Programmeringsparadigm

Internet F1

Vahid Mosavat

Innehåll

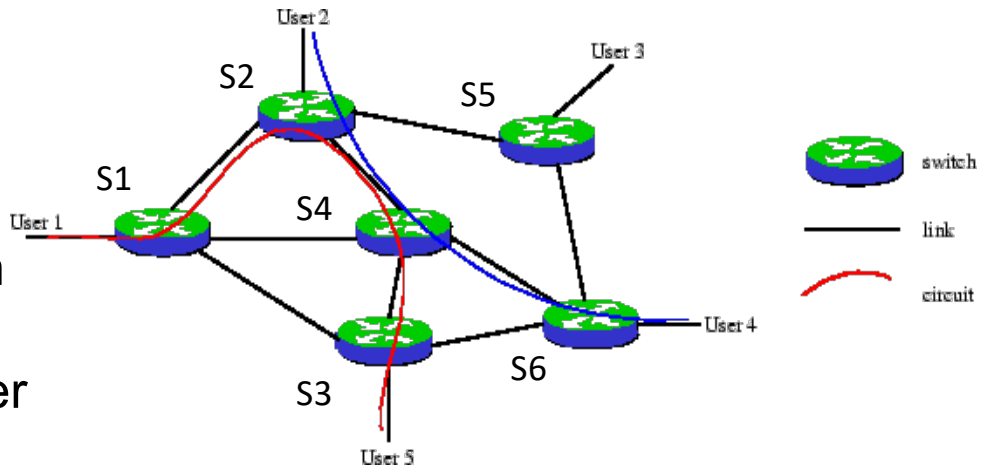
- Kretskopplade vs paketväxlande nätverk
- Förbindelselös och förbindelseorienterad
- IP, TCP och UDP
- Socket och portar

Grundläggande begrepp

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

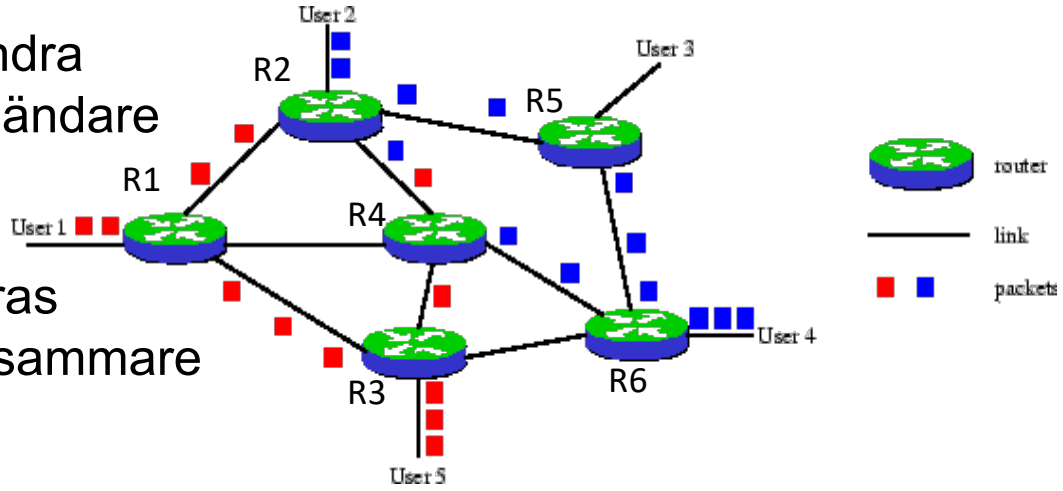
Kretskopplade nätverk

- 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 avbryts hela kommunikationen
- Begränsad kapacitet
- Exempel: traditionellt telefon-samtal



Paketväxlande nätverk

- Förbindelselös kommunikation
- Paketen 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)



Nätverksprotokoll

- Ett nätverksprotokoll är en samling av regler och överenskommelser mellan två enheter för att kunna genomföra kommunikationen
- Innehåller bl.a uppgifter om:
  - Förbindelser
  - Vägval
  - Sönderdelning av data
  - Sammansättning av data
  - Upprättande av ordningsföljd vid överföring
  - Felkorrigering

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

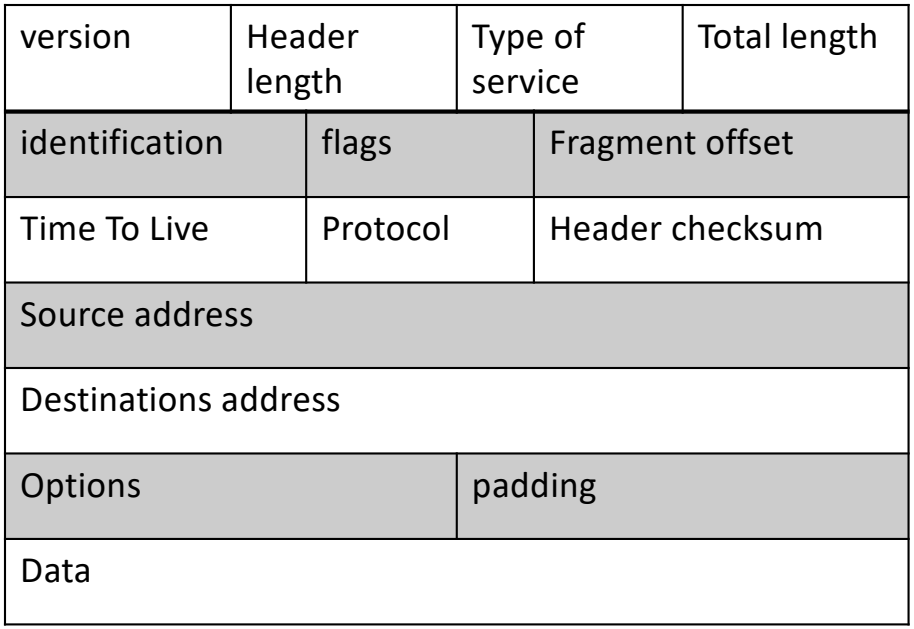
- TTL (Time To Live) : kontrollfält

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

IP-paket

Innehåller följande information

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



version	Header length	Type of service	Total length
identification	flags	Fragment offset	
Time To Live	Protocol	Header checksum	
Source address			
Destinations address			
Options		padding	
Data			

TCP

- **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”

Source port		Destination port	
Sequence number			
Acknowledge number			
header	reserved	code	window
checksum		urgent	
option		padding	
data			

UDP

- **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

Source port	Destination port
length	checksum
Data	

Socket

- Socket är gränssnitt mot TCP/IP och används för att upprätthålla en anslutning baserat på IP mellan två datorer
- En socketanslutning använder en port t.ex 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("127.0.0.1",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. D.v.s. endast en klient kan betjänas i taget.
- Nya anslutningar accepteras, meddelanden tas emot men inget mer förrän första anslutningen stängs och därmed blockeringen upphävs.
- Trådar (klassen Thread) kan användas för att lösa blockeringen.

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();
       }
}
```