP, FTP

6. Sloj prezentacije (Presentation) – formatiranje podataka i zaštita

5. Sloj sesije ( Session ) – uspostavljanje i održavanje sesija

4. Sloj transporta ( Transport ) – osiguranje prenosa s kraja na kraj

3. Sloj mreže (Network ) – isporuka jedinica informacije, uključujući usmeravanje (routing)

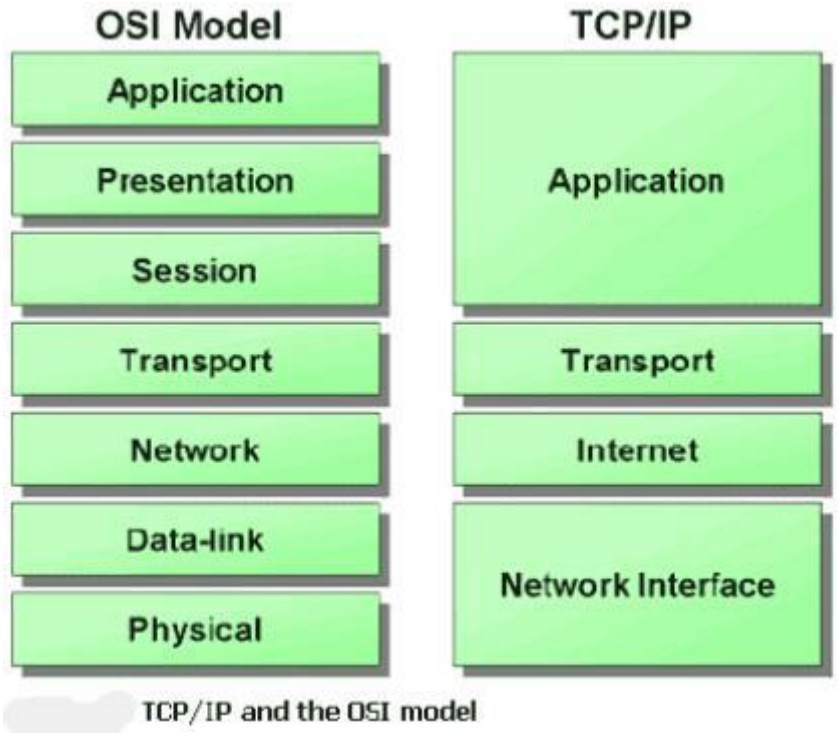
2. Sloj veze ( Data Link ) – prenos jedinica informacije s proverom greške

1. Fizički sloj (Physical) – prenos binarnih podataka kroz medij

Ovi slojevi su uopšteno gledano podeljeni u dve grupe, odnosno na slojeve višeg nivoa, koji čine sloj aplikacije, prezentacije i sesije, i slojevi nižeg nivoa, odnosno preostala četiri sloja. Viši nivo je bliži krajnjem korisniku, gde je najbliži aplikacijski sloj, dok niži nivo definiše kako se prenose podaci sa jednog na drugi računar.

## 2.1. Arhitektura TCP/IP protokola

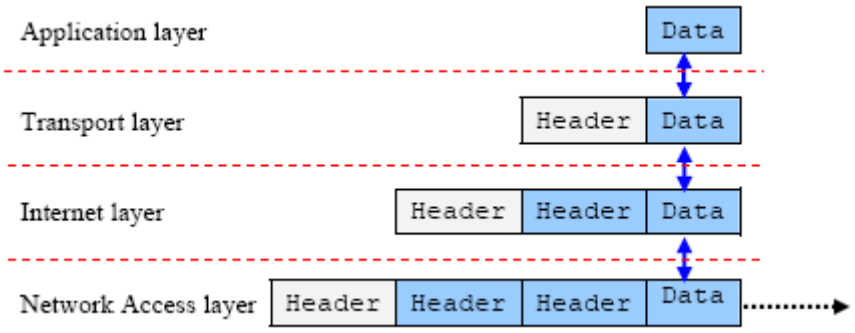
TCP/IP protokol je sastavljen od manjeg broja slojeva od onih korištenih u OSI modelu. Mnogi opisi TCP/IP definišu sa od tri do pet funkcionalnih slojeva arhitekture protokola. Slika 2.1 pokazuje četvoroslojni TCP/IP model u korelaciji sa OSI modelom. TCP/IP model je baziran na tri sloja ( aplikacijski, prenosni i mrežni ) i dodatnim Internet slojem.




Slika 2.1. OSI i TCP/IP model

Transportni sloj u ovakvom modelu predstavlja TCP segment, dok Internet predstavlja IP nivo. Podaci se u TCP/IP-u šalju niz „protocol stack“ iz aplikacijskog sloja sve do fizičke mreže. Svaki sloj u „stacku“ dodaje kontrolnu informaciju ( header ) da bi potvrdio sigurnu

dostavu, koju postavlja na početak podatka koji treba poslati. Svaki sloj tretira sve informacije koje primi iz sloja iznad podatke i stavlja svoj header na početak podatka koji prosleđuje ( slika 2.2 ). Suprotno, kada su podaci primljeni, svaki sloj skida svoje zaglavlje pre nego pošalje podatke sloju iznad.



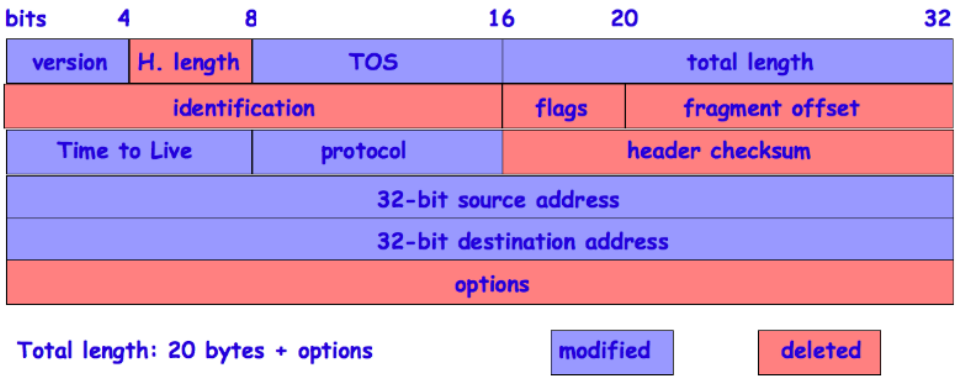

Slika 2.2. Prolaz podataka kroz slojeve

## 2.1.1. IP protokol

IP ( Internet Protocol ) se koristi za komunikaciju između računara, a za prenos ga koriste TCP i UDP. Odgovoran je za pravilno adresiranje računara i prosleđivanje paketa, ali ne garantuje njihovo dospeće. IP može razbiti podatke na manje pakete i sastaviti ih na odredištu, s time da svaki deo može ići drugim putem kroz mrežu.

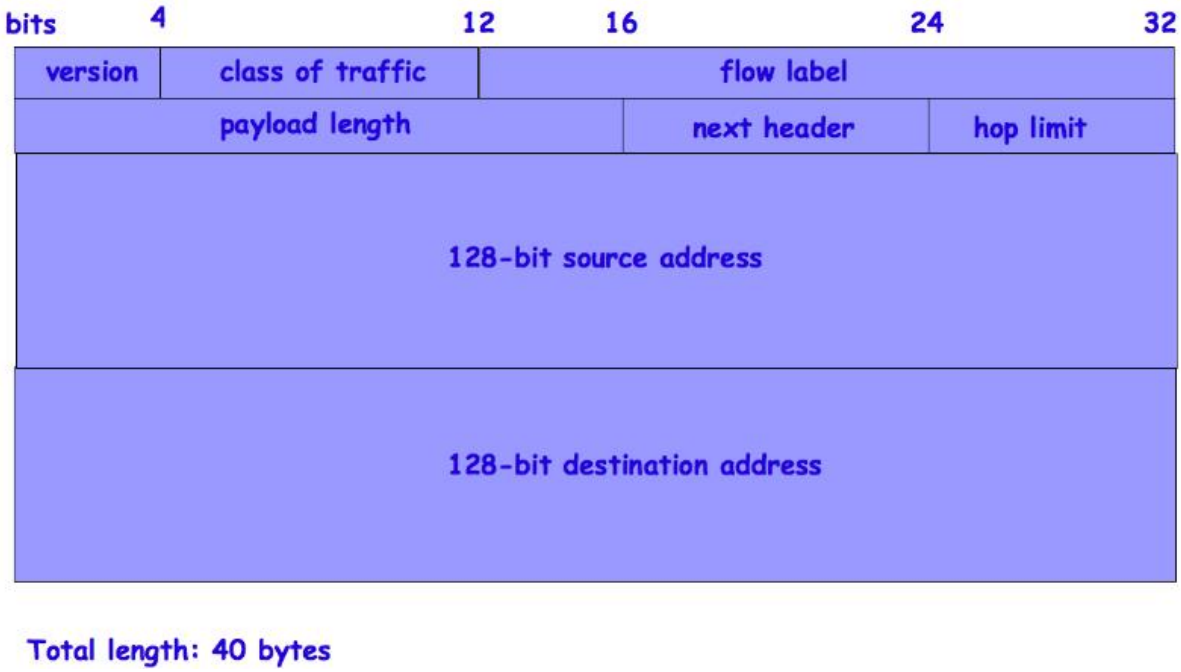
Svaki računar na Internetu ima svoju IP adresu, koja je u IPv4 standardu predstavljena u formatu x.y.z.q, gde su x,y,z,q brojevi iz skupa od 0 do 255 što ovu adresu čini 32-bitnom. Kako je slobodnih IPv4 adresa sve manje, uveden je novi standard IPv6 koji podržava rad sa 128-bitnim adresama.

IP protokol prima informaciju od transportnog sloja, dodaje joj zaglavlje (Header) i time dobija rezultantni paket koji se naziva Internet Datagram. Header sadrži informacije o izvorišnoj i odredišnoj IP adresi, verziju IP protokola i ostale podatke bitne za funkcionisanje.




Slika 2.3. IPv4 Header

Na slici 2.3 je prikazan format IPv4 headera. Kako IPv6 protokol prelazi u sve širu upotrebu, time postaje i značajniji za razumevanje. Kao što je rečeno, adrese IPv6 protokola su 128-bitne, sa heksadecimalnim zapisom u osam kolona, odvojenih dvotačkom, npr. 3ffe:1900:4545:3:200:f8ff:fe21:67cf. Slika 2.4 prikazuje format IPv6 headera.




Slika 2.4. Format IPv6 headera

- Version number (4 bita) – sadrži broj verzije protokola
- Class of traffic (8 bita) – omogućava razlikovanje paketa i definisanje prioriteta
- Flow label (20 bita) – omogućava izvoru da definiše naziv paketa istok toka
- Payload length (16 bita) – definiše obim IP datograma
- Next header (8 bita) – pokazuje koje zaglavlje sledi posle IPv6 zaglavlja
- Hop limit (8 bita) – definiše maksimalan broj skokova koe datagram može da obavi pri prolasku kroz mrežu. Svakim skokom granica se smanjuje za 1 i kad dostigne nulu, datagram se izbacuje
- Source address (128 bita) – sadrži IPv6 adresu sa koje je paket originalno poslat
- Destination address ( 128 bita) – sadrži IPv6 adresu onog kome je paket namenjen

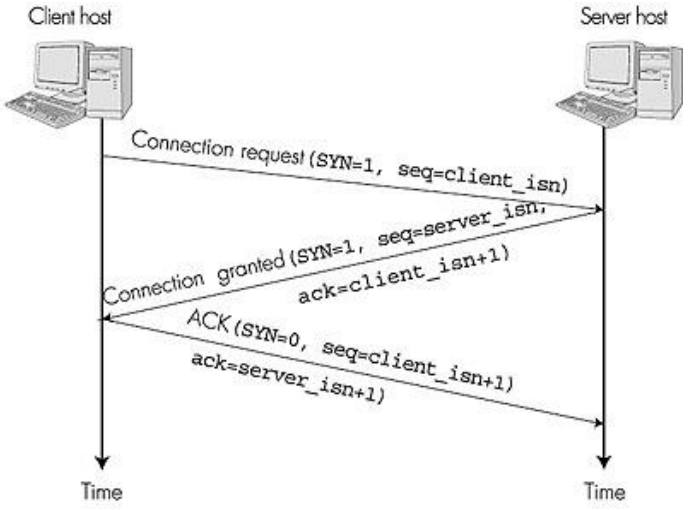
# 3. TCP PROTOKOL

Osnovni protokoli na transportnom sloju su TCP i UDP. TCP (Transmission Control Protocol) je obostrani spojni protokol koji spaja 2 računara. On je odgovoran za podelu poruka u datagrame, ponovno slanje datagrama koji nisu stigli i sastavljanje poruke na drugom kraju. Razlika između TCP i UDP protokola je što će TCP tražiti ponovno slanje paketa ukoliko se neki izgubi, te je pogodan za sigurniju razmenu podataka gde brzina nije toliko bitna. UDP protokol nema mogućnost ponovnog zahtevanja izgubljenog paketa.

Osnovna jedinica TCP protokola je segment, koji se pakuje u IP pakete i šalje preko mreže. Dozvoljava dvosmerni prenos podataka, kao i multipleksiranje, odnosno demultipleksiranje podatka. Multipleksiranje je proces u kome su paketi pristigli iz više različitih izvora, ukombinovani i poslati kao jedan tok. Demultipleksiranje je suprotan proces, gde se kombinovani tok deli na originalne tokove podataka.

## 3.1 Način rada TCP protokola

Kako bi TCP funkcionisao, neophodno je obaviti proces **uspostavljanja veze**. Uspostavljanje veze u TCP protokolu funkcioniše po klijent – server principu, odnosno neophodno je izvršiti proces na jednom računaru radi uspostavljanja veze sa drugim. Prilikom uspostavljanja veze, razmenjuju se 3 specijalna segmenta ( three way handshake), slika 3.1, kada klijent šalje serveru prvi specijalni TCP segment, na šta server odgovara drugim specijalnim segmentom, nakon čega klijent odgovara trećim specijalnim TCP segmentom.




Slika 3.1. Three way handshake

Klijent prvi šalje specijalni TCP segment serveru. Taj specijalni segment ne sadrži podatke aplikacijskog nivoa. Ima jedan od bitova zastavica u zaglavlju segmenta. To je tzv. SYN bit, postavljen na 1. Iz tog razloga, taj specijalni segment zove se SYN segment. Nadalje, klijent odabire inicijalni redni broj (client_isn) i stavlja ga u polje za redni broj inicijalnog TCP SYN segmenta. Taj segment je uhvaćen u IP datagramu i poslan na internet.

Pod pretpostavkom da IP datagram koji sadrži TCP SYN segment stigne do servera on izdvaja TCP SYN segment iz datagrama, alocira TCP spremnik i varijable i šalje segment kojim odobrava uspostavu veze klijentu. Taj segment odobravanja veze također ne sadrži podatke aplikacijskog sloja, ali sadrži tri važne informacije u zaglavlju segmenta. Prvo, SYN bit je postavljen na 1. Drugo, Acknowledgment polje zaglavlja TCP segmenta se namešta na isn+1. Na kraju, server odabire svoj inicijalni redni broj (server_isn) i stavlja vrednost u polje zaglavlja TCP segmenta.

Kada klijent primi segment odobravanja veze, takođe alocira spremnik i varijable u vezi. Klijent tada šalje server još jedan segment koji potvrđuje da je dobio segment odobravanja veze. To radi tako da stavi vrednost server_isn+1 u acknowledgement polje zaglavlja. SYN bit postavlja se u 0 budući da je veza uspostavljena.

Kada je veza uspostavljena, može se pristupiti **razmeni podataka**. TCP entiteti razmenjuju podatke u obliku segmenata. Segment se sastoji od zaglavlja koje ima 20 bajtova (uz opcionalni deo) za kojim sledi nula ili više bajtova podataka, a nastaje skupljanjem podataka od nekoliko upisivanja ili razbijanjem podataka od jednog upisivanja. Veličina segmenta je varijabilna uz dva ograničenja:

-Svaki segment uključujući i TCP zaglavlje mora stati u 65 535 bajtova IP paketa

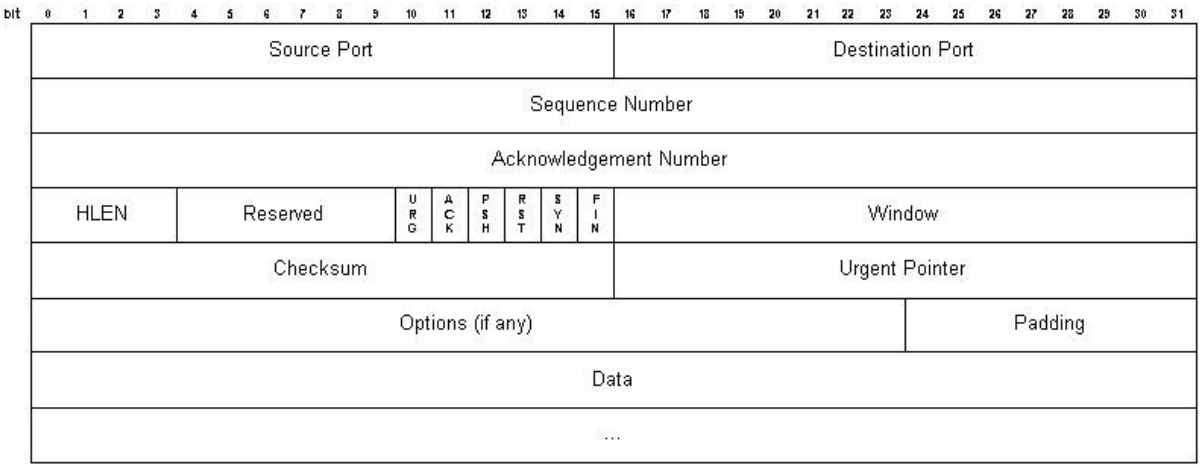
-Svaka mreža ima svoj MTU (Maximum Transmission Unit), a to je najveća dopuštena jedinica za prenos koja definiše gornju granicu veličine segmenta.

Ako je segment prevelik za mrežu kroz koju mora proći, čvor vrši fragmentaciju u više manjih segmenata od kojih svaki dobija svoje IP zaglavlje.Osnovni protokol kojeg koriste TCP entiteti je protokol s klizajućim prozorom (Sliding Window):

- Nakon slanja segmenta predajnik pokreće brojač (engl. timer)

- Kad segment stigne na odredište, prijemnik šalje u segmentu potvrdu s brojem jednakim sledećem broju segmenta kojeg očekuje

- Ako brojač istekne pre nego što je primljena potvrda, segment se šalje ponovno




Slika 3.2. TCP header

Format zaglavlja ( slika 3.2 ):

-Izvorišni i odredišni (source ,destination) port -broj 16-bitne vrednosti koji identifikuje broj izvorišne tačke, odnosno odredišne tačke.

-Redni broj (Sequence number ) –identifikuje tekući redni broj segmenta podataka. Na taj način svaki „izgubljeni“ segment je lako identificirati.

-Acknowledgament number – broj potvrde, ako je postavljen bit (tj.zastavica A (ACK) ,polje sadrži redni broj sledećeg bajta koga primatelj očekuje.

-Data offset- je 32-bitna vrednost koja identufikuje početak podataka.

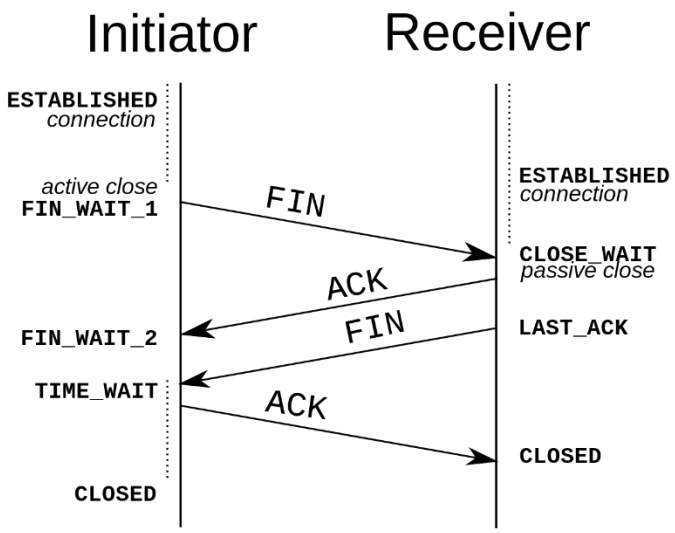
-Flags polje- je polje sa kontrolnim bitovima (flagovi ili zastavice ili markeri) koji se mogu postaviti na 1 ili na 0. Imamo šest vrsta zastavica i to :

1. U (URG) –urgentni marker
2. A (ACK)- marker potvrde
3. P (PSH) –push funkcija koja inicira prosleđivanje svih do tada ne prosleđenih podataka korisniku
4. R ( RST) –reset marker –ponovo inicijalizuje vezu
5. S ( SYN) –sihronizuje redne brojeve
6. F (FTN) – marker koji označava kraj prenosa

-Window Size –predstavlja polje koje označava koliko bajtova podataka prijemni host može da primi u datom trenutku.

-Urgent pointer –urgentni pokazivač koji pokazuje na redni broj bajta gdje se nalaze hitni podaci ( aktivan je samo ako je postavljen U marker)

Do **prekida veze** dolazi kad klijent odluči prekinuti vezu sa serverom. Procedura prekida veze je prikazana na slici 3.3:




Slika 3.3. Prekid veze

-Klijent prvo šalje TCP segment sa FIN bitom postavljenim na 1 i uđe u FIN_WAIT_1 stanje.

- Dok je u FIN_WAIT_1 stanju, klijentski TCP čeka TCP segment potvrde od strane servera.

-Kad primi navedeni segment, klijentski TCP ulazi u FIN_WAIT_2 ,stanje u kome klijent čeka sledeći segment od strane strane servera sa FIN bitom postavljenim na 1.

-Nakon što primi taj segment klijentski TCP ulazi u TIME_WAIT stanje.To stanje dopušta TCP klijentu klijentu da pošalje finalnu potvrdu u slučaju da je ACK izgubljen.

-Vreme provedeno u TIME_WAIT stanju zavisi od implementacije, ali tipične vrednosti su tridesetak sekundi, jedna minuta ili dve minute.

-Nakon ovog čekanja veza se na kraju formalno zatvori.

Protokoli pripremaju podatke po linearnoj osnovi. U predaji to bi bilo od gornjih slojeva prema donjima te obrnuto u prijemu. Svaki sloj zavisi o funkcionisanju sloja ispod njega, a da bi se to ostvarilo niži sloj koristi **enkapsulaciju** da bi prihvatio **PDU (Protocol Dana Unit)** višeg sloja. Svaki od slojeva unutar OSI modela ima neki oblik pakovanja podataka - **enkapsulacija**, i PDU je naziv za način pakovanja podataka u sloju pred propuštanje u naredni sloj. Enkapsulacijom se prihvaćenom PDU dodaje novo zaglavlje i završetak te time pravi novi PDU za naredni niži sloj. Svaki PDU ima svoje ime zavisno o sloju u kojem je kreiran (segment, packet, frame, bits). Dakle, računari međusobno komuniciraju kroz slojeve. Prilikom prijema

podataka, pri prelasku PDU-a iz nižeg sloja u viši oduzimaju se pojedina dodana zaglavlja i taj proces naziva se **dekapsulacija**.

## 3.2 TCP portovi

Tcp upotrebljava određen raspon portova kojima dodeljuje namenske programe na strani pošiljaoca i primaoca. Svaka strana TCP konekcije ima dodeljenu 16-bitnu oynaku za obe strane aplikacije ( slanje, primanje). Portovi su podeljeni u tri kategorije: poznati portovi, registrovani portovi i dinamički portovi.

Opšte poznati portovi (well known ports) su dodijeljeni od strane Internet Assigned Numbers Authority,organizacije koja se brine za IP adresni prostor i druge detalje vezane uz IP protokol. Ovi portovi su najčešće korišteni od strane sistemskih procesa, koje koriste poznate aplikacije kada primaju konekcije pasivno slušajući promet na tim portovima.Neki primeri opšte poznatih portova su: FTP (TCP port 21), Telnet (23), SMTP (25) i HTTP (80).

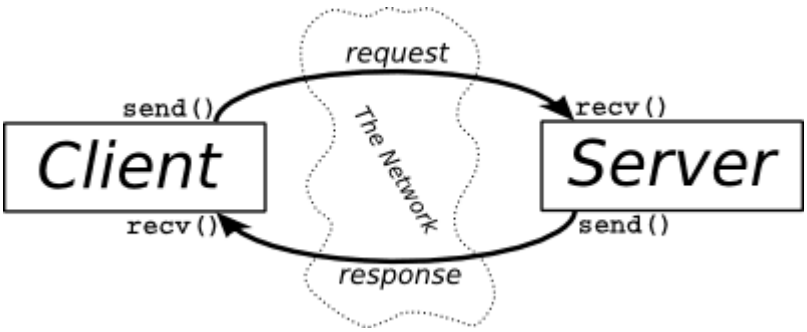
Registrovani portovi se koriste kod aplikacija krajnjih korisnika kao izvorišni portovi prilikom konekcije servera, kao i za identifikaciju servisa registrovanih od trećih strana.

Dinamički portovi se koriste i na strani aplikacija krajnjih korisnika, ali nešto ređe. Dinamički/privatni protovi imaju samo lokalno značenje za određenu TCP konekciju.

TCP protokol koristi portove od 1 do 65535, tj. ukupno ima 65535 mogućih različitih portova.

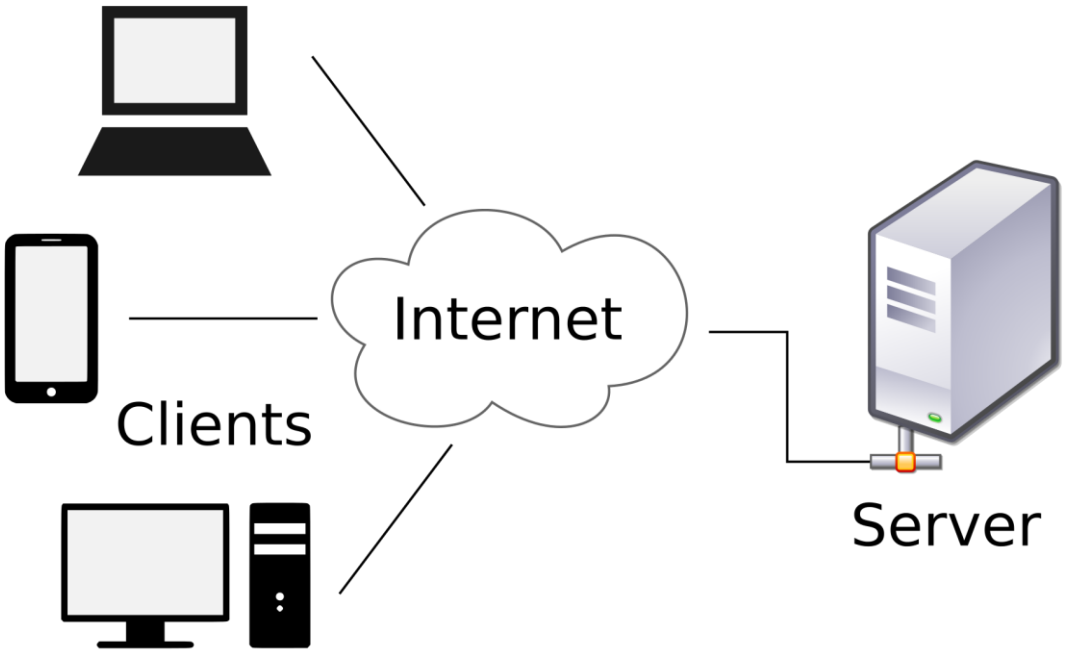
# 4. MREŽNO PROGRAMIRANjE

Mrežno programiranje obuhvata pisanje programa koji komuniciraju sa drugim programima preko računarske mreže. Jedan program se uobičajeno naziva klijent a drugi server. Jedan od primera klijent-server komunikacije je WEB browser u funkciji klijenta i WEB server kao server. Većinu mrežnih aplikacija možemo podeliti na te 2 grupe, klijent i server (Slika 4.1).



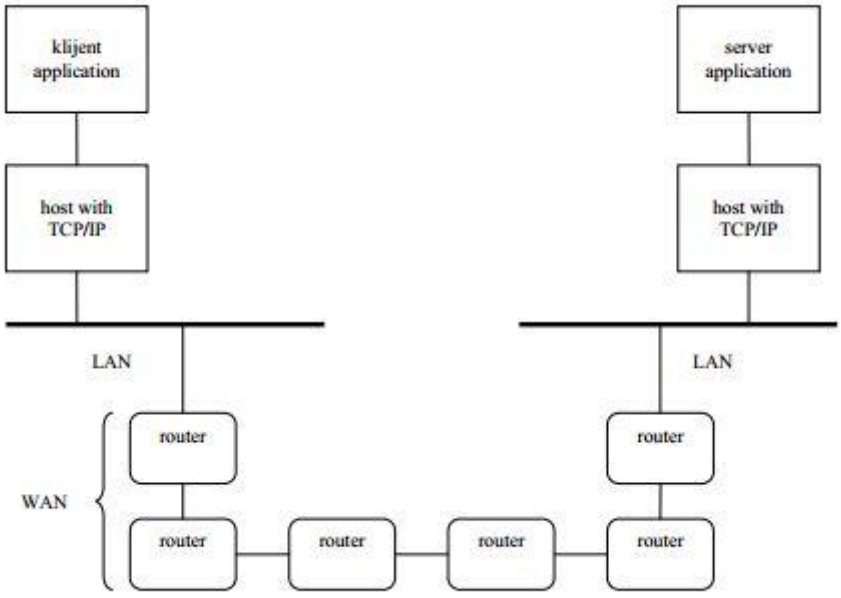

Slika 4.1 Klijent – Server arhitektura

Klijenti uobičajeno komuniciraju sa jednim serverom u isto vreme, mada u slučaju WEB browsera, često komuniciramo sa više servera. Za servere je uobičajeno da u jednom trenutku ima uspostavljenu vezu sa više različitih klijenata (slika 4.1).




Slika 4.2 Komunikacija servera sa više klijenata istovremeno

Klijent i server mogu se nalaziti u istoj lokalnoj mreži (Ethernet), ali mogu biti i u različitim mrežama, što je uobičajeno i slučaj (slika 4.3).




Slika 4.3. Klijent-Server na različitim mrežama

## 4.1 Primer upotrebe TCP protokola u programskom jeziku C#

Ako dva programa žele da komuniciraju preko mreže, moraju obrazovati po jedan **soket (socket)**. Socket se definiše kao jedan kraj komunikacijskog kanala. Za komunikaciju para procesa potrebaj je par socketa. Socket je određen IP adresom i brojem porta. Soketi se uglavnom koriste u klijent – server arhitekturi. Server čeka zahteve klijenata tako da sluša na određenom portu socketa. Serveri obično implementiraju specifične servise na određenom portu. Svi brojevi ispod 1024 su uglavnom korišteni od strane poznatih aplikacija.

Kada se klijent spoji na socket, dodeljuje mu se broj porta koji je veći od 1024, npr 53100. Kada se još jedan klijent spoji na server, dodeljuje mu se novi port, veći od 1024 a različit od 53100. Na taj način imamo uvek jedinstven par soketa za svaki par klijent-server.

Kako bi komunikacija bila uspešna, klijent mora znati kako IP adresu servera, tako i broj porta na kome server očekuje zahvtev.

U **klijentu** to izgleda:

```
IPHostEntry IPHost = Dns.GetHostByName(Dns.GetHostName());
textBox2.Text = IPHost.AddressList[0].ToString();
TcpClient client = new TcpClient(textBox2.Text, 53100);
```

U **serveru**:

```
TcpListener list = new TcpListener(53100);
list.Start();
TcpClient client = list.AcceptTcpClient();
```

Ovim smo izvukli IP adresu mreže u kojoj se računar nalazi i dodelili je metodi TcpClient zajedno sa brojem porta. U serveru, koristimo konstruktor TcpListener koji „osluškuje“ definisan port ( 53100) i klasu TcpClient koja omogućava prihvatanje i povezivanje klijenta na server.

Nakon prihvatanja i uspostavljanja konekcije, preko te konekcije se konstruišu ulazni i izlazni tok, StreamWriter i StreamReader, koji zapisuju i čitaju poslate podatke u toku definisanim Get.Stream() metodom:

U klijentu:

```
StreamWriter sw = new StreamWriter(client.GetStream());
```

U serveru:

StreamReader sr = new StreamReader(client.GetStream());

Cela aplikacija, i klijentski i serverski deo, izgledaju ovako:

**Klijent:**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.IO;

namespace Klijent
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      string n;
      byte[] b1;
```

```
OpenFileDialog op;
private void button1_Click(object sender, EventArgs e)
{
   op = new OpenFileDialog();
   if (op.ShowDialog() == DialogResult.OK)
   {
      string t = textBox1.Text;
      t = op.FileName;
      FileInfo fi = new FileInfo(textBox1.Text = op.FileName);
      n = fi.Name + "." + fi.Length;
      TcpClient client = new TcpClient(textBox2.Text, 53100);
      StreamWriter sw = new StreamWriter(client.GetStream());
      sw.WriteLine(n);
      sw.Flush();
      label1.Text = "Fajl pronadjen....";
   }
}
private void Form1_Load(object sender, EventArgs e)
{
   IPHostEntry IPHost = Dns.GetHostByName(Dns.GetHostName());
   textBox2.Text = IPHost.AddressList[0].ToString();
}
private void button2_Click(object sender, EventArgs e)
{
   TcpClient client = new TcpClient(textBox2.Text, 53100);
   Stream s = client.GetStream();
   b1 = File.ReadAllBytes(op.FileName);
   s.Write(b1, 0, b1.Length);
   client.Close();
```

```
        label1.Text = "Fajl poslat....";
    }
  }
}
```

**Server:**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net.Sockets;
using System.IO;
using System.Net;
using System.Threading;

namespace Server
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }



        StreamReader sr;
        Thread th;
        string rd;
        byte[] b1;
        string v;
        int m;
        TcpListener list;

        public void listThread()
        {
            TcpListener list = new TcpListener(53100);
            list.Start();
            TcpClient client = list.AcceptTcpClient();
            label1.Text = "Uspostavljanje konekcije....";
            StreamReader sr = new StreamReader(client.GetStream());
            rd = sr.ReadLine();
            v = rd.Substring(rd.LastIndexOf('.') + 1);
            m = int.Parse(v);
            list.Stop();
            client.Close();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
```

```
            IPHostEntry IPHost = Dns.GetHostByName(Dns.GetHostName());
            label1.Text = "IP adresa je " + IPHost.AddressList[0].ToString();
            Thread th = new Thread(new ThreadStart(listThread));
            th.Start();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
            {
                textBox1.Text = folderBrowserDialog1.SelectedPath;
                list = new TcpListener(53100);
                list.Start();
                TcpClient client = list.AcceptTcpClient();
                Stream s = client.GetStream();
                b1 = new byte[m];
                s.Read(b1, 0, b1.Length);
                File.WriteAllBytes(textBox1.Text + "\\" + rd.Substring(0,
rd.LastIndexOf('.')), b1);
                list.Stop();
                client.Close();
                label1.Text = "Fajl primljen......";
            }

        }


    }
}
```

# 5.LITERATURA

[1] http://searchnetworking.techtarget.com/definition/TCP-IP

[2] https://en.wikipedia.org/wiki/OSI_model

[3] http://www.sveznadar.info/20-WINTipsTricks/100-MrezaUvod/16-TCP.html

[4] http://mreze.layer-x.com/s010200-0.html

[5] http://www.opus1.com/ipv6/whatdoesanaddresslooklike.html

[6] http://stackoverflow.com/

[7] http://www.carstvolokvanja.com/knjige/programiranje/TCP-IP.pdf

[8] https://msdn.microsoft.com/en-us/default.aspx

[9] http://www.tcpipguide.com/index.htm

[10] https://en.wikipedia.org/wiki/Transmission_Control_Protocol

[11] http://www.thegeekstuff.com/2012/03/ip-protocol-header/

[12] http://www.sveznadar.info/20-WINTipsTricks/100-MrezaUvod/12-Protokol-OSI-TCPIP.html

[13] http://www.informatika.buzdo.com/s918-internet-tcp-ip-skup-protokola.htm

[14] Rob Miles, C# Yellow Book,(2011) University of Hull