Internetprogrammering DD1386

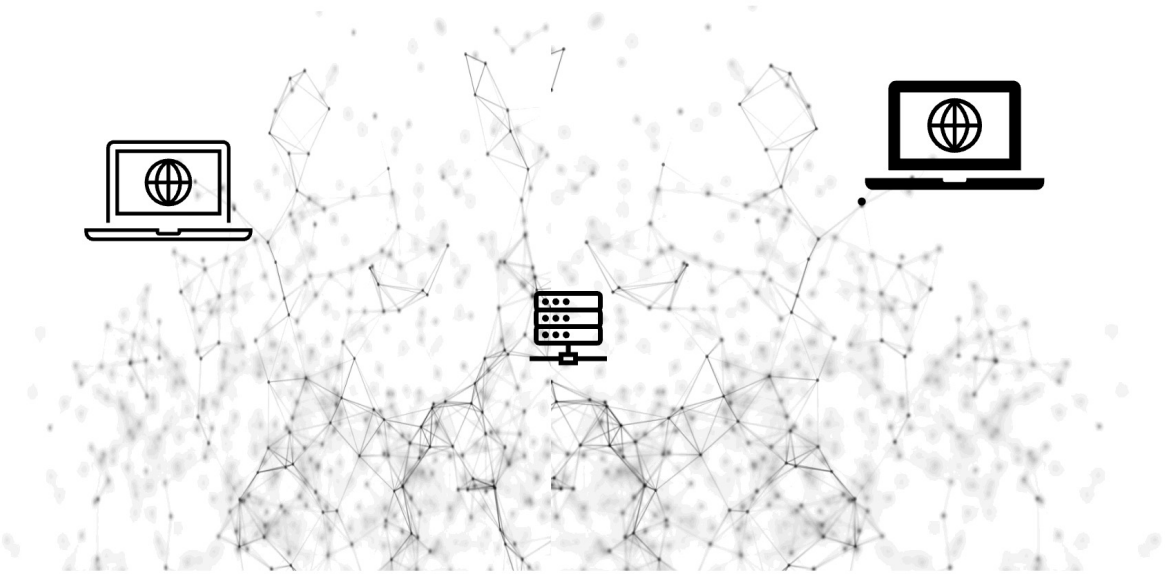
Föreläsning 1’ & 2’

- Kursinformation
- På klientsidan
- På serversidan
- Olika nätverkstyper
- Nätverksprotokoll
- HTTP- protokollet
- OSI-modellen
- Socket
- HTTP-protokollet
  - HttpClient
  - HttpServer
- Hantering tillstånd i HTTP-kommunikation
  - Headerfältet Set-cookie
  - Headerfältet Cookie

Kursinformation

- Kursledare och examinator: Vahid Mosavat, vahid@kth.se
- Kursaktiviteter: föreläsningar, räkne-stuga, laborationer
- Kursmaterial: labbinstruktioner, föreläsningsanteckningar, länkar till externa webbsidor
- Examination:
  - LAB1 (3.0 hp): 5 obligatoriska labbar labb 1-5 (grupp av 2)
  - PRO1(3.0 hp): Obligatoriskt projekt (grupp av 2) (Betyg E-A)
  - ÖVN1 (1.5 hp): övningsuppgifter i form av Quiz.
  - 3 bonusgivande labbuppgifter (frivilliga men påverkar betyget)
- Kurshemsidan: https://canvas.kth.se/courses/36975

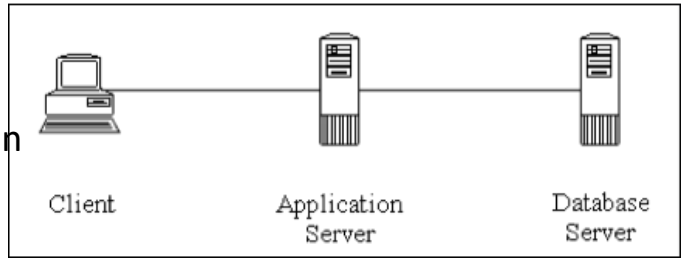
Internet



Allmänt

- Applikationer baserad på Client/Server modellen där klienten (vanligtvis) utgörs av en webbläsare och servern av en webbapplikation

- Treskiktslöningar
  - Klienten utgör presentations skiktet av tillämpningen
  - Servern hanterar den bakomliggande logiken
  - Databassystem utgör också ett eget skikt men behandlas endast i ringa omfattning i denna kurs



- Programspråket Java (Socket) och JavaScript (Node.js och andra ramverk så som Express, Vue, Vite) användas i labbarna och projektet.

Klientsidan

- Presentations skiktet behandlar främst layout men även enklare logiska funktioner som servern inte behöver blandas in t.ex. validering av inmatning. Vi kommer främst använda HTML, CSS och JavaScript.
- Modern webb använder sig av Ajax-teknik och liknande för att gå runt ”problemet” med tillståndslöshet (state less) som är grund till HTTP.

Serversidan

- Webbserver
  - HTTP, HTTPS (Hypertext Transfer Protocol)
  - Port 80
  - Tillståndslös
- Server-side scripting (kräver webbserver)
  - Genererar dynamisk HTML-sida
  - *PHP (5.3 och äldre versioner),*
  - *CGI- skript: c, perl, osv (egentligen nästan alla programspråk)*
- Webbapplikation
  - Genererar dynamisk HTML-sida med en utformning skräddarsydd för ett visst ändamål, t.ex webbmail, instant messaging, osv...
  - Node.js

… och dessutom

- Grundläggande nätverksterminologi
- Socket
- Kryptering (TSL/SSL)

Kursfordringar

Obligatoriska examinationer	Frivilliga uppgifter

• Ladok-moment LAB2 (3.0 hp) (P/F)
  • Labb 1: Gissa tal (HTTP-protokollet, Java, Socket)
  • Labb 2: Webbaserat Chomp (DHTML, JavaScript)
  • Labb 3: Webbaserat inloggnings sekvens (NodeJs)
  • Labb 4 (2 delar): Bokningssystem (NodeJs, Vue, Express)
  • Labb 5: Krypterad anslutning (TSL/SSL)
 	
• Ladok-moment PRO2 (3.0 hp)
  • Projekt (F-A)
 	
• Ladok-moment ÖVN1: Quiz (1.5hp) (P/F)
	• Bonusgivande uppgifter: Laborationerna 1,3 och 4 har var sin bonusgivande labbuppgift, X1, X3 och X4.
	• Bonus ges endast om de redovisas innan deadline.

Deadlines

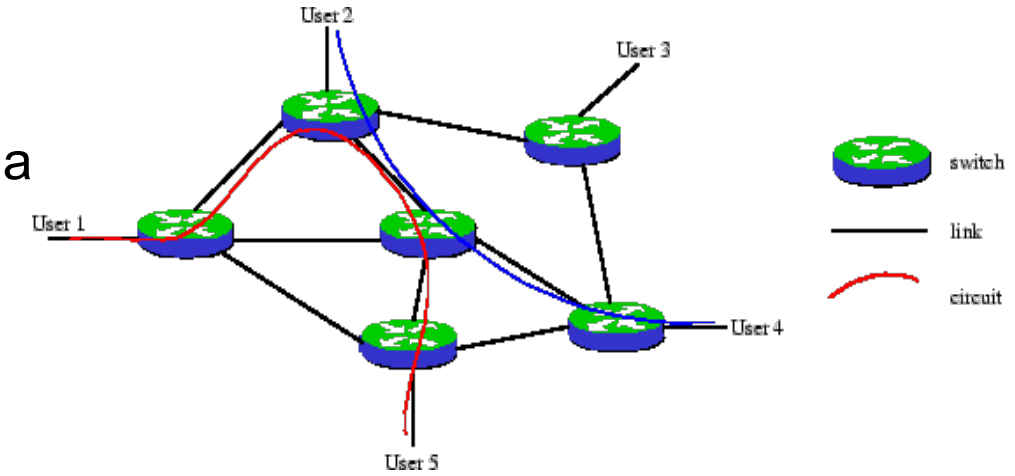
Uppgift	**Deadline**
Labb1X	2023-02-03
Labb3X	2023-02-24
Labb4X	2023-04-06
Projekt	2023-05-11

Grundläggande begrepp

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

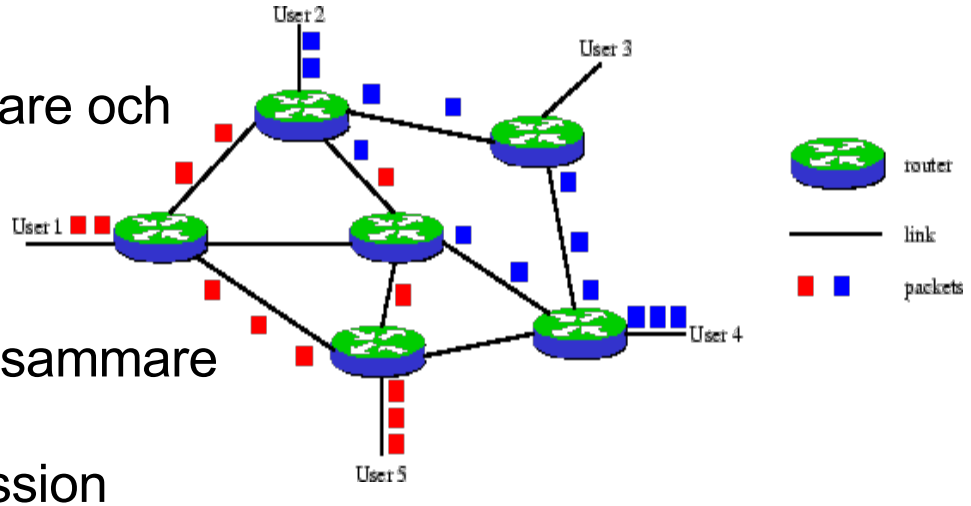
Kretskopplade nätverk

- 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

- Förbindelse-lös 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 utan bara långsammare
- Hållbar överföring
- T.ex: IP(Internet Protocol), TCP(Transmission Control Protocol),
UDP (User Datagram Protocol)



Protokoll

- Ett protokoll 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

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 2020 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

Respon s

Connection closed by foreign host.

~>

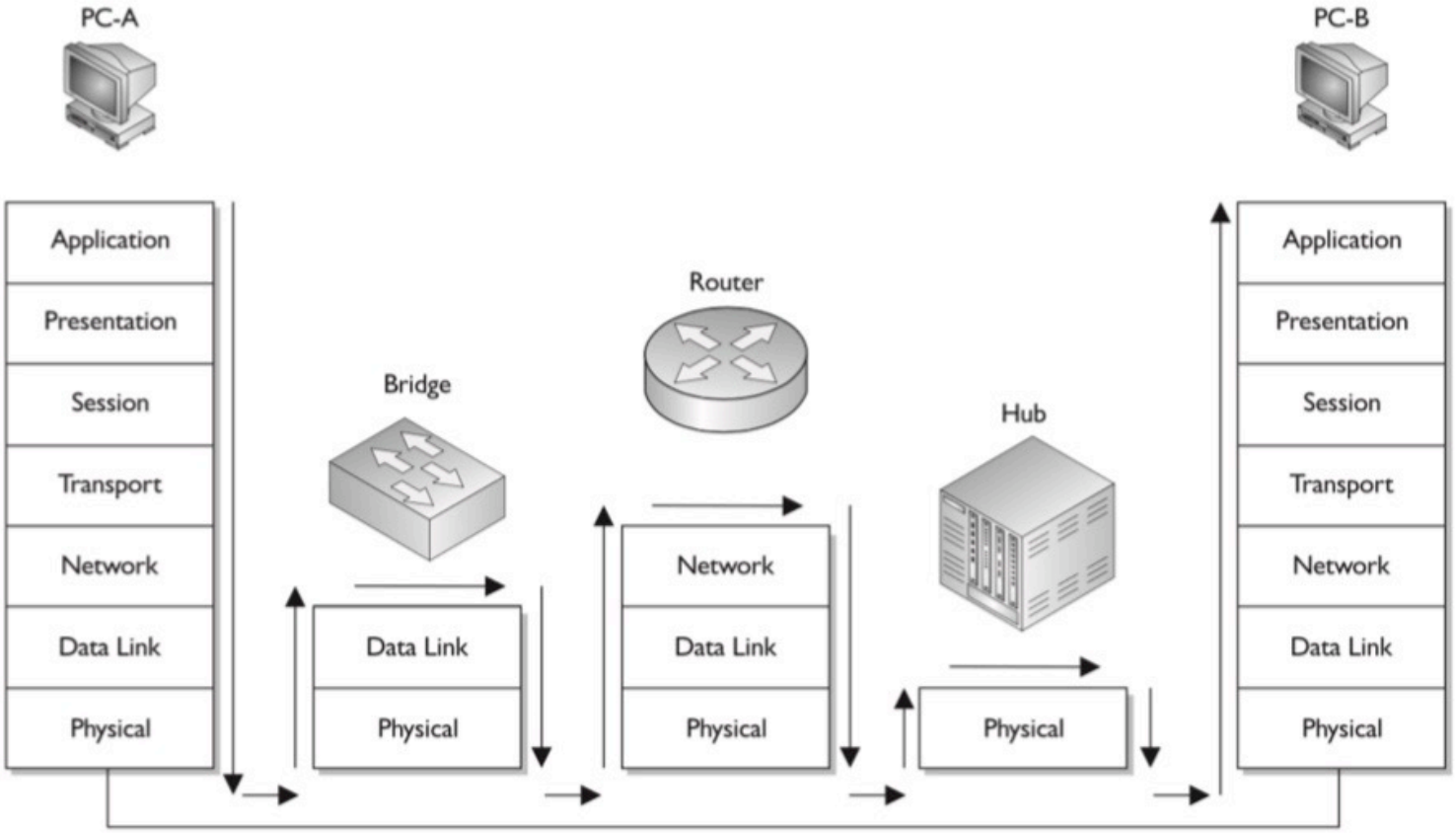
Exempel på HTTP protokoll

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

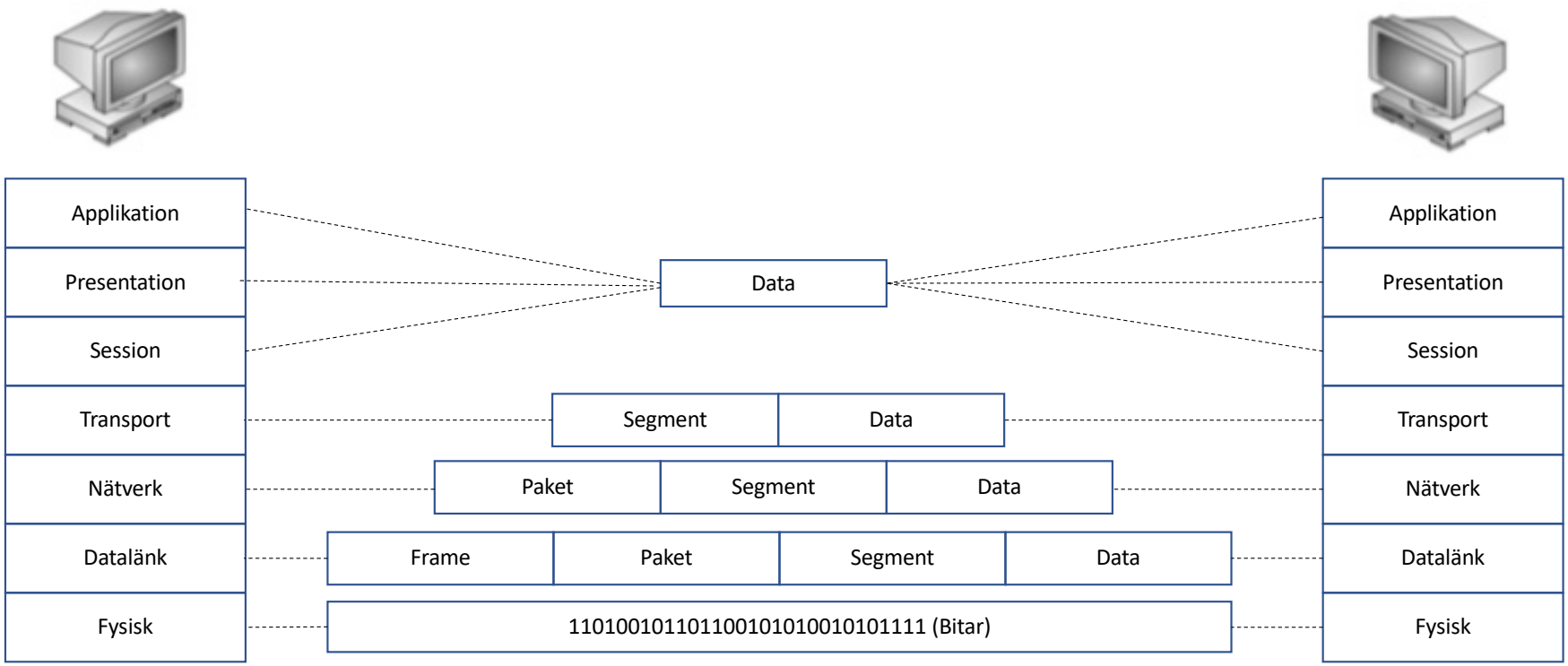
curl https://www.csc.kth.se/~vahid/intnet23/f1.html

< HTTP/2 200
< date: Mon, 16 Jan 2023 22:56:22 GMT
< server: Apache
< accept-ranges: bytes
< referrer-policy: origin-when-cross-origin
< content-length: 32
< content-type: text/html
< strict-transport-security: max-age=63072000

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)

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.

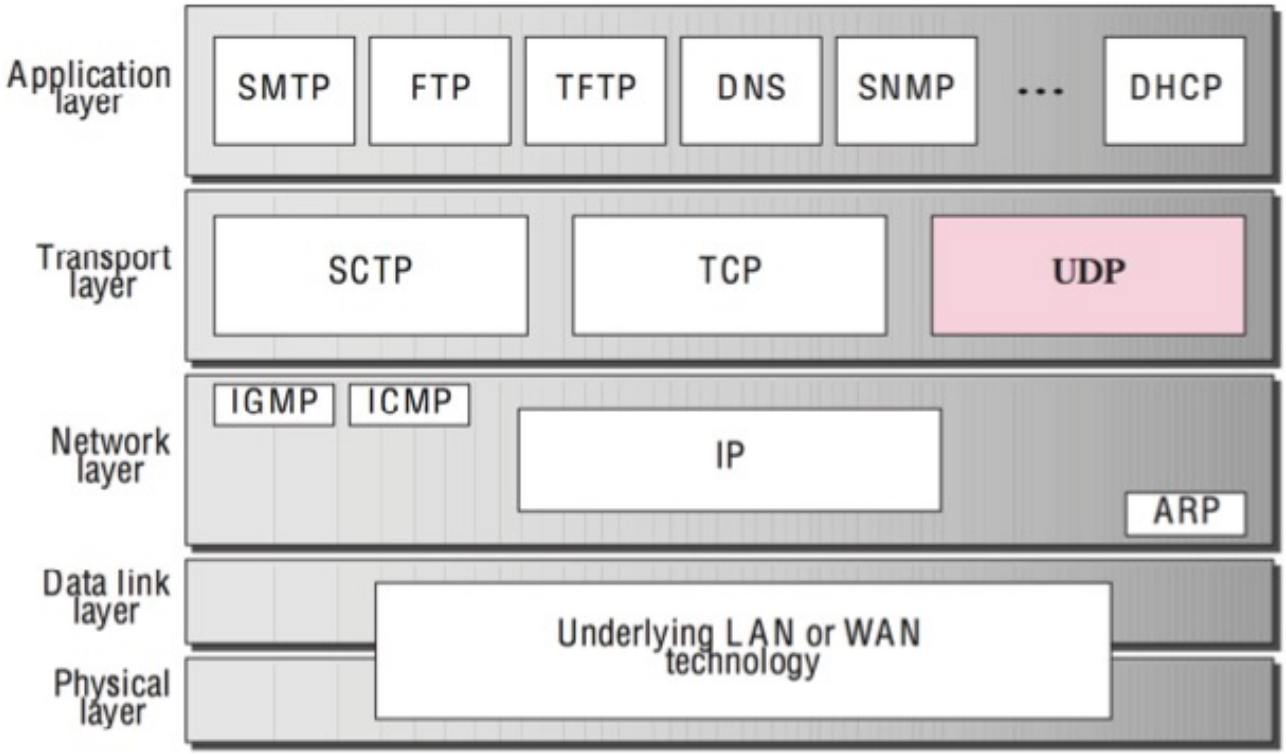
Transportlager: TCP, UDP och ICMP

- 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 Control Message Protocol)
  - Kopplat till TCP
  - Skickas av mottagare som väntat på ett paket som inte kommit fram
  - `traceroute`

Applikationslager

- 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 UDP/IP och används alltså för att upprätta en anslutning baserat på IP mellan två maskiner
- En socketanslutning som exempelvis använder 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.net.*;

import java.io.*;

public class Server{
        public static void main(String[] args) throws Exception{
                ServerSocket serverSckt = new ServerSocket(1234);

                Socket sckt = serverSckt.accept();
                while( (sckt != null){

                        BufferedReader indata = new BufferedReader(
                        new InputStreamReader(sckt.getInputStream()));

                        String text = null;
                        while( (text = indata.readLine()) != null){

                                System.out.println(text);

                        }

                        sckt.shutdownInput();

                        sckt = serverSckt.accept();

                }

        }

}
```

Client.java

```
import java.net.*;

import java.io.*;

public class Client{
   public static void main(String[] args){

      try{
         Socket sckt = new Socket("share-02.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!");

      }

  }}
```

HTTP protokollet

- Applikationslagret
- Tillståndslös
- Proceduren är följande:
  - Klienten öppnar en anslutning
  - Klienten skickar en förfråga (request)
  - Servern skickar ett svar (response)
  - Servern stänger anslutningen
- HTTP-protokollet är en specifikation för hur denna kommunikation ska ske. Det kan betraktas som en grammatisk kravspecifikation för två rollerna klient och server
  - HTTP-request (skickas alltid från en klient till en server som en begäran)
  - HTTP-response (skickas alltid från en server till en klient som svar till requesten)
- Det är helt upp till utvecklaren av webbläsaren / webbservern att implementera stödet för denna specifikation.

URL

- En URL (***U**niform **R**esource **L**ocator)* är en form av resurspekare (adress till en resurs över internet)

- Vad är en resurs?
  - En statisk fil som ska skickas bit för bit till klienten.
  - Eller en exekverbar fil som ska exekveras hos servern och utdata från exekveringen ska skickas till klienten. De filer som ska exekveras förs har konfigurerats hos servern så att servern inte skickar den exekverbara filen till klienten.

URL

- Exempel på url:
  - http://localhost:8080/index.html
  - http://www.eecs.kth.se/
  - ftp://ftp.sunet.se

- Standardiserat format:
  protokoll://värd[:port]/sökvägen till resursen
  t.ex:
- default port för webbservrar är 80 vilket betyder att det behöver inte anges i urlen.

HTTP-request

- `GET /~vahid/jag.jpg HTTP/1.1`
- `Accept: text/html, application/xhtml+xml, */*`
- `Accept-Language: sv-SE`
- `User-Agent: Mozilla/5.0 (Windows NT 6.1…`
- `Accept-Encoding: gzip, deflate`
- `Host: www.csc.kth.se`
- `Connection: close`

HTTP-request

- Formatet för en http-request är:
  - Start line: <metod> <sökväg> <HTTP-version> ↲
  - Header 1: <variabel> : <värde> ↲
  - Header 2: <variabel> : <värde> ↲
  - Header 3: <variabel> : <värde> ↲
  - ...
  - Header N: <variabel> : <värde> ↲
  - ↲
  - Payload: Eventuell data...

OBS väldigt viktigt!!
Två efterföljande radbrytningar innan payload börjar

- Vanligaste metoder är GET och POST. Med GET blir formulärparametrar en del av URL:en och med POST lagras dem i meddelandekroppen.
- I header:n lagras information om webbläsaren, dess miljö och eventuella cookies som klienten skickar med till servern.
- Sökväg anges relativt serverns rot, t.ex `/index.html?file=meny.html`

HTTP-metoder

Viktiga Metoder	**Beskrivning**
GET	Frågar efter en resurs
POST	Skicka data till servern för att skapa en resurs

Mindre viktiga Metoder	**Beskrivning**
HEAD	Frågar efter information om en resurs
PUT	Skicka data till servern för att uppdatera en resurs
DELETE	Ta bort en resurs
TRACE	När förfrågan måste passera genom en proxy, gateway etc
OPTIONS	Frågar efter tillåtna (tillgängliga) metoder

Viktiga headerfält i en request

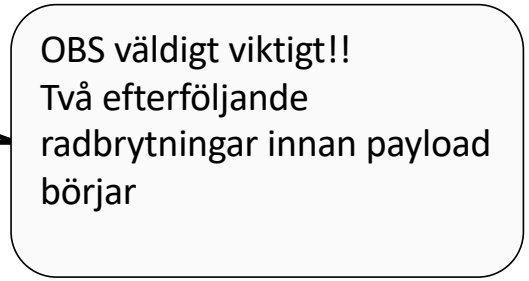
- Accept: vilka mime-type av resurs kan hanteras
- User-Agent: t.ex Mozilla/4.75[en] (Win98)
- Host: serverns domän namn och portnummer
- Cookie: används för identifiering
- Date: när paketet skapades

HTTP-response (binär exempel)

- HTTP/1.1 200 OK↲
- Date: Sun, 21 Jan 2018 15:27:34 GMT↲
- Server: Apache↲
- Accept-Ranges: bytes↲
- Content-Length: 2946↲
- Keep-Alive: timeout=3, max=98↲
- Connection: Keep-Alive↲
- Content-Type: image/gif↲
- ↲
- ***GIF89a***

HTTP-response

- Formatet för en http-response är:
  - Start line: <http-version> <statuskod> <OK/Error>↲
  - header: <variabel> : <värde> ↲
  - …
  - header: <variabel> : <värde> ↲
  - ↲
  - payload: data...



- I header:n bör alltid *Content-Type* ingå för att webbläsaren ska kunna skilja text från binär information etc. Anges I MIME-format (t.ex *text/plain, text/html, image/jpeg*) och specificerar innehållet i *message*.
- OBS!!! Det är endast *payload* (*body*) som syns när man väljer “view source” i webbläsaren.

Viktiga headerfält i en response

- Content-Type: mimetype
- Content-Length: storlek av data
- Server: info om server, t.ex: Apache
- Set-Cookie: skickar kaka
- Date: datum då servern skapade responsen

Hanteraing av tillstånd

- Applikationslagret
- HTTP är tillståndslös, d.v.s. det finns inte stöd för att servern ska komma ihåg något om tidigare request
- Hantering av tillstånd:
  - Headern Set-Cookie i responsen
  - Headern Cookie i request för att hantera tillstånd

Cookies

- Då man i en webserverapplikation vanligtvis vill kunna skilja de olika klienterna från varandra och detta inte stöds av HTTP låter man servern spara en s.k. *cookie* hos klienten.
- En cookie har ett namn och ett värde och sparas i webbläsarens minne.
- Servern definierar cookie:n i response header:n
  - – Set-Cookie: <namn> = <värde>↲
  - – expires = <datum>↲
  - – domain = <domän>↲
  - – path = <bibliotek>↲
  - – *Set-Cookie: JSESSIONID=751668B0C61F226904B; Path=/*↲
- Om klienten vid ett senare tillfälle försöker komma åt samma domän/sökväg skickas cookie:n med i request-header:n. – *"Cookie: JSESSIONID=751668B0C61F226904B;"*

Response med set-cookie

- `HTTP/1.1 200 OK↵`
- `Content-type: text/html↵`
- **`Set-Cookie: status = morgonpigg`**`↵`
- **`Set-Cookie:  likes = 1038`**`↵`
- `↵`
- `[Payload]`

Request med Cookie

- `GET /step1 HTTP/1.1`
- `Host: u-shell.csc.kth.se`
- **`Cookie: vardagar = morgonpigg; likes = 1038`**

Format

- Set-Cookie: <name>=<value> ← raderas när man stänger av webbläsaren
- Set-Cookie: <name>=<value>; Expires=<date>
- Set-Cookie: <name>=<value>; Max-Age=<non-zero-digit>
- Set-Cookie: <name>=<value>; Domain=<domain-value>
- Set-Cookie: <name>=<value>; Path=<path-value>

…

Exempel:

- Set-Cookie: id= 47w123
- Set-Cookie: id= 47w123; Max-Age=3600
- Set-Cookie: id= 47w123; Expires=Fri, 21 Jan 2022 09:00:00 GMT;
- Set-Cookie: id= 47w123; Domain=kth.se

Exempel

- Vi går igenom Form.java och lägger till Set-Cookie för den.