Programmeringsparadigm

Internet F1

Vahid Mosavat

Innehåll

- Nätverksparadigm
- Kretskopplade vs paketväxlande nätverk
- Förbindelselös och förbindelseorienterad
- OSI lager
- protokoll
- Socket och portar

Nätverksparadigm

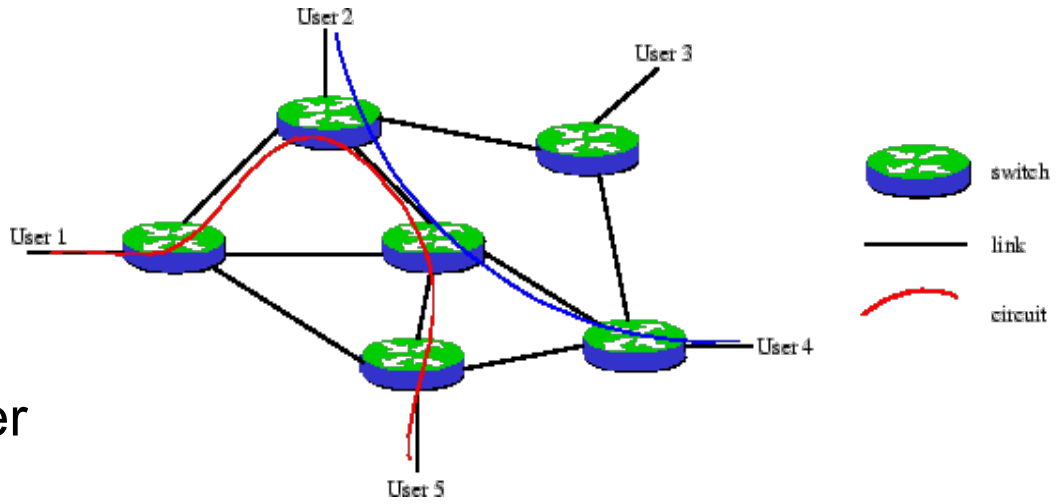
- Anslutningar
- Avstånd
- Levande struktur
- Komplexitet och icke linjäritet

Grundläggande begrepp

- Nätverkstyper
- Protokoll
- OSI-Lager
- Socket, enkel server-klient applikation

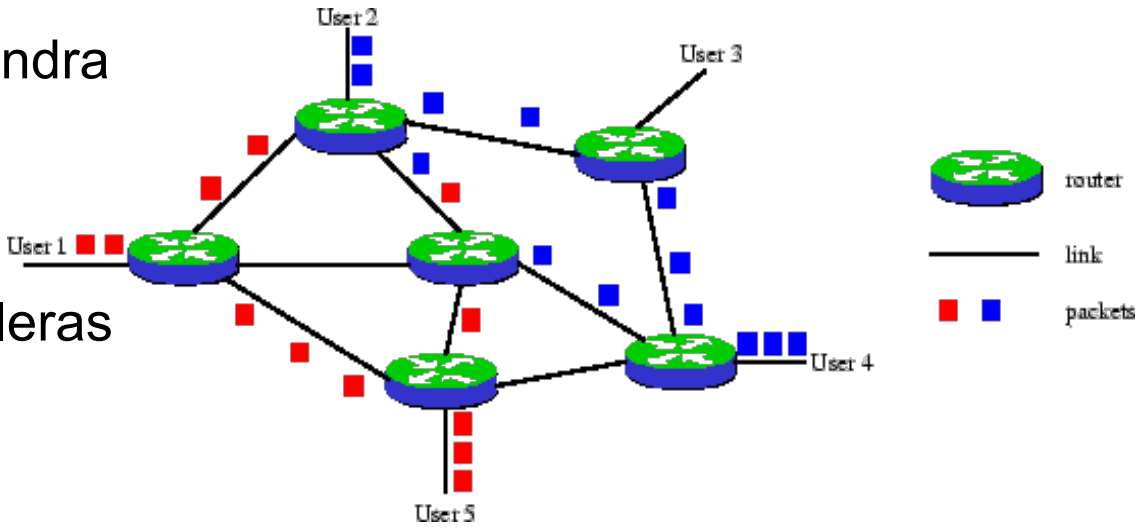
Kretskopplade nätverk

- En förbindelse mellan två enheter upprätts med hjälp av växlar innan överföringen kan starta.
- Alla inblandade resurser låses av denna uppkoppling
- Det kan ta lång tid att upprätthålla en anslutning
- Ömtålig, t.ex om en resurs går sönder under överföringen
- Begränsad kapacitet
- Analog med traditionellt telefon-samtal



Paketväxlande nätverk

- Förbindelselös kommunikation
- Pakten överförs oberoende av varandra
- Fullständig adressinformation om avsändare och mottagare
- Paketen kan ta olika vägar
- Paketen kan tappas bort eller dubbleras
- Nätet kan inte bli upptaget bara långsammare
- Hållbar överföring
- Ex: IP(Internet Protocol), TCP(Transmission Control Protocol), UDP (User Datagram Protocol)



internet och Internet

- Ett internet är en samling av nätverk som kan kommunicera med varandra, mans kulle kunna säga ett nätverk av nätverk

- Det mest kända internet kallas ”Internet”

Nätverksprotokoll

- Ett nätverksprotokoll är en samling av regler och överenskommelser kring kommunikation mellan två enheter som kommunicerar med varandra
- Innehåller bl.a uppgifter om:
  - Förbindelser
  - Vägval
  - Sönderdelning av data
  - Sammansättning av data
  - Upprättande av ordningsföljd vid överföring
  - Felkorrigering

Protokoll

- På lokala nätverk och Internet är dessa organiserade i fem följande lager:
  - Fysiska lagret: t.ex fiberoptik, seriell kabel etc
  - Länklagret: skapar förbindelser mellan enheter
  - **Nätverkslagret: hanterar vägval och flödesreglering**
  - **Transportlager: tar hand om ordningsföljd och felkorrigering**
  - **Applikationslagret: högsta nivån, här finns själva tillämpningsprotokollet, t.ex http, ftp, osv…**

- Datamängden som ska sändas delas upp i ett eller flera paket (datagram). Paketen från högre nivå förpackas i paket på lägre nivå. Denna struktur brukar benämnas protokollstacken.

Exempel på HTTP protokoll

~> **telnet www.nada.kth.se 80**

Trying 130.237.227.116...
Connected to sippans.csc.kth.se. Escape character is '^]'.

```
GET /~vahid/intnet18/f1.html HTTP/1.0
```

```
HTTP/1.1 200 OK
Date: Mon, 15 Jan 2018 22:14:15 GMT
Server: Apache
Last-Modified: Tue, 05 Sep 2017 12:58:38 GMT
ETag: "4eea4522-27-c9285b80”
Accept-Ranges: bytes
Content-Length: 39
Connection: close
Content-Type: text/html

<b>Internetprogrammering är roligt!</b>
```

Förfråga

Respons

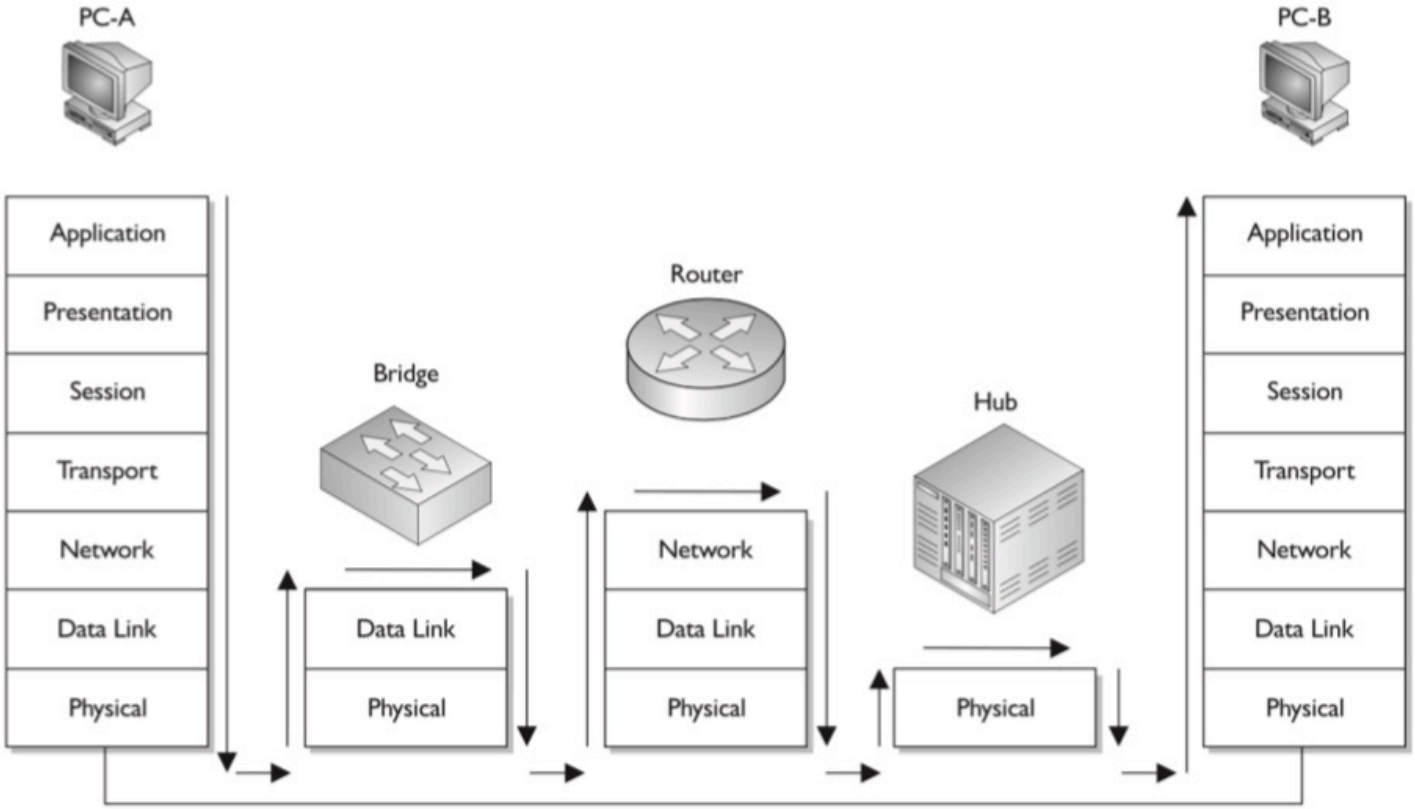
Connection closed by foreign host.

OSI-modellen

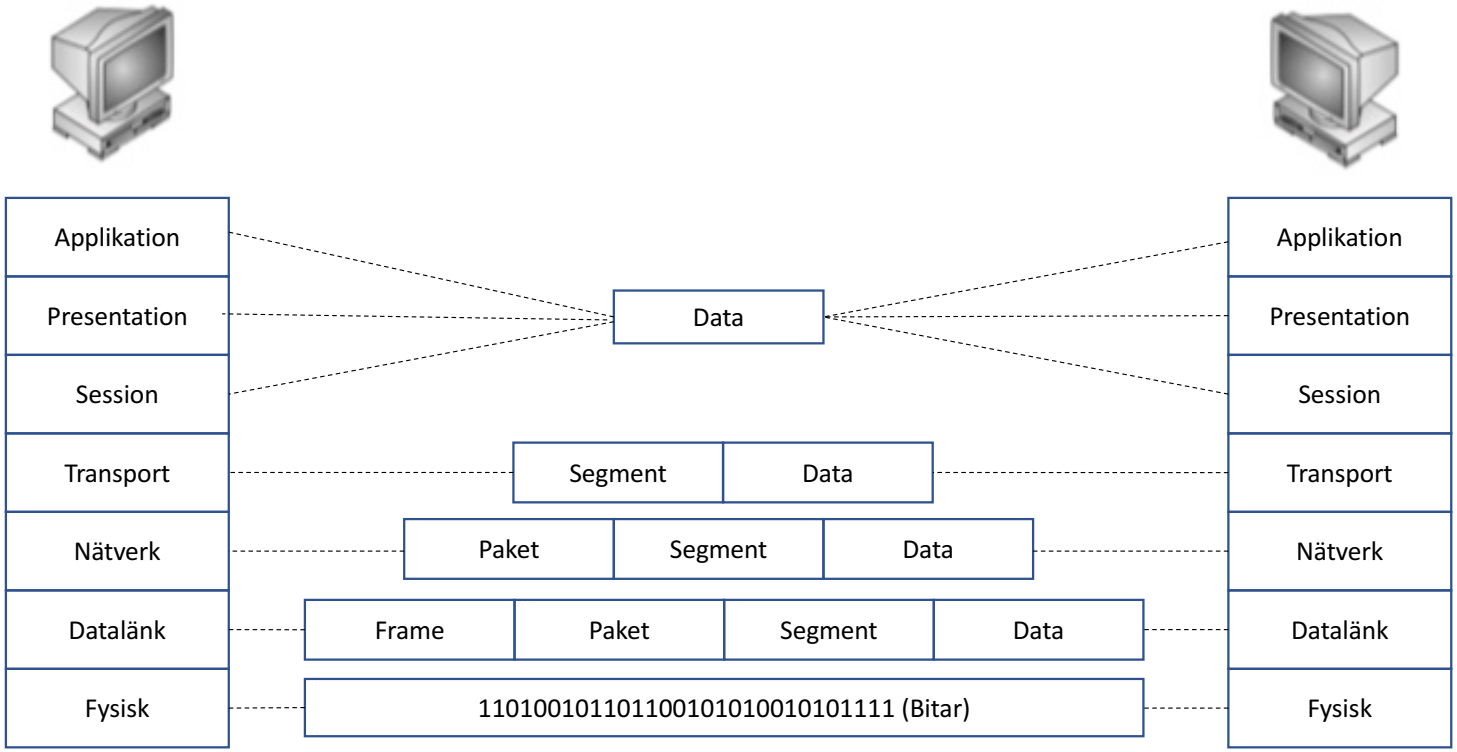
- Internationella organisationen för standardisering **(ISO)** har utvecklat en referensmodell bestående av sju lager, Open Systems Interconnection **(OSI)**, som beskriver hur information överförs från en maskin till en annan maskin.

- Modellen hjälper leverantörer och nätverksadministrationer få en bättre förståelse om hur data hanteras och överförs mellan olika nätverks anordning så väl som att förse en riktlinje för implementering av nya nätverks standard och teknologier.

Protokoll stacken



Protokollstacken



Datagram

Innehåller följande information

- Header
  - Avsändaradress
  - Destinationsadress
  - Typ av innehåll (t.ex TCP, UDP etc)
  - Längd av data
  - Felkontroll
- Data
  - Själva användarinformationen (**payload**)

Länklagret (Ethernet)

- Länklagret representerar den fysiska nivån på vilken all verklig kommunikation sker.
- På LAN utgörs länklagret oftast av Ethernet. Varje nätverkskort har här ett unikt 6 byte hexadecimalt identifikationsnummer, s.k. MAC-adress (Media Access Control).
- På WAN utgörs länklagret av t ex ATM. (**A**synchronous **T**ransfer **M**ode)

Nätverkslagret (IP)

- Ett protokoll för att överföra data mellan olika nätverk (därav ordet internet). Är oberoende av det underliggande nätets implementation (t.ex Ethernet, ATM)
- Då den verkliga kommunikationen endast sker på länklagret finns en teknik för att översätta mellan IP-adress och fysisk (länk) adress, ARP (adress resolution protocol).
- IP är ett tillståndslöst protokoll
- Paketen kan fördubblas eller försvinna
- En IP-adress (IPv4) består av 4 bytes och har formatet 130.237.225.94

- TTL (Time To Live) : kontrollfält

- Dagens IPv4 håller på att ersättas med IPv6 som har en adressrymd på 128 bitar.

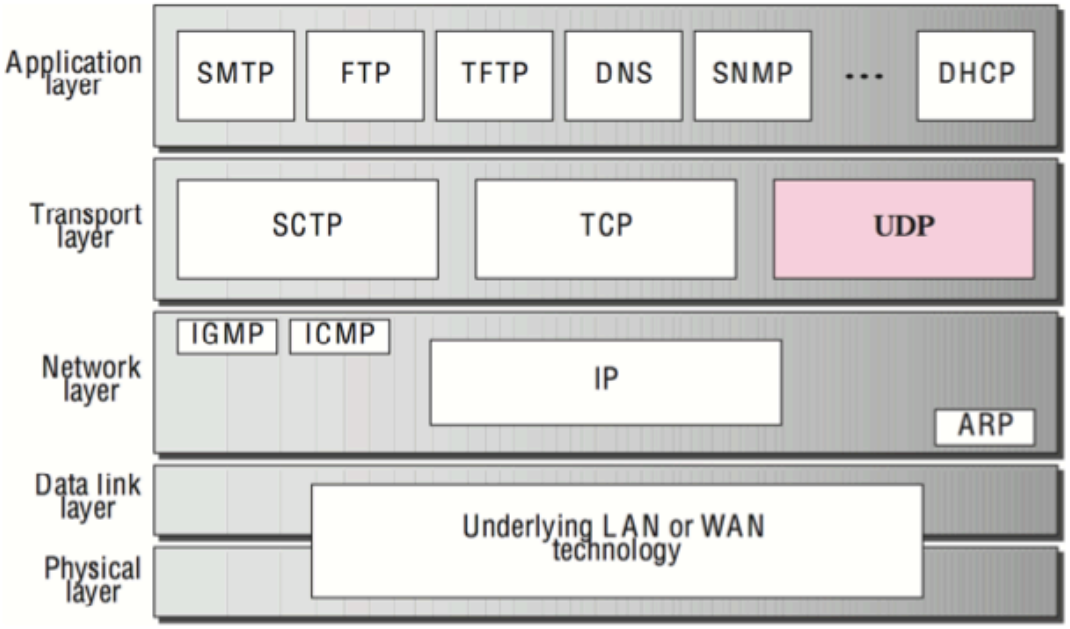
Transportlager: TCP, UDP och ICMP

- TCP (Transmission Control Protocol)
  - Förbindelseorienterad (logiskt)
  - Säker transport (blanda inte med sekretess)
  - Data förloras inte (förlorad data skickas igen)
  - Data fördubblas inte (fördubblad data slängs)
  - Data kommer fram i rätt ordning
- UDP (User Datagram protocol)
  - Förbindelselöst
  - Data kan förloras (kommer inte heller skickas igen)
  - Data kan fördubblas (fördubblad data slängs inte)
  - Data kan komma fram i fel ordning
- ICMP (Internet Controll Message Protocol)
  - Kopplat till TCP
  - Skickas av mottagare som väntat på ett paket som inte kommit fram
  - traceroute

Applikationslager

- I applikationslagret finns en mängd olika protokoll beroende på vilken tjänst som används:
  - HTTP
    - Surfning
  - HTTPS
    - Krypterad surfning
  - FTP
    - Filöverföring
  - POP3
    - Epost-mottagande
  - SMTP
    - Epost-skickande
  - DNS
    - IP adress <--> DNS-namn översättning
  - Telnet
    - Okrypterad fjärrinloggning
  - SSH
    - krypterad överföring, ofta för fjärrinloggning mot ett unix-skal
  - DHCP
    - För att tilldela klienter ett dynamisk ip-adress i ett nätverk.
  - Ping
    - För att kolla om en värd är ”uppe”

Förhållanden mellan protokollen



Socket

- Socket är gränssnitt mot TCP/IP och används alltså för att upprätta en anslutning baserat på IP mellan två maskiner
- En socketanslutning som t.ex som port 22 för att initieras men den fortsatta kommunikationen kan ske över valfri port på servern
- I java finns följande klasser implementerade
  - java.net.Socket
  - java.net.ServerSocket
- Klientens port slumpas

Server.java

```
import java.io.*;
import java.net.*;
public class Server{
  public static void main(String[] args) throws Exception{
    ServerSocket serverSckt = new ServerSocket(1234);
    Socket sckt = null;
    while((sckt = serverSckt.accept()) != null){
       BufferedReader indata = new BufferedReader(
                  new InputStreamReader(sckt.getInputStream()));
       String text = null;
       while( (text = indata.readLine()) != null)
              System.out.println(text);
       sckt.shutdownInput();
    }
  }
}
```

Client.java

```
import java.net.*;

import java.io.*;

public class Client{
  public static void main(String[] args){
    try{
      Socket sckt = new Socket("leguin.csc.kth.se",1234);
      PrintStream out = new PrintStream(sckt.getOutputStream());
      BufferedReader indata = new BufferedReader(new InputStreamReader(System.in));
      String text;
      while((text = indata.readLine()) != null)
             out.println(text);
      sckt.shutdownOutput();
    }catch (Exception e){System.err.println("Ett fel intraffade!");}

  }

}
```

Blockeringsproblem

- I exemplet blockeras ServerSocket av första anslutningen.
- Nya anslutningar accepteras, meddelanden tas emot men inget mer förrän första anslutningen stängs och därmed blockeringen upphävs.
- Trådar (klassen Thread) kan lösa problemet.

Server.java

```
import java.io.*;
import java.net.*;
public class Server{
  public static void main(String[] args) throws Exception{
    ServerSocket serverSckt = new ServerSocket(1234);
    Socket sckt = null;
    while((sckt = serverSckt.accept()) != null){

      Servant svnt = new Servant(sckt);

      svnt.start();
  }
}
```

Servant.java

```
class Servant extends Thread{
     Socket sckt;
     public Servant(Socket s){
      this.sckt=s;
     }
     public void run(){
        try{
          BufferedReader indata = new BufferedReader
            (new InputStreamReader(sckt.getInputStream()));
          String text = null;
          while( (text = indata.readLine()) != null)
                System.out.println(text);

          sckt.shutdownInput();

       }catch(IOException ioe){

          System.out.println("nagot fel intraffade!");}}}
```

UDPServer.java

```
class UDPServer{
   public static void main(String[] args) throws IOException{
      DatagramSocket ds = new DatagramSocket(1234);
      byte[] buf = new byte[1024];
      DatagramPacket dp = new DatagramPacket(buf, 1024);
      ds.receive(dp);
      String str = new String(dp.getData(), 0, dp.getLength());
      System.out.println(str);
      ds.close();
       }

  }
```

UDPClient.java

```
public class UDPClient{
    public static void main(String[] args) throws Exception {
      DatagramSocket ds = new DatagramSocket();
      String str = "hej hej";
      InetAddress ip = InetAddress.getByName("127.0.0.1");
      DatagramPacket dp = new DatagramPacket
                          (str.getBytes(), str.length(), ip, 1234);
      ds.send(dp);
      ds.close();
       }
}
```