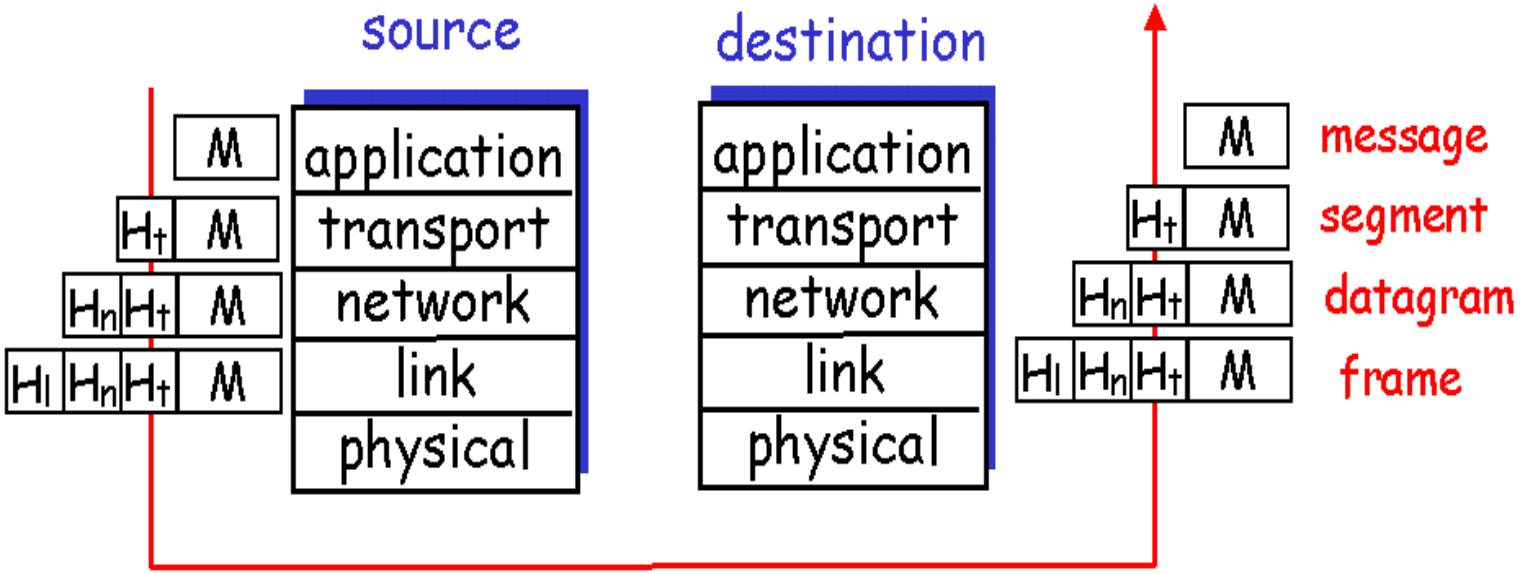
# TK1100
# Digital teknologi

## 7. Forelesning: Applikasjonslaget

# Sist

- Generelt
  - Internett er infrastruktur
- Linje- vs Pakke-svitsjing
  - Forsinkelser
- Lagdelt protokoll-stack:

- Protokoller
  - Regelverket som aktive nettverkskomponenter (applikasjoner, vertsmaskiner, routere) benytter når de kommuniserer
  - Definerer
    - Syntax: Format/rekkefølger på data/meldinger som utveksles (når)
    - Semantikk: Hvilke handlinger som skal foretas når en gitt melding mottas
    - Timing
    - Se **RFC**



# Sist (2)

- HTTP
  - request/response
  - format, header
  - GET, POST, …
  - 200 OK, 404 …
  - betinget GET og cacheing (304)
  - tilstandsløs
    - Autentisering/autorisering **må** følge med i header
    - **Cookies** for å lage ID'er som følger med i header

| Navn på laget | Betegnelse på overføringsenhet | Viktigste oppgaver/funksjoner<br>Exempel på protokoller/standarder |
|---|---|---|
| Applikasjonslaget | Melding (Message) | Støtte nettverksapplikasjoner<br>Ex: HTTP, DNS, FTP, SMTP, POP3…. |
| Transportlaget | Segment | Transport av applikasjonslagsmeldinger mellom klient- og tjener-sidene til en applikasjon; herunder mux/demux, ulike nivåer av pålitelighet med mer<br>Ex: TCP, UDP,.. |
| Nettverkslaget | Datagram | Routing av datagram fra/til vertsmaskin gjennom nettverkskjernen<br>Ex: IP (v4 og v6), ICMP, RIP, OSPF, BGP,… |
| Datalinjelaget | Ramme (Frame) | (Pålitelig) Levering av ramme fra nabo-node til nabo-node<br>Ex: Ethernet II, FDDI, IEEE 802.11 … |
| Fysisk | Bit | (Kode og) Flytte enkeltbit mellom kommunikasjonspartnere<br>Ex: 10BaseT, … |

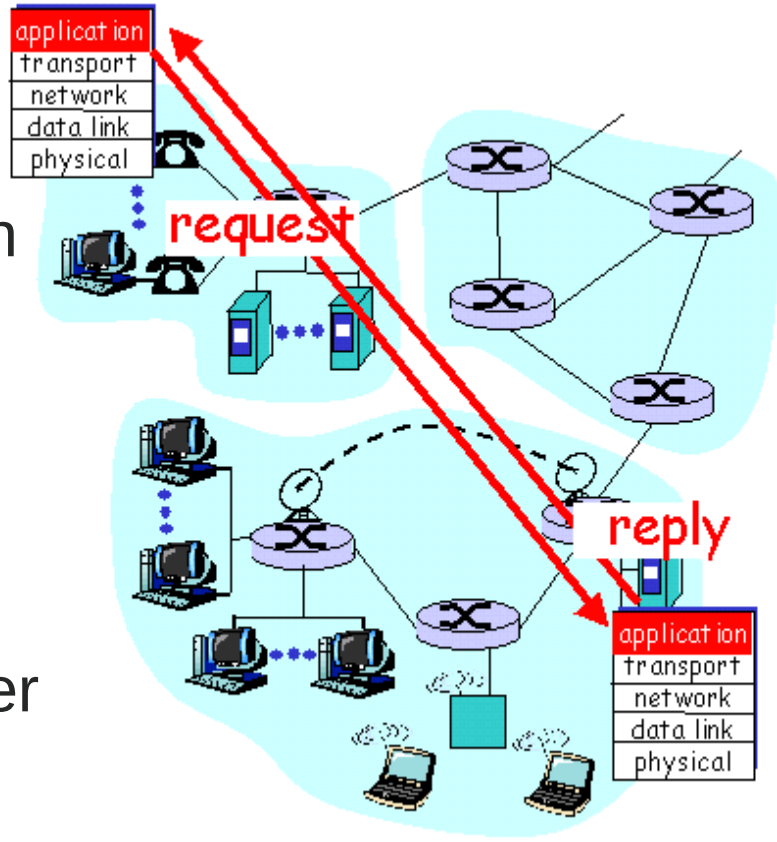
# APPLIKASJONS-LAGET

# Applikasjonslaget

- Hensikten med nettverket er å kjøre applikasjoner som fysisk er lokalisert på flere steder (distribuert)
- Vi skal se på nettverkskonsepter
  - Klient/tjener
  - Service modeller (ToS/QoS)
    - Tjeneste Typer, Tjeneste Kvalitet
- Se nærmere på noen protokoller
  - HTTP, (FTP), DNS, SMTP og (POP3/IMAP)

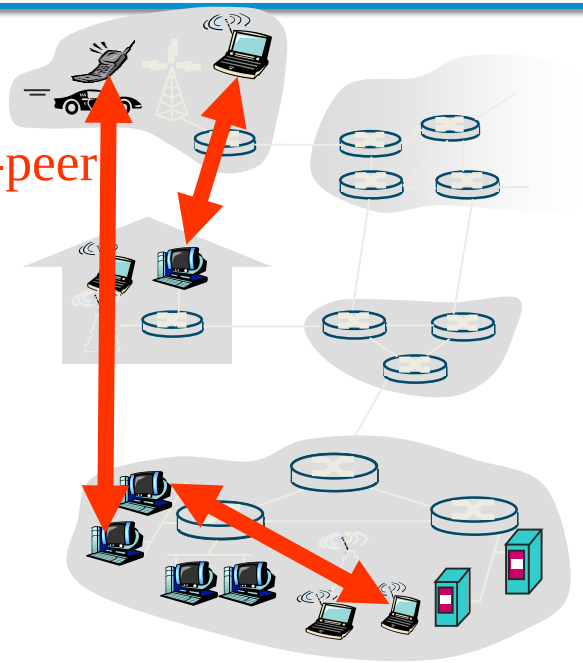
# Klient/tjener

- Typisk oppsett i et nettverk
- **Klient**
  - Tar initiativet
  - Ber om en service fra tjeneren
  - På web er klienten i browseren
- **Tjener**
  - Leverer etterspurt service til klienten
  - Står «alltid på»
  - Har en fast, velkjent adresse
  - Er «flaskehals» fordi alle bruker den samme serveren/server-parken (lastbalansering mulig/nødvendig)



# Peer-to-Peer (P2P)

- Minimalt/intet behov for at noen alltid står på.
- Alle kan både be om og levere tjenesten
- BitTorrent, LimeWire, Skype,…
- Selv-skalerende
  - I et fildelingsnettverk vil hver «klient» også øke antall «tjenere» og samlet kapasitet
- Noen problemer
  - Opphavsrett og fildeling
  - ASDL, kabel m.fl. er laget for **asymmetriske** (klient/tjener) trafikk: mye ned-, lite opp-lasting. Problematisk for ISPer.
  - **Sikkerhet og pålitelighet** er **vanskelig** i distribuerte systemer
  - Mange brukere struper opplasting og maksimerer nedlasting, noe som gjøre P2P ineffektivt. (Hvor mange vil *egentlig* dele computeren sin med andre?)

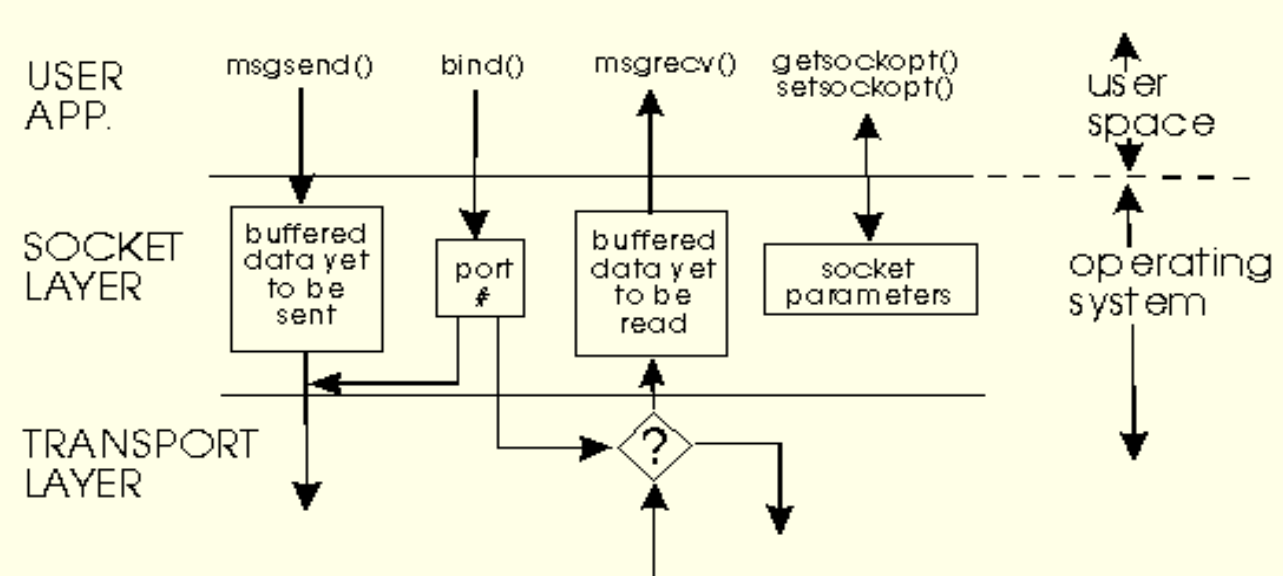


# Hybrid klient/tjener P2P løsninger

- Skype
  - Sentral tjener for å slå opp adresse til mottager – registrer seg selv der
  - Samtale er P2P
- MSN o.l.
  - Sentral tjener
    - der du logger på og (automatisk) oppgir kontaktinfo (IP-adresse mm)
    - kontakter sentral-register for å få adresse til samtalepartner.
  - Samtaler/Chat er P2P

# Sockets (API)

- Definerer forbindelsen (”grensesnittet”) mellom applikasjons- og transport-laget
- Socket = ”Internett API”
  - To prosesser kommuniserer med hverandre over Internett ved å sende data inn i socket og lese data ut fra socket
- Adresse til ønsket kommunikasjons-partner dannes av IP-adresse (**vertsmaskin**-«id») **og** port-nummer (**prosess**-”id”)



# Kritiske tjenestenivåer for applikasjoner

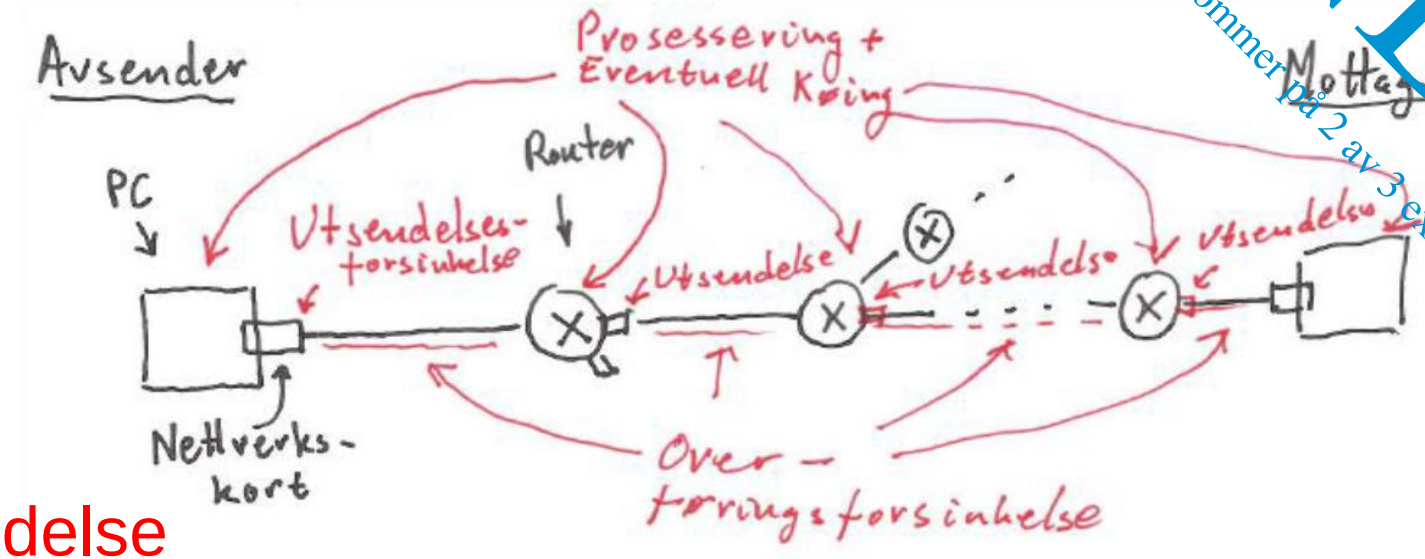
- Tap av data
  - Noen applikasjoner tåler litt tap av data
    - Audio, video
  - Andre må ha 100% pålitelig dataoverføring
    - Filoverføring
- Båndbredde/bit-rate (bps)
  - Noen applikasjoner må ha en viss båndbredde
    - Multimedia, streaming
    - Caching kan forbedre brukeropplevelsen, men kun dersom man ikke har sanntidskrav
  - Andre kan bruke båndbredden dynamisk
    - Filoverføring
- Timing
  - Noen applikasjoner tåler ikke mye tidsforsinkelse («latency»)
    - Sanntidsprosesser, spill

# Service-eksempler

| Application | Data loss | Bandwidth | Time Sensitive |
|---|---|---|---|
| file transfer | no loss | elastic | no |
| e-mail | no loss | elastic | no |
| Web documents | loss-tolerant | elastic | no |
| real-time audio/video | loss-tolerant | audio: 5Kb-1Mb<br>video:10Kb-5Mb | yes, 100's msec |
| stored audio/video | loss-tolerant | same as above | yes, few secs |
| interactive games | loss-tolerant | few Kbps up | yes, 100's msec |
| financial apps | no loss | elastic | yes and no |

# Forsinkelse-typer

NB!
(kommer på 2 av 3 eksamener)




- Utsendelse
  - Bestemt av **nettverkskort**, medium og protokoll (F.eks. 54 Mbps i trådløst)
- Overføring
  - Bestemt av **fysisk avstand** og signal-hastigheten
  - Ca 2/3 av lyshastigheten (200 000 000 m/s) i kobber/fiber, nesten lyshastigheten i luft
- Prosessering
  - Tiden det tar å lese/endre headere avhenger av hastigheten på **hardware i router/switch**
- Køing
  - Tiden en pakke må vente før den videresendes fra hver router, avhenger av **trafikken**

# Service i transport protokoller

- TCP
  - Forbindelsesorientert
  - Pålitelig transport
  - Flytkontroll
  - Trafikkork-kontroll
  - Ikke timing kontroll eller minimumsgaranti for båndbredde
- UDP
  - Lettvekts-protokoll uten garantier
  - Brukes der applikasjonen selv kan sørge for pålitelighet, der det er små meldinger og korte avstander mm
- Dette skal vi se mye nærmere på i neste forelesning!

# Applikasjoner og tr-protokoller

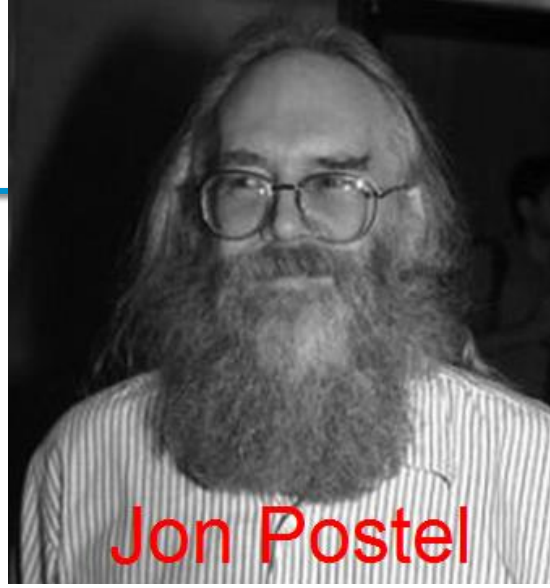
| Application | Application layer protocol | Underlying transport protocol |
|---|---|---|
| e-mail | smtp [RFC 821] | TCP |
| remote terminal access | telnet [RFC 854] | TCP |
| Web | http [RFC 2068] | TCP |
| file transfer | ftp [RFC 959] | TCP |
| streaming multimedia | proprietary (e.g. RealNetworks) | TCP or UDP |
| remote file server | NSF | TCP or UDP |
| Internet telephony | proprietary (e.g., Vocaltec) | typically UDP |

# DOMAIN NAME SYSTEM

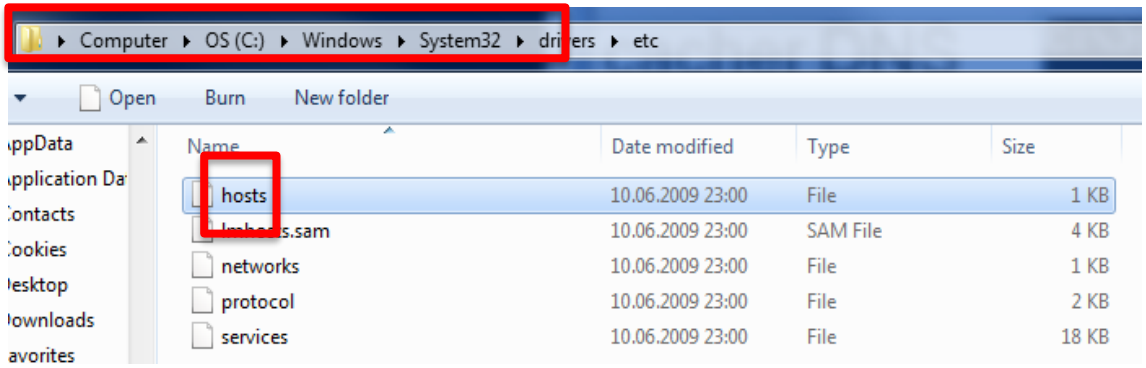
# DNS (Domain Name System)

- Mennesker
  - Navn, **person-nummer**
  - ”Ola Nordmann”, **161165 42796**
- Internett
    - **IP-adresse**, Navn
    - **54.221.226.116**, www2.nith.no
- Løser dette på Internett med DNS
    - Distribuert «database» på mange navne-tjenere
    - Protokoll i applikasjons-laget for å knytte navn og IP-adresser

# Hosts (før DNS, ca 1982)




- På NT og Linux/OSX finner du en **fil** som heter `hosts`
  - `C:\Windows\System32\Drivers\etc\hosts`



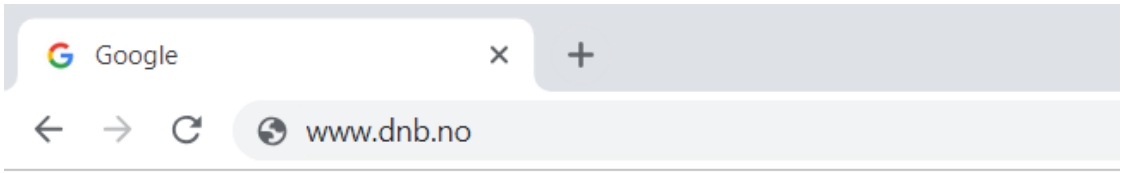

  - `Linux: /etc/hosts`
  - `OSX:/private/etc/hosts`

- Denne filen var, og benyttes fremdeles, til å oversette mellom «bokstavnavn» og IP-adresser
  - Overstyrer DNS-oppslag
  - Krever admin-/root-privilegier for å endre

# Demo av `hosts`

`195.88.54.16` `www.dnb.no`

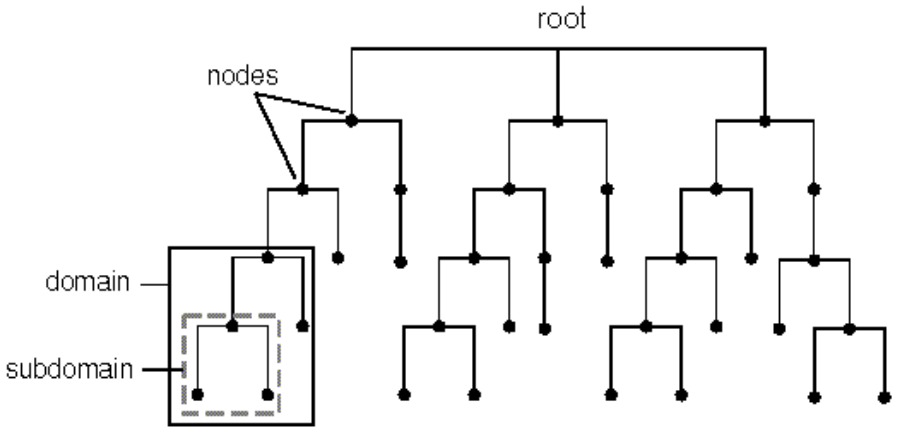
Tenk gjennom hvordan dette kan (mis-) brukes!

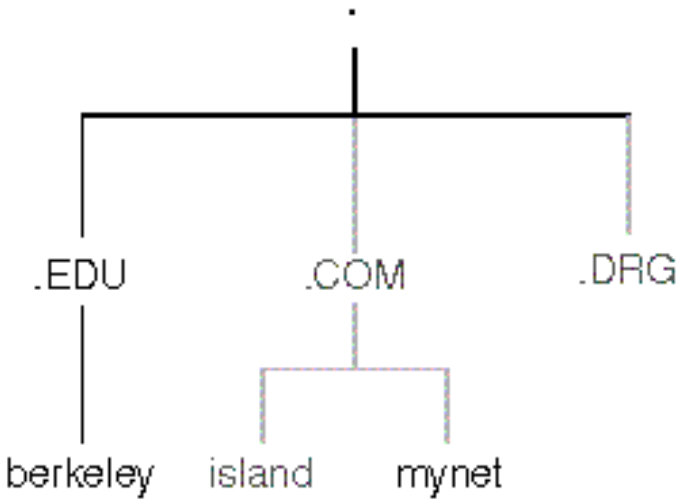



Gjør det samme mot www.nordea.no, hva er HSTS?

# DNS navne-tjenere

- Ikke sentralisert!
  - Unngå at hele nettet går ned med navne-tjeneren
  - Unngå opphopning av trafikk
  - Sentralisert database ligger alltid ”langt” vekk
  - Kan skaleres
- Navne-tjenere fordeles hierarkisk



# Hoved (root) navne-tjenere

- Kontaktes av lokale tjenere ved behov
  - Har kun oversikt over TLD (toppnivådomenene)
- 13 hoved navne-tjenere

# nslookup

```
C:\Users\blistog>nslookup
Default Server:  google-public-dns-a.goog
Address:  8.8.8.8

> set type=NS
> .
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
(root)  nameserver = m.root-servers.net
(root)  nameserver = h.root-servers.net
(root)  nameserver = b.root-servers.net
(root)  nameserver = g.root-servers.net
(root)  nameserver = l.root-servers.net
(root)  nameserver = c.root-servers.net
(root)  nameserver = a.root-servers.net
(root)  nameserver = d.root-servers.net
(root)  nameserver = i.root-servers.net
(root)  nameserver = f.root-servers.net
(root)  nameserver = k.root-servers.net
(root)  nameserver = e.root-servers.net
(root)  nameserver = j.root-servers.net
```

```
> set type=A
> i.root-servers.net
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
Name:    i.root-servers.net
Address:  192.36.148.17
```

```
> set type=NS
> com. 192.36.148.17
Server:  [192.36.148.17]
Address:  192.36.148.17

com     nameserver = g.gtld-servers.net
com     nameserver = f.gtld-servers.net
com     nameserver = b.gtld-servers.net
com     nameserver = k.gtld-servers.net
com     nameserver = l.gtld-servers.net
com     nameserver = d.gtld-servers.net
com     nameserver = m.gtld-servers.net
com     nameserver = e.gtld-servers.net
com     nameserver = c.gtld-servers.net
com     nameserver = j.gtld-servers.net
com     nameserver = i.gtld-servers.net
com     nameserver = a.gtld-servers.net
com     nameserver = h.gtld-servers.net
a.gtld-servers.net      internet address = 192.5.6.30
a.gtld-servers.net      AAAA IPv6 address = 2001:503:a83e
b.gtld-servers.net      internet address = 192.33.14.30
b.gtld-servers.net      AAAA IPv6 address = 2001:503:231d
c.gtld-servers.net      internet address = 192.26.92.30
d.gtld-servers.net      internet address = 192.31.80.30
e.gtld-servers.net      internet address = 192.12.94.30
```

# Top Level Domain (TLD-) navnetjenere

- com., no., se., uk., gov., net. osv har alle (flere) egne TLD-navnetjenere

```
C:\Users\blistog>nslookup
Default Server:  UnKnown
Address:  2001:700:2e00::4

> set type=NS
> no
Server:  UnKnown
Address:  2001:700:2e00::4

Non-authoritative answer:
no      nameserver = y.nic.no
no      nameserver = not.norid.no
no      nameserver = z.nic.no
no      nameserver = i.nic.no
no      nameserver = x.nic.no
no      nameserver = njet.norid.no

y.nic.no        internet address = 193.75.4.22
y.nic.no        AAAA IPv6 address = 2001:8c0:8200:1::2
not.norid.no    internet address = 156.154.100.12
not.norid.no    AAAA IPv6 address = 2001:502:ad09::12
z.nic.no        internet address = 158.38.8.133
```

# nslookup

```
C:\>nslookup
Default Server:  ns2.nith.no
Address:  2001:700:2e00::4

> set type=NS
> .
Server:  ns2.nith.no
Address:  2001:700:2e00::4

Non-authoritative answer:
(root)  nameserver = g.root-servers.net
(root)  nameserver = l.root-servers.net
(root)  nameserver = d.root-servers.net
(root)  nameserver = a.root-servers.net
(root)  nameserver = m.root-servers.net
(root)  nameserver = h.root-servers.net
(root)  nameserver = i.root-servers.net
(root)  nameserver = b.root-servers.net
(root)  nameserver = c.root-servers.net
(root)  nameserver = e.root-servers.net
(root)  nameserver = j.root-servers.net
(root)  nameserver = k.root-servers.net
(root)  nameserver = f.root-servers.net
> no.
Server:  ns2.nith.no
Address:  2001:700:2e00::4

Non-authoritative answer:
no      nameserver = z.nic.no
no      nameserver = i.nic.no
no      nameserver = njet.norid.no
no      nameserver = x.nic.no
no      nameserver = y.nic.no
no      nameserver = not.norid.no

i.nic.no        internet address = 194.146.106.6
x.nic.no        internet address = 128.39.8.40
y.nic.no        internet address = 193.75.4.22
y.nic.no        AAAA IPv6 address = 2001:8c0:8200:1::2
not.norid.no    internet address = 156.154.100.12
not.norid.no    AAAA IPv6 address = 2001:502:ad09::12
njet.norid.no   internet address = 156.154.101.12
> _
```

- . er rot-navntjenerne
- no. er no-domenets TLD-navnetjener
  - Det er disse vi må spørre dersom vi vil vite hva som er **navnetjener** for kristiania.no, google.no, osv

# Autoritative navnetjenere

- Det er sone-filene som kopler sammen ulike typer IP-adresser og DNS-navnene deres
- F.eks.
  - `nith.no` domenet har autoritative navnetjenere:

```
> set type=NS
> nith.no
Server:  eks-dns02.ad.nith.no
Address:  2001:700:2e00::4

nith.no nameserver = nn.uninett.no
nith.no nameserver = eks-dns01.ad.nith.no
nith.no nameserver = eks-dns02.ad.nith.no
nn.uninett.no   internet address = 158.38.0.181
nn.uninett.no   AAAA IPv6 address = 2001:700:0:503::aa:5302
eks-dns01.ad.nith.no    internet address = 158.36.131.3
eks-dns01.ad.nith.no    AAAA IPv6 address = 2001:700:2e00::3
eks-dns02.ad.nith.no    internet address = 158.36.131.4
eks-dns02.ad.nith.no    AAAA IPv6 address = 2001:700:2e00::4
```

# Autoritativ navnetjener

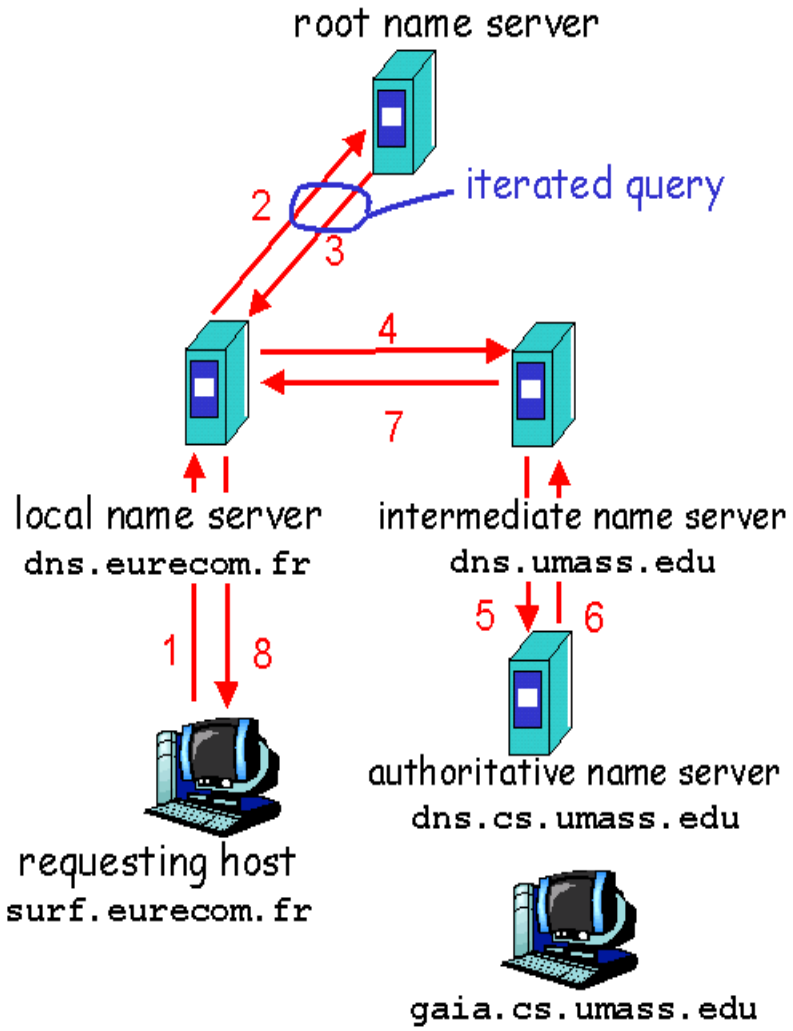
- Den navnetjeneren der sonefilene med navn og IP-adresser befinner seg er autoritativ for et domene

```
C:\>nslookup
Default Server:  ns2.nith.no
Address:  158.36.131.4

> home.nith.no
Server:  ns2.nith.no
Address:  158.36.131.4

Name:    nih-stud-web02.osl.basefarm.n
Address:  79.171.82.156
Aliases:  home.nith.no

> vg.no
Server:  ns2.nith.no
Address:  158.36.131.4

Non-authoritative answer:
Name:    vg.no
Addresses:  2a02:c0:1010::16
          2a00:1b60:1010::16
          195.88.54.16
          195.88.55.16
```

```
> set type=NS
> vg.no
Server:  ns2.nith.no
Address:  158.36.131.4

Non-authoritative answer:
vg.no    nameserver = ns-foo.linpro.net
vg.no    nameserver = ns-zoo.linpro.net
vg.no    nameserver = ns-bar.linpro.net

ns-foo.linpro.net       internet address = 87.238.32.254
ns-foo.linpro.net       AAAA IPv6 address = 2a02:c0:1001:1::2
ns-zoo.linpro.net       internet address = 67.18.176.124
> set type=A
> vg.no ns-zoo.linpro.net
Server:  ns-zoo.linpro.net
Address:  67.18.176.124

Name:    vg.no
Addresses:  195.88.55.16
          195.88.54.16
```

# Gjentatte spørringer

- Vanligvis er spørringene rekursive
  - A spør på vegne av B og returnerer svaret til B
  - til lokal (autoritativ) navnetjener
  - Typisk fra bruker
- Spørringene kan også være iterative
  - A spør på vegne av B og returnerer neste tjeners adresse til A, som deretter spør denne selv
  - typisk fra lokal navntjener til rot-, TLD (Top Level Domain) og andre lokale navntjenere



# Caching og oppdatering

- Navne-tjenere cacher DNS kartlegging
- Lagring i cache forsvinner etter en tid (TTL timeout)
- DNS resolveren på eget OS cacher også..

- Mekanismer for innmelding og oppdatering er under utvikling hos IETF (Internet Engineering Task Force)
  - Dynamisk oppdatering
  - Sikkerhet
  - mmm

```
C:\>ipconfig /flushdns

Windows IP Configuration

Successfully flushed the DNS Resolver Cache.

C:\>ipconfig /displaydns

Windows IP Configuration

    www.google.no
    ----------------------------------------
    Record Name . . . . . : www.google.no
    Record Type . . . . . : 5
    Time To Live  . . . . : 73
    Data Length . . . . . : 8
    Section . . . . . : Answer
    CNAME Record  . . . . : www.google.com


    maps.google.no
    ----------------------------------------
    Record Name . . . . . : maps.google.no
    Record Type . . . . . : 5
    Time To Live  . . . . : 31
    Data Length . . . . . : 8
    Section . . . . . : Answer
    CNAME Record  . . . . : maps.google.com
```

# DNS records

Distribuert database lagrer RR (resource records)

RR format: navn, verdi, type, ttl

- Type=A
  - Navn=vertsnavn, verdi=IPv4-adresse
  - **AAAA**-typen er IPv6-adresser
- Type=NS
  - Navn=domene, verdi=IP-adresse til navne-tjener
- Type=CNAME
  - Navn=alias, verdi=virkelig navn
- Type=MX
  - Navn=alias, verdi=post tjener

# DNS Records: MX

- MX-oppslag utføres for å finner hvilken SMTP-tjener epost skal sendes til:

```
> set type=MX
> westerdals.no
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
westerdals.no   MX preference = 1, mail exchanger = aspmx.l.google.com
westerdals.no   MX preference = 10, mail exchanger = aspmx5
westerdals.no   MX preference = 5, mail exchanger = alt1.as
westerdals.no   MX preference = 10, mail exchanger = aspmx2
westerdals.no   MX preference = 5, mail exchanger = alt2.as
westerdals.no   MX preference = 10, mail exchanger = aspmx3
westerdals.no   MX preference = 10, mail exchanger = aspmx4
```

```
> set type=A
> aspmx.l.google.com
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
Name:    aspmx.l.google.com
Address:  64.233.162.27
```

- Når noen skal sende epost til bengt@westerdals.no må MX-oppslag for westerdals.no foregå i forkant.

# DNS-records: A, AAAA, PTR

- PTR-records benyttes for å finne navnet som tilhører en bestemt IP-adresse

```
51.131.36.158.in-addr.arpa PTR test.kristiania.no.
```

- A-records kopler navn med IPv4

```
test.kristiania.no. A 158.36.131.51
```

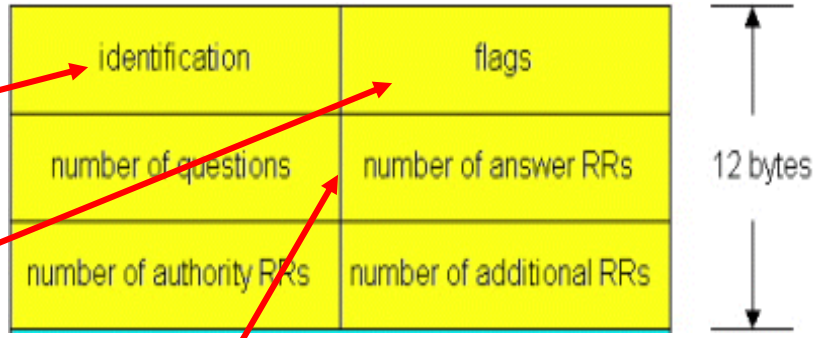
- AAAA-records kopler navn med IPv6

```
test.kristiania.no. AAAA 2001:700:2e00::51
```

- CNAME
  - Lar samme IP-adresse tilsvare flere ulike navn under domenet

# DNS: Headerformat

- Spørring og svar har samme format
- Header
  - Identifisering
    - 16 bit nummer
    - Samme i spørring og svar
  - Flagg
    - Spørring eller svar
    - Rekursjon ønsket
    - Rekursjon tilgjengelig
    - Autoritativ tjener
  - Antall spørringer og svar



# Norske bokstaver?

- For å støtte andre bokstav-sett enn ASCII benytter DNS nå *Punycode*: `øl.no` er i sonefilen notert `xn--l-4ga.no`:

```
xn--l-4ga.no
----------------------------------------
Record Name . . . . . : xn--l-4ga.no
Record Type . . . . . : 1
Time To Live  . . . . : 71852
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 83.143.81.86
```

- **Punycode** (RFC 3492) er ikke pensum, men greit å vite at finnes..

# E-POST

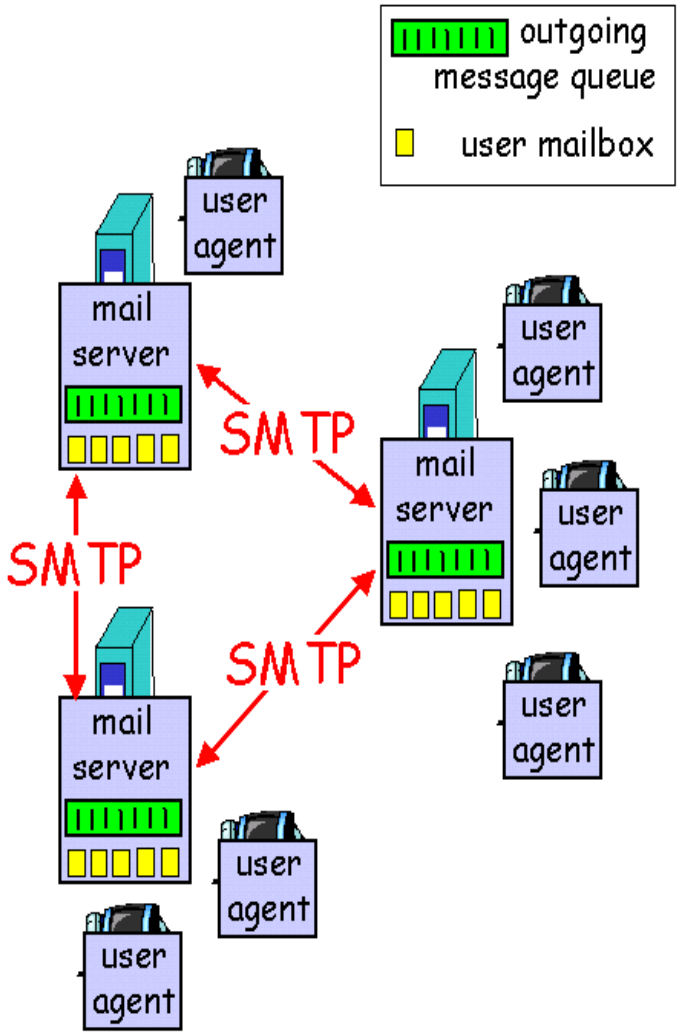
**RFC 821**
RFC 974
RFC 1123
RFC 1869
RFC 2821

# Elektronisk post

- Tre hovedkomponenter
  - Bruker agent
  - Post tjener (SMTP-tjener)
  - SMTP (Simple Mail Transfer Protocol)

- Bruker agent
  - Eget program (mail reader)
  - Les, skriv, sett sammen mail
  - Post lagres i utgangspunktet på tjeneren og hentes med POP3 eller IMAP
  - Eudora, Outlook, Messenger

  - (Etter hvert) svært vanlig å bruke web-grensesnitt.



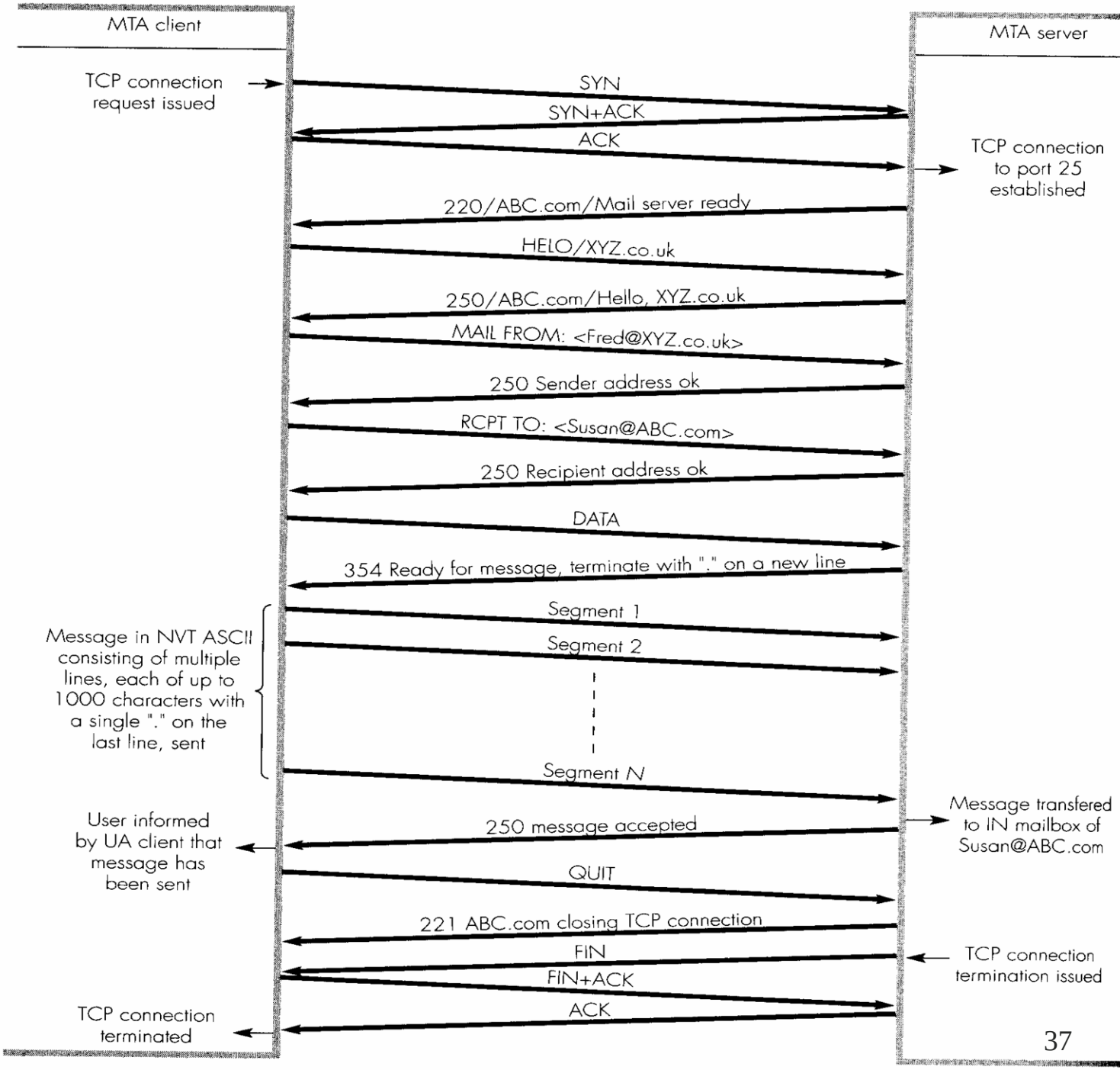
# Post tjener

- Postboks (hus) som inneholder ulest post for brukeren
- Kø for post som skal sendes
- SMTP protokoll mellom tjenere for å sende posten
- Klienten virker som en avsender-tjener til SMTP-tjeneren den er satt opp til å bruke.

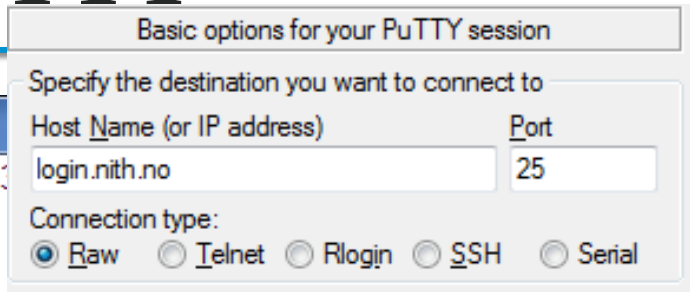
- For å motta epost brukes f.eks. POP3 eller IMAP

# Simple Mail Transfer Protocol

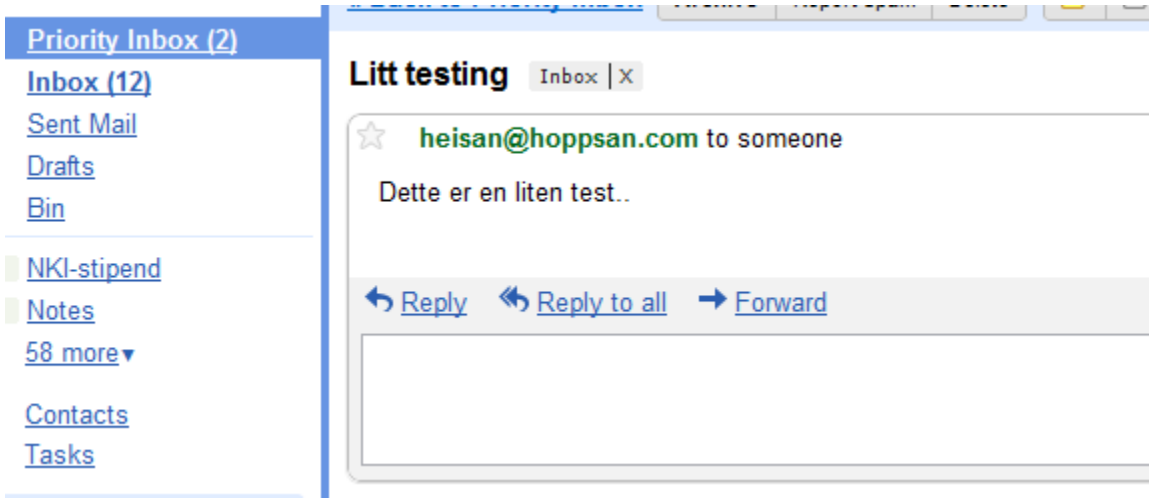
- Bruker TCP for å overføre post fra klient til tjener, port 25
  - Port 587
- Direkte overføring fra tjener til tjener
- 3 overføringsfaser
  - Handshake
  - Overføring
  - Avslutning
- Overføring i ASCII tekst
  - Kommandoer og statuskoder
- Meldingsdelen er i 7-bits ASCII


MTA client
MTA server
TCP connection request issued
SYN
SYN+ACK
ACK
TCP connection to port 25 established
220/ABC.com/Mail server ready
HELO/XYZ.co.uk
250/ABC.com/Hello, XYZ.co.uk
MAIL FROM: <Fred@XYZ.co.uk>
250 Sender address ok
RCPT TO: <Susan@ABC.com>
250 Recipient address ok
DATA
354 Ready for message, terminate with "." on a new line
Message in NVT ASCII consisting of multiple lines, each of up to 1000 characters with a single "." on the last line, sent
Segment 1
Segment 2
Segment N
Message transfered to IN mailbox of Susan@ABC.com
User informed by UA client that message has been sent
250 message accepted
QUIT
221 ABC.com closing TCP connection
FIN
TCP connection termination issued
FIN+ACK
ACK
TCP connection terminated

# Eksempel (2010) - PuTTY




```
login.nith.no - PuTTY
220 login.nith.no ESMTP Exim 4.69 Tue, 08 Feb 2011 13:1
HELO nith.no
250 login.nith.no Hello blistog.nith.no [158.36.131.51]
MAIL FROM: blistog@nith.no
250 OK
RCPT TO: blistog@nith.no
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
To: someone@somewhere.else.no
From: heisan@hoppsan.com
Subject: Litt testing

Dette er en liten test..
.
250 OK id=1PmmTU-0000oK-Ea
```

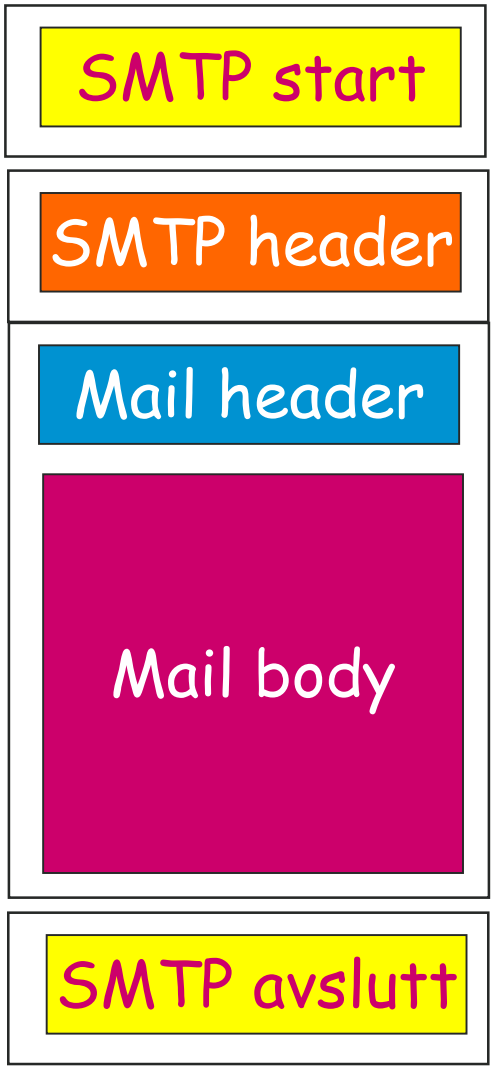


# SMTP kontra HTTP

- SMTP bruker vedholdende forbindelse
- Noen karakterstrenger er ulovlige i meldinger
- Alt går i ASCII kode
  - Bl.a derfor omkodes meldingen (base64,Uuencode, hex64 ...)
  - CRLF
    .
    CRLF
    avslutter en melding
- HTTP henter data (pull), epost skyver data (push)
- HTTP overfører (vanligvis) ett objekt pr melding
- SMTP kan overføre mange (omkodet til ASCII) objekter pr enkeltmelding («vedlegg»)

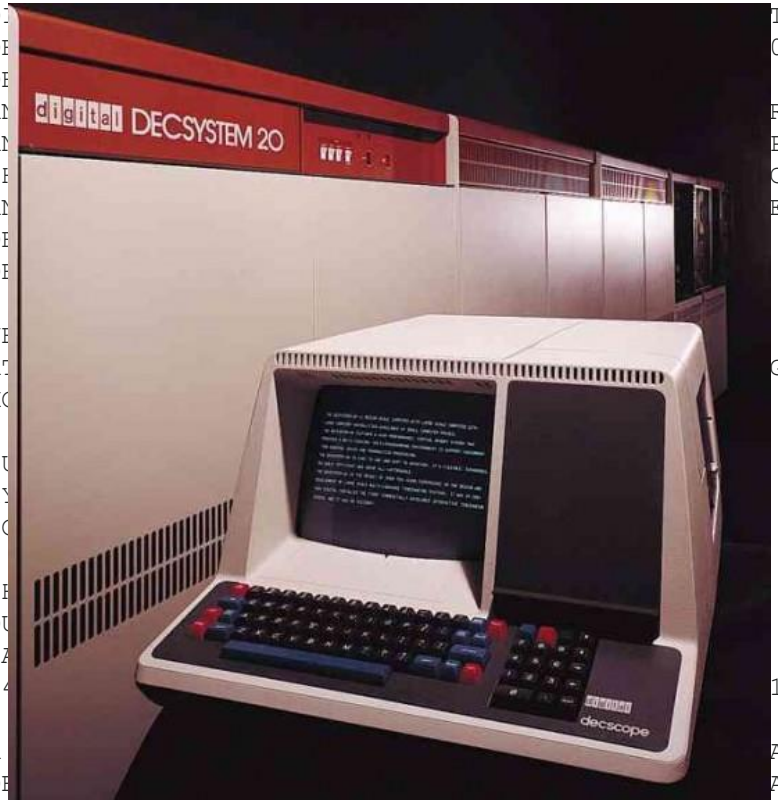
# Post format

- SMTP konversasjon
  - HELO
- SMTP header
  - MAIL FROM:
  - RCPT TO:
  - DATA
- Mail header
  - From:
  - To:
  - Subject:
  - Dette er **ikke SMTP** kommandoene!
- Blank linje (To CRLF)
- Body
  - Bare 7 bit ASCII tekst
  - Sendes med ett punktum på starten av en linje fulgt av linjeskift
- QUIT



# SPAM


# 3. Mai 1978:




D THE NEWEST MEMB
D 060, AND 2060T.
D THE TENEX OPER
A RE. BOTH THE DE
A PS-20 OPERATING
T CURRENT DECSYST
A END MEMBER OF THE
D WITH ALL OF THE OTHER
D

W THE DECSYSTEM-20 FAMILY
A G IN CALIFORNIA THIS
M

T
H
L

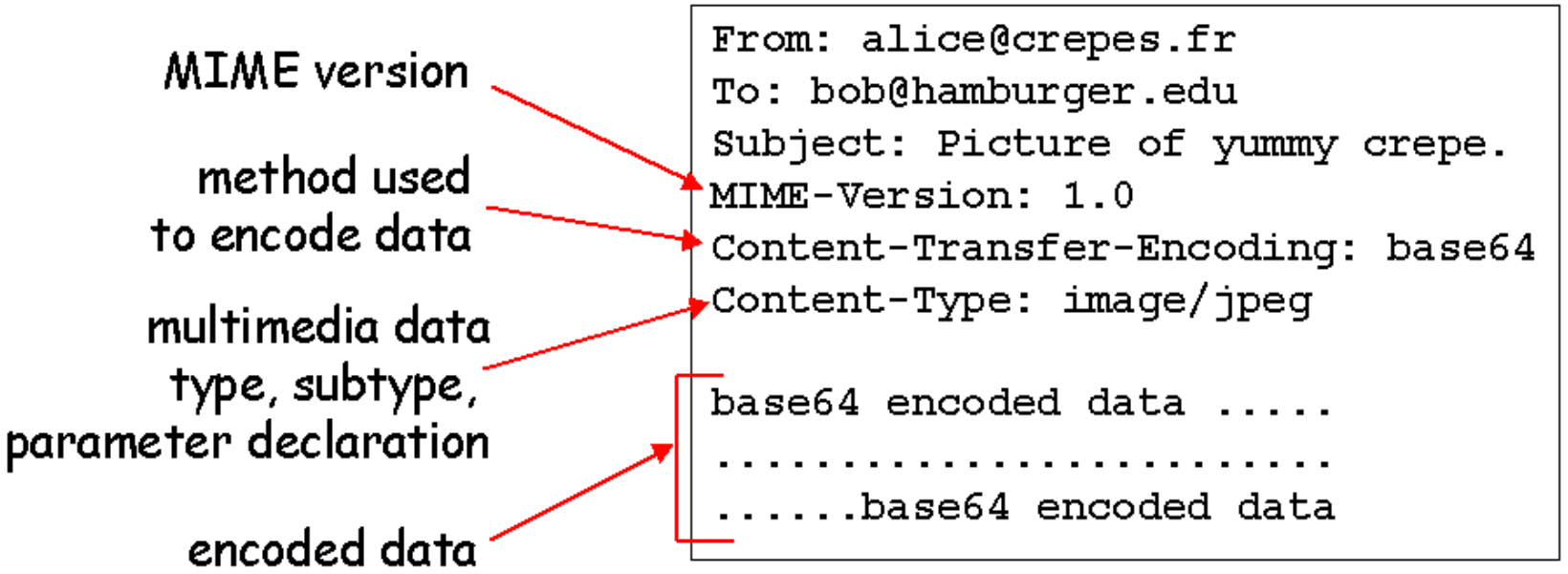
T
D
S
( 1 AND RT 92)

A ALS ON-LINE TO OTHER
D ARE UNABLE TO ATTEND,
PLEASE FEEL FREE TO CONTACT THE NEAREST DEC OFFICE
FOR MORE INFORMATION ABOUT THE EXCITING DECSYSTEM-20 FAMILY.

Gary Thuerk

# MIME (Multipurpose Internet Mail Extensions)

- Tilleggslinjer i header for MIME innhold
  - Tekst, bilde, audio, video, applikasjon



Westerdals
Oslo A.C.T
Avdeling for Teknologi

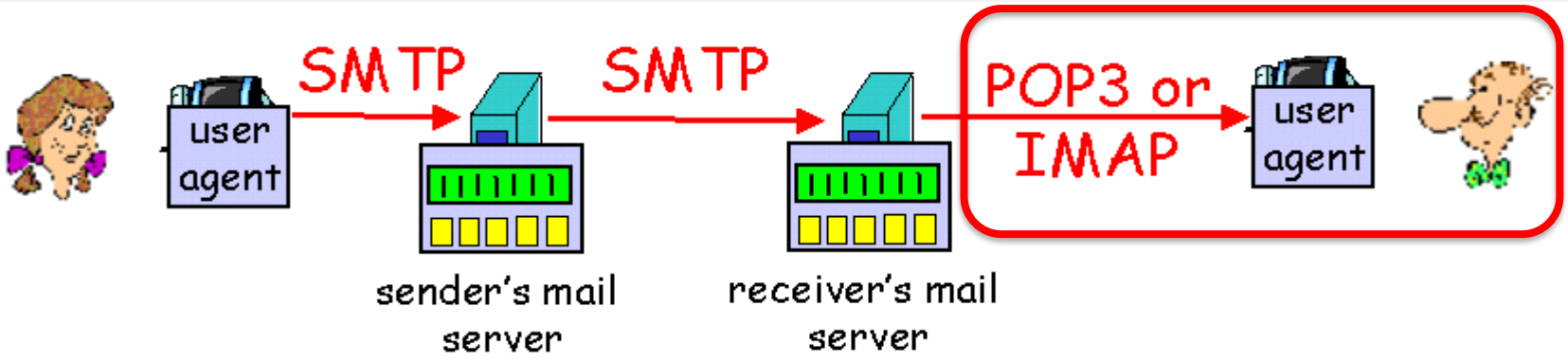
# MIME: multipart

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
--98766789--
```

# Eksempel: gmail

- Subject-feltet sendes som UTF-8 omkodet til base64
- Bokstavene sendes som ASCII-representasjon av de numeriske verdiene i UTF-8, innledet med =-tegn..

# Post tilgangs-protokoller




- POP3 (Post Office Protocol) #2!
  - Enkel protokoll, autorisasjon og overføring til klient
- IMAP (Internet Message Access Protocol)
  - Kompleks protokoll, håndteringsmulighet på tjener
- HTTP
  - Mail på web, Hotmail (1995) .....

# POP3 protokoll

- Autorisasjons-fase
  - user: Bruker ID
  - pass: Passord
- Transaksjons-fase
  - list: Liste av meldings-nummer
  - retr: Hent meldings-nummer
  - dele: Fjern meldings-nummer
  - quit: Avslutt

```
S: +OK POP3 server ready
C: user alice
S: +OK
C: pass hungry
S: +OK user successfully logged
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

Westerdals
Oslo A.C.T
Avdeling for Teknologi

# Eksempel (POP3)

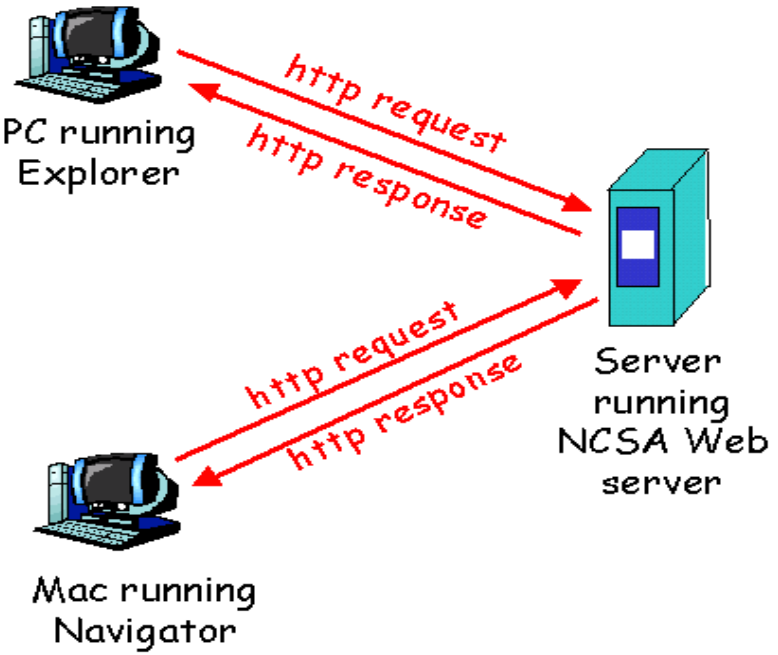
```
S: +OK mail.oslo.dph.no POP3 server (Netscape Messaging Server -
   Version 3.6) ready Wed, 5 Feb 2006 15:42:30 +0100
C: user blistog
S: +OK Password required for blistog
C: pass passord
S: +OK blistog's maildrop has 206 messages (17625186 octets)
C: retr 206
S: +OK 366 octets
S: Status: U
S: Return-Path: <bol@dill.no>
S: Received: from [10.21.11.65] by mail.oslo.dph.no (Netscape
   Messaging Server 3.6)  with SMTP id AAA5D for
   <blistog@nith.no>; Wed, 5 Feb 2006 15:41:54 +0100
S: Date: Wed, 5 Feb 2006 15:41:54 +0100
S: Message-ID: <7763AAE2100D.AAA5D@mail.oslo.dph.no>
S: From: <bol@dill.no>
S:
S: Hei!
S: Dette er ikke noe å spare på!
S: .
C: quit
S: +OK mail.oslo.dph.no POP3 server closing connection
```

http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html

# HyperText Transfer Protocol

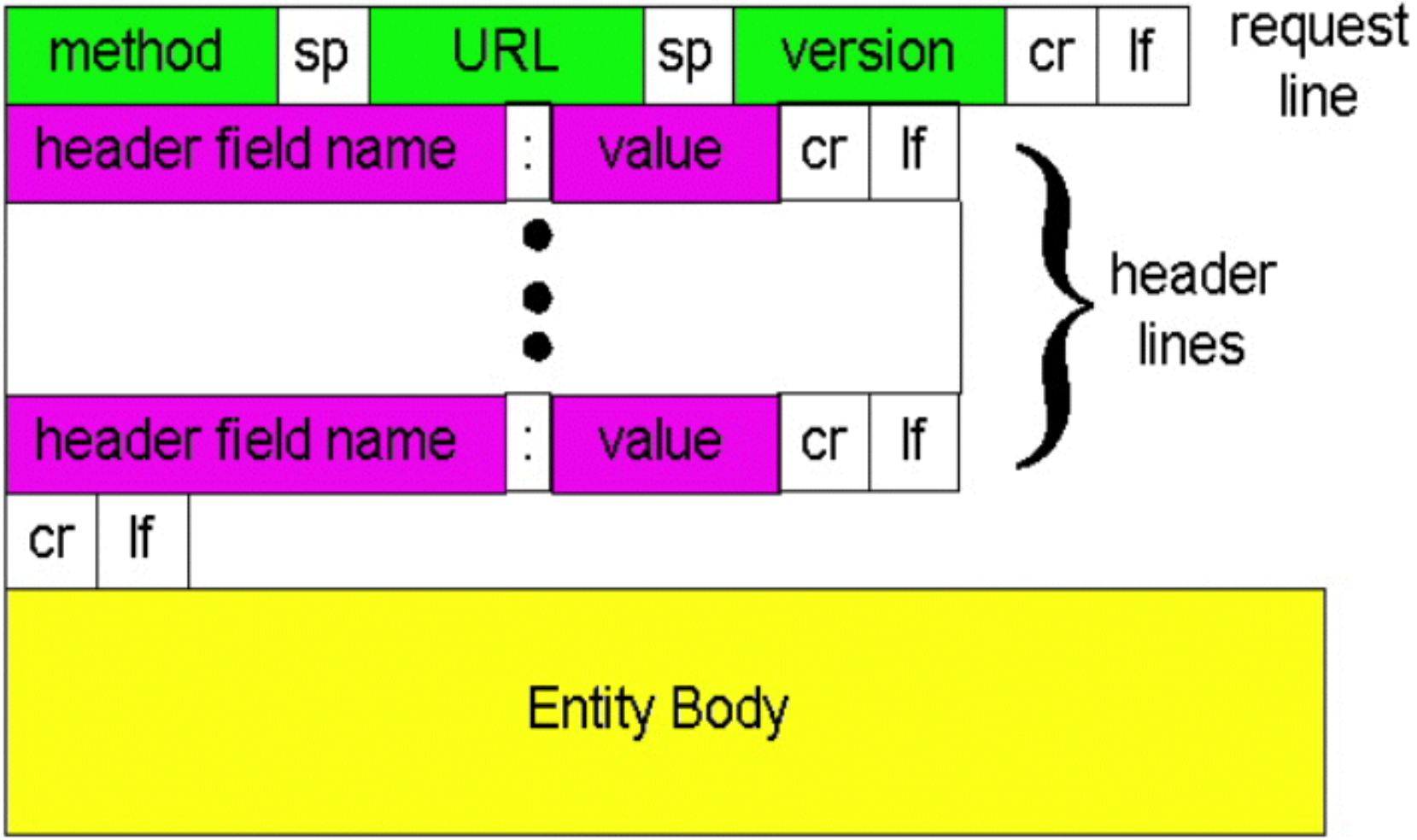
# HTTP (HyperText Transfer Protocol)

- Webens applikasjons-protokoll
  - En *enkel* filoverføringsprotokoll…
- Klient/tjener modell
  - Klienten spør etter, mottar og viser web ”objekter”
  - Tjeneren sender objekter på etterspørsel

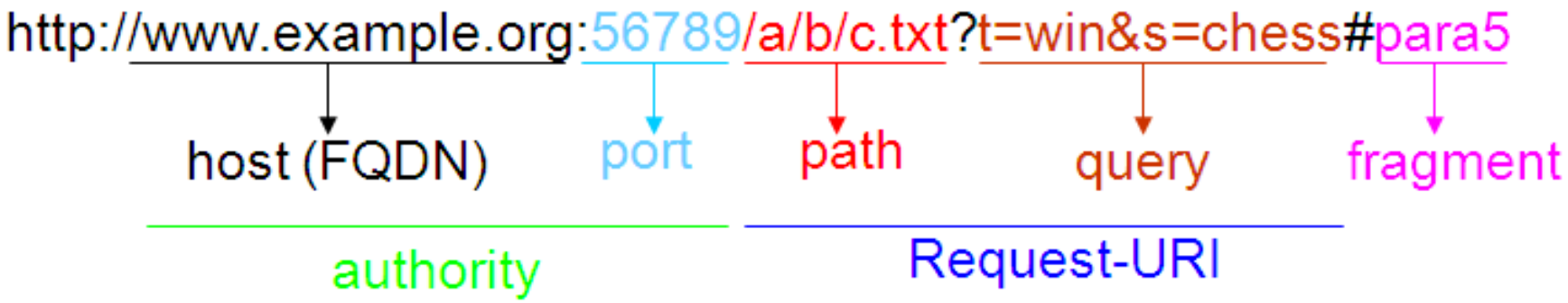


# HTTP meldingsformat: spørring

- Meldingsheaderen er kodet i 7 bit ASCII-format



# HTTP URL




- Browseren foretar et DNS-oppslag og oppretter en TCP-forbindelse til ”authority”.
- Så følger ”filsti” på server (ressurs-ID)
- Etter ? Følger argumenter til script/program
- Etter # typisk et anker/posisjon innenfor ressurs (”dokument”)

Westerdals
Oslo A.C.T
Avdeling for Teknologi

# Typer metoder

## HTTP/1.0

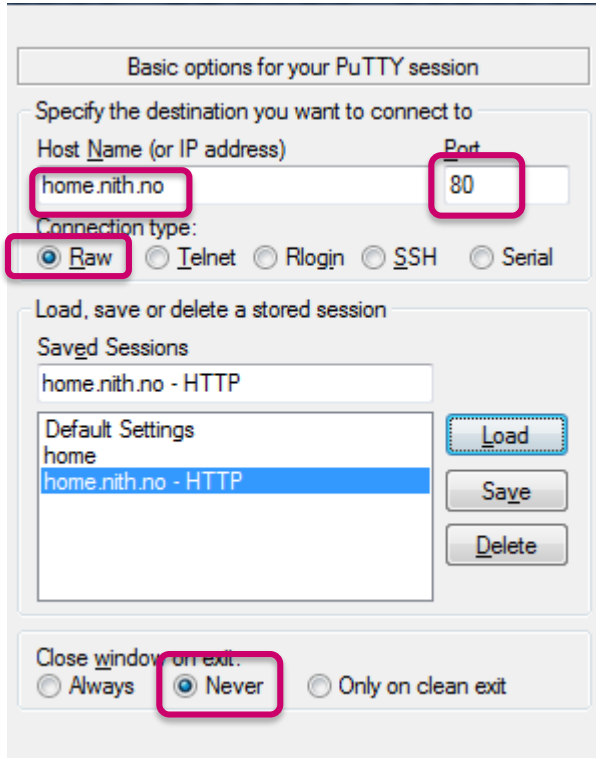
- GET
- POST
- HEAD
  - Spør bare server om metainformasjon = headere

## HTTP/1.1

- GET, POST, HEAD
- PUT
  - Laster opp en fil til adressen som er spesifisert i URL-feltet
- DELETE
  - Sletter filen som er spesifisert i URL-feltet
- OPTION
- TRACE

# «Manuell» spørring 2 (Windows 7)

- Kan bruke PuTTY til å opprette en TCP-forbindelse mot web-tjeneren (port 80)




```
GET /~blistog/index.html HTTP/1.1
Host: home.nith.no
```

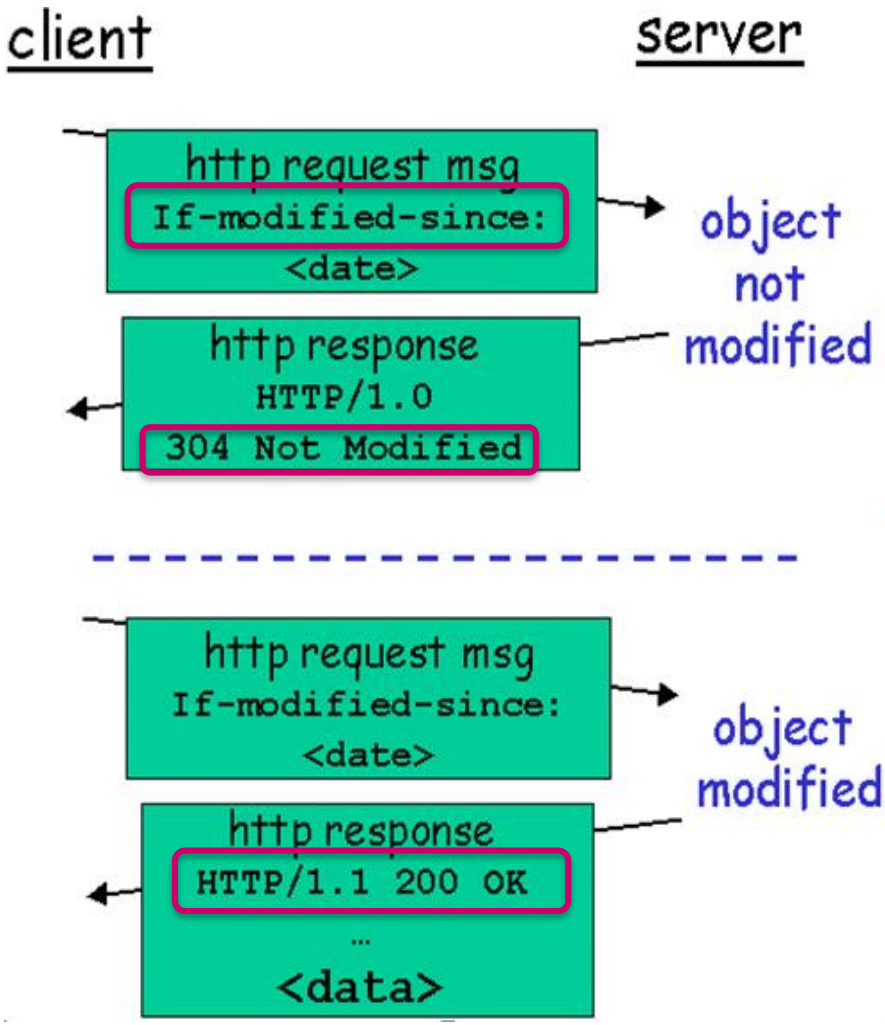
Spørring

```
HTTP/1.1 200 OK
Date: Mon, 31 Jan 2011 15:42:28 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Wed, 24 Nov 2010 07:29:12 GMT
ETag: "70043a-923-495c772174200"
Accept-Ranges: bytes
Content-Length: 2339
Cache-Control: no-stor , no cache, mus revalidate,
ax-age=0
Connection: close
Content-Type: text/html; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
```

SVAR

# Klient-tjener kommunikasjon (Ex)

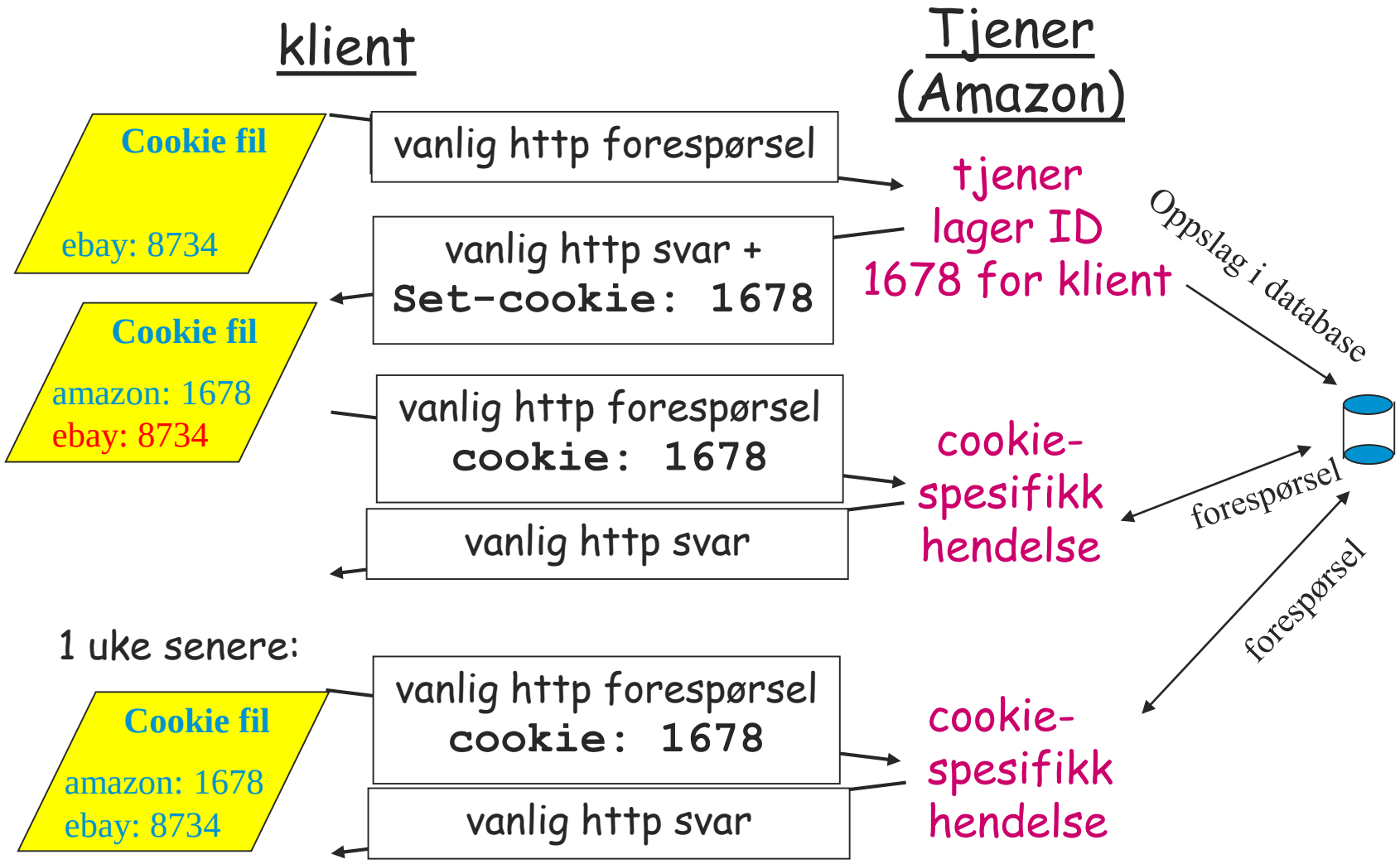
- Betinget GET
  - If-modified-since:
  - Ikke send svar hvis klient har oppdatert versjon
  - Sjekker tidsstempelet på filen
- Klient
  - Spesifiser dato for **cachet** fil
- Tjener
  - Statuskode 304 dersom ikke oppdatert
  - Dersom endret kommer en vanlig «200 OK» og det oppdaterte filinnholdet



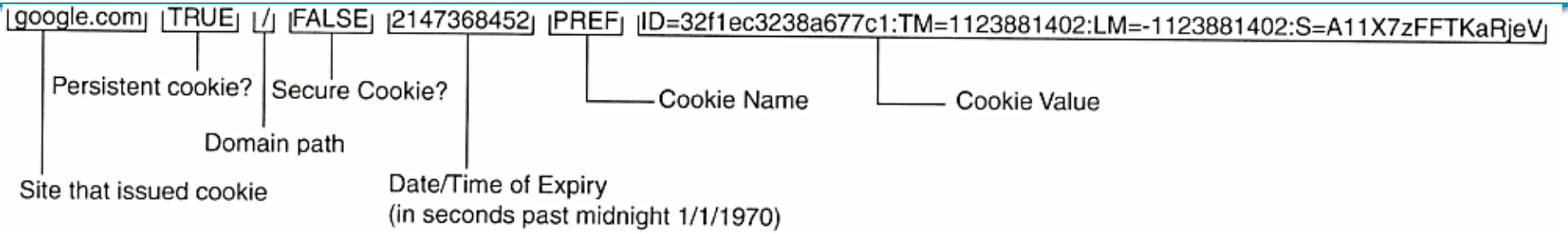
# Beholde tilstanden med cookie

- Mange Web-steder benytter cookies
- En cookie har «4» hoved-elementer
  - Cookie header linje i http-responsen
  - Cookie header linje i http-forespørselen
  - Cookie(-fil) som kan ligge hos klienten
  - «Database» over cookies hos tjeneren
- Cookie kan
  - Bevare tilstand
  - ”Huske” autorisasjoner og settinger

# Beholde tilstanden med cookie (1)

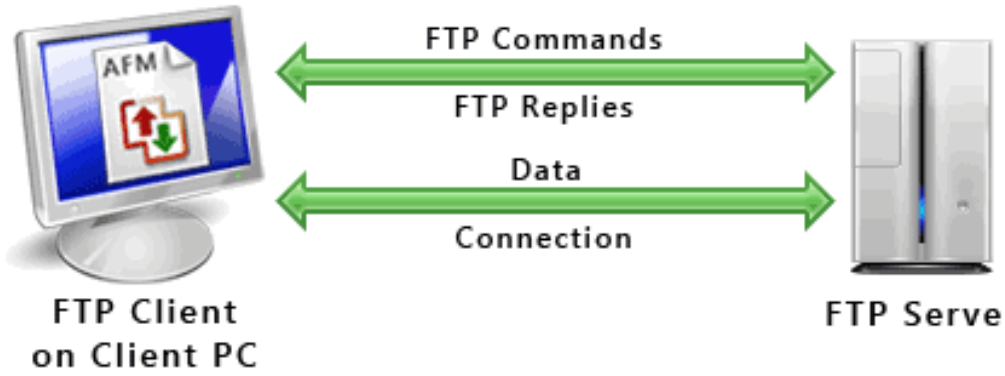


# Cookies (2)



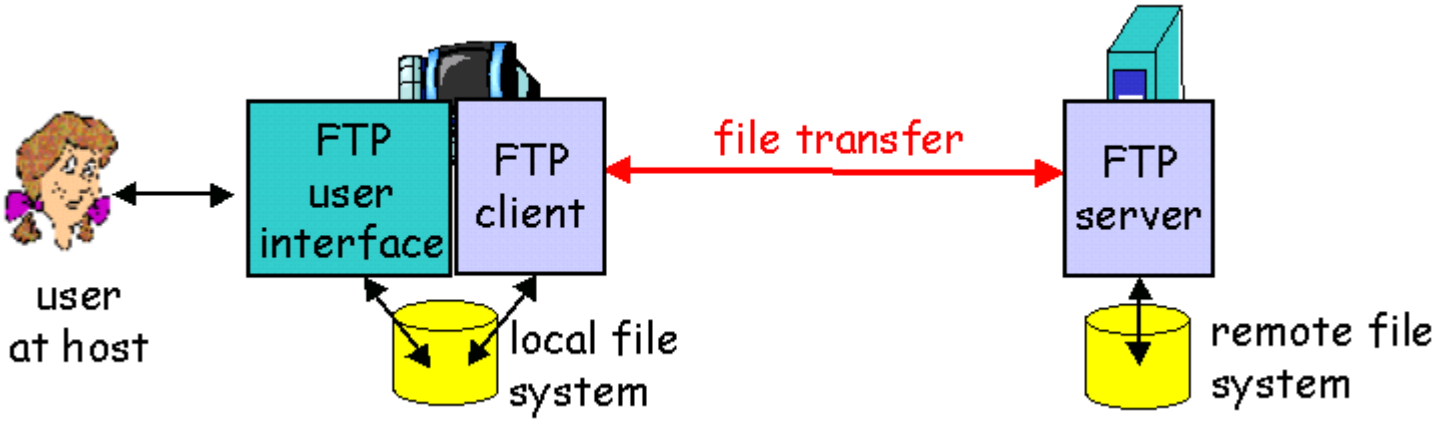

- Cookies kan være persistente eller ikke-persistente
  - Persistente lagres på klient-maskin frem til utløpsdatoen
  - Ikke-persistente brukes kun i den opprettede sesjonen og slettes når browser avsluttes.
- Cookies kan være sikre eller usikre
  - Sikre cookies sendes kun over HTTPS (SSL/TLS)
- Ulike browsere lagrer persistente cookies i proprietære format
  - Eksempelet over er Firefox, IE lagrer i separate txt-filer, Chrome i SQLite database…

# FILE TRANSFER PROTOCOL




RFC 959

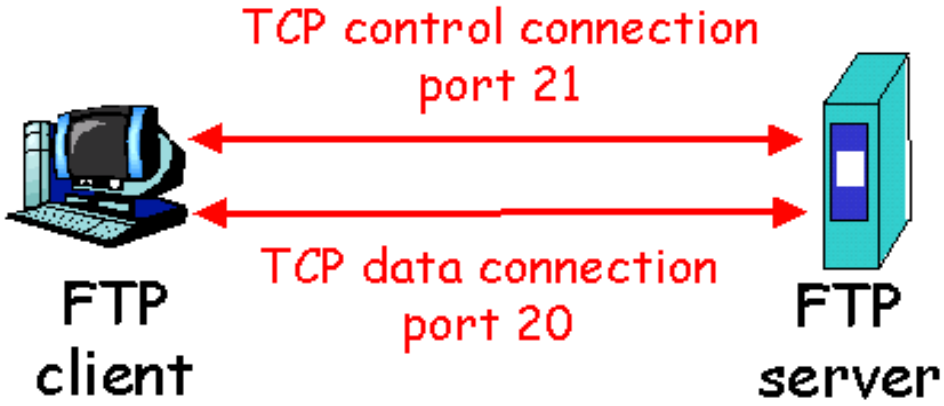
# FTP (File Transfer Protocol)




- Overføring av filer
- Bruker klient/tjener modell
- Rask overføring
- *Kan* bruke browser som grensesnitt

# FTP dataforbindelse

- Klienten kontakter tjener på port 21 med TCP




- To parallelle forbindelser åpnes
  - Kontroll (port 21): Overfører kommandoer og svar
  - Data (20): Overfører data
- FTP er ikke tilstandsløs («stateless»)
  - Klient og tjener har en **delt** «forståelse» er hva som skal gjøres i hvilken rekkefølge, og hvor langt man har kommet.
  - «Husker» f.eks. ID/passord og gyldig mappe
- RFC 959

# FTP kommandoer og returkoder

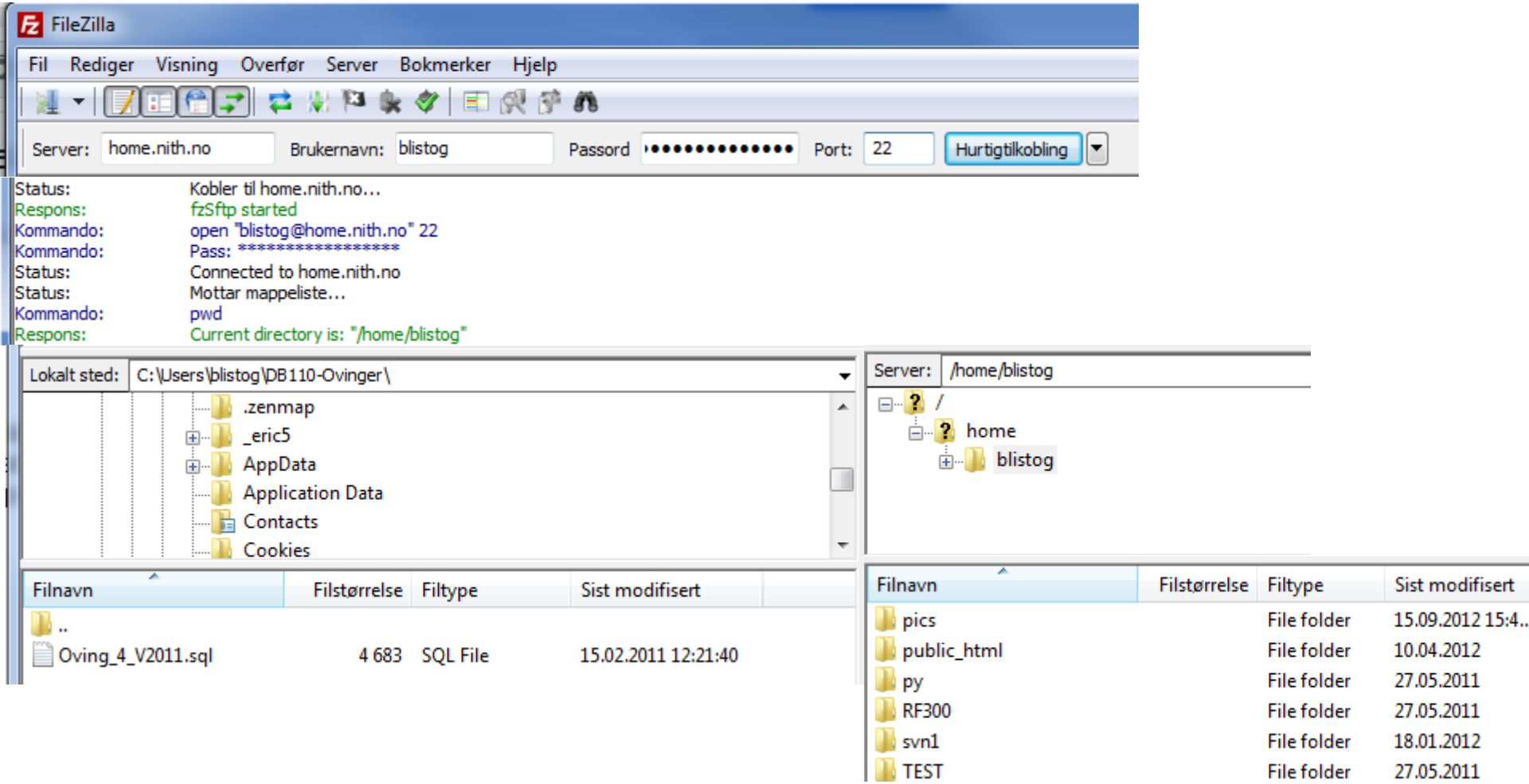
- Sendes som ASCII tekst
- Kommandoer
  - USER *brukernavn,* PASS *passord*
  - LIST gir liste av filer i mappen
  - RETR *filnavn*, STOR *filnavn* henter/lagrer fil
- Returkoder ligner på HTTP
  - 125 Data connection already open; transfer starting
  - 331 Username OK, password required
  - 425 Can’t open data connection
  - 452 Error writing file

# Sikkerhet, HTTP og FTP

- SFTP er ikke en ny versjon av FTP, men en helt ny protokoll.
  - Vanligvis benytter den SSH (Secure Shell) til å opprette en kryptert forbindelse til en **sftp-filserver.**
  - Standardport 22
- HTTPS er vanlig HTTP over en kryptert transportlagsforbindelse
  - benytter vanligvis TLS/SSL-protokollene for å lage en kryptert TCP-forbindelse
  - Standard port 443.

# sftp mot et hjemmeområde

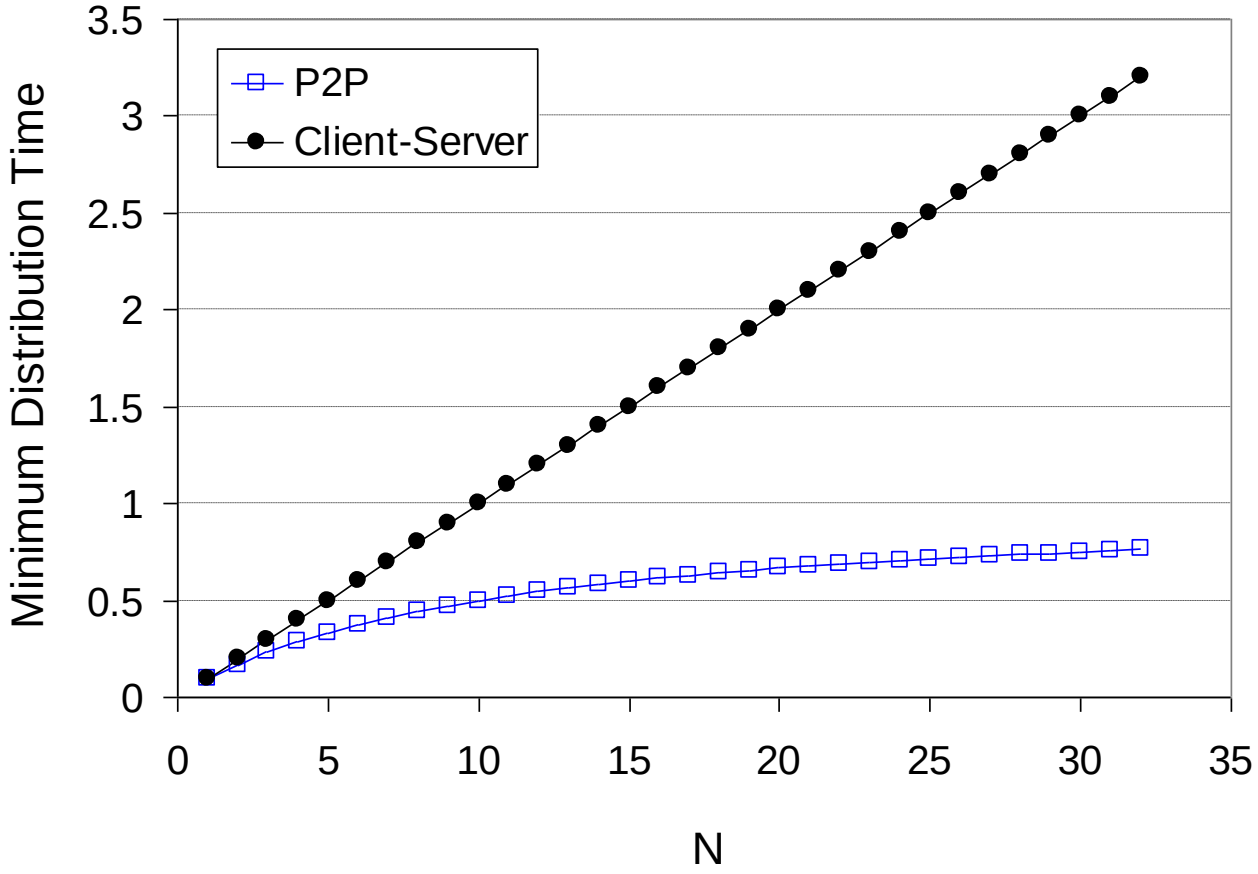
- Filezilla er en enkel og grei GUI-basert sftp-klient med drag'n'drop av filer fra lokal disk til server



«Om vi får tid så ser vi også litt på…»

# P2P: TORRENT

# Klient-tjener vs. P2P

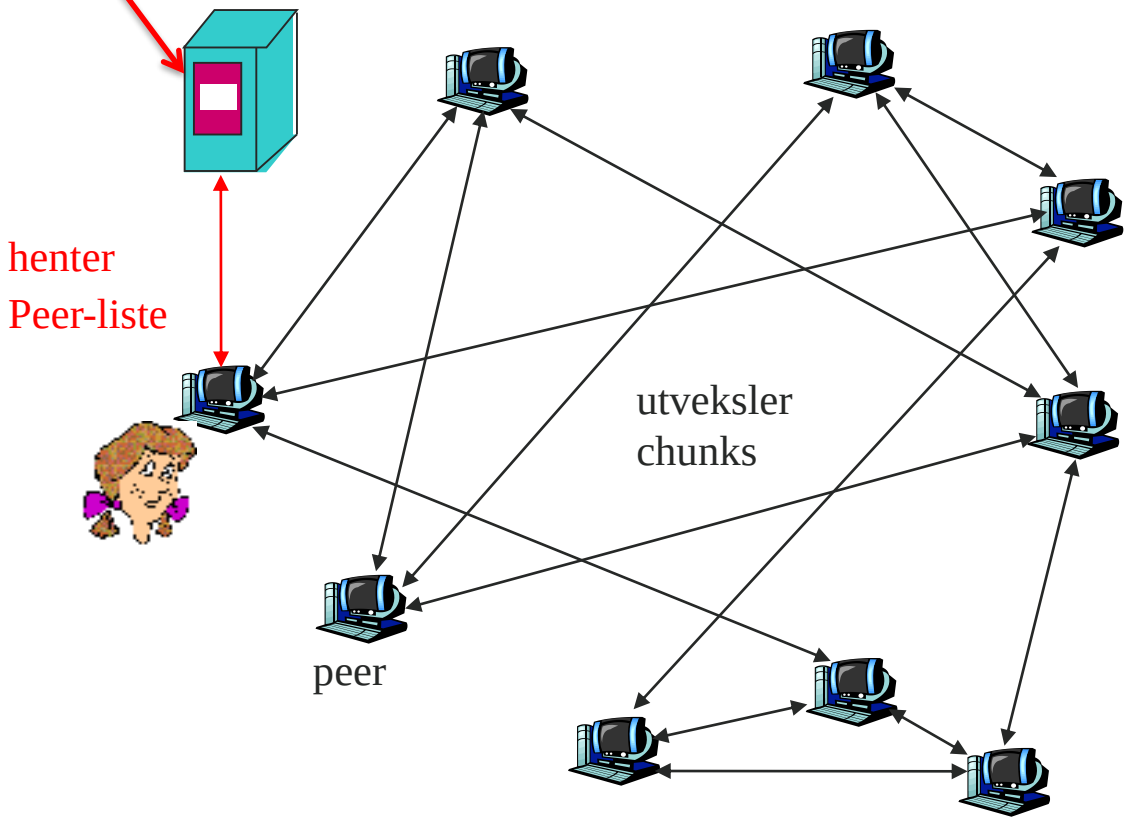



*P2P kan redusere (samlet) nedlastingstiden da det "fjerner"* ***flaskehalsen inn til tjeneren***

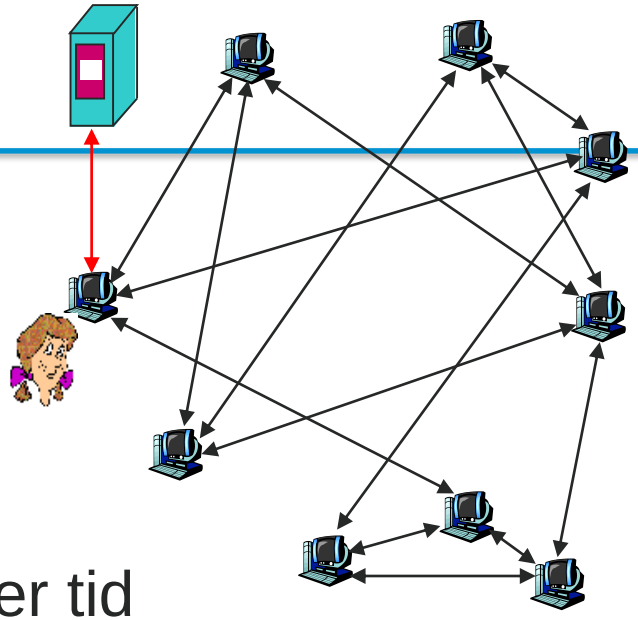
# Fildistribusjon: BitTorrent

*tracker:* overvåker (tracks) hvilke "peers" som deltar i en torrent

*torrent:* gruppe av "peers" som utveksler "chunks" av en fil



# BitTorrent (1)



- Filen deles opp i  256KiB store *chunks*.
- peer meldes inn i torrent:
  - Har ingen chunks, men skaffer seg over tid
  - Registreres hos tracker for å få liste over peers, oppretter forbindelse til en undermengde av peers (“naboer”)
- Både laster opp og ned chunks til andre peers.
  - Prioriterer chunks som er “sjeldenest”
- Peers kommer og forsvinner som de vil.

# BitTorrent (2)

## Hente Chunks

- Ulike peers har ulike deler av filen på et gitt tidspunkt
- periodisk, spør en peer (Anne) naboene om hvilke chunks de har
- Anne etterspør så de manglende delene ("chunks"
  - sjeldneste først)

## Levere Chunks: "tit-for-tat"

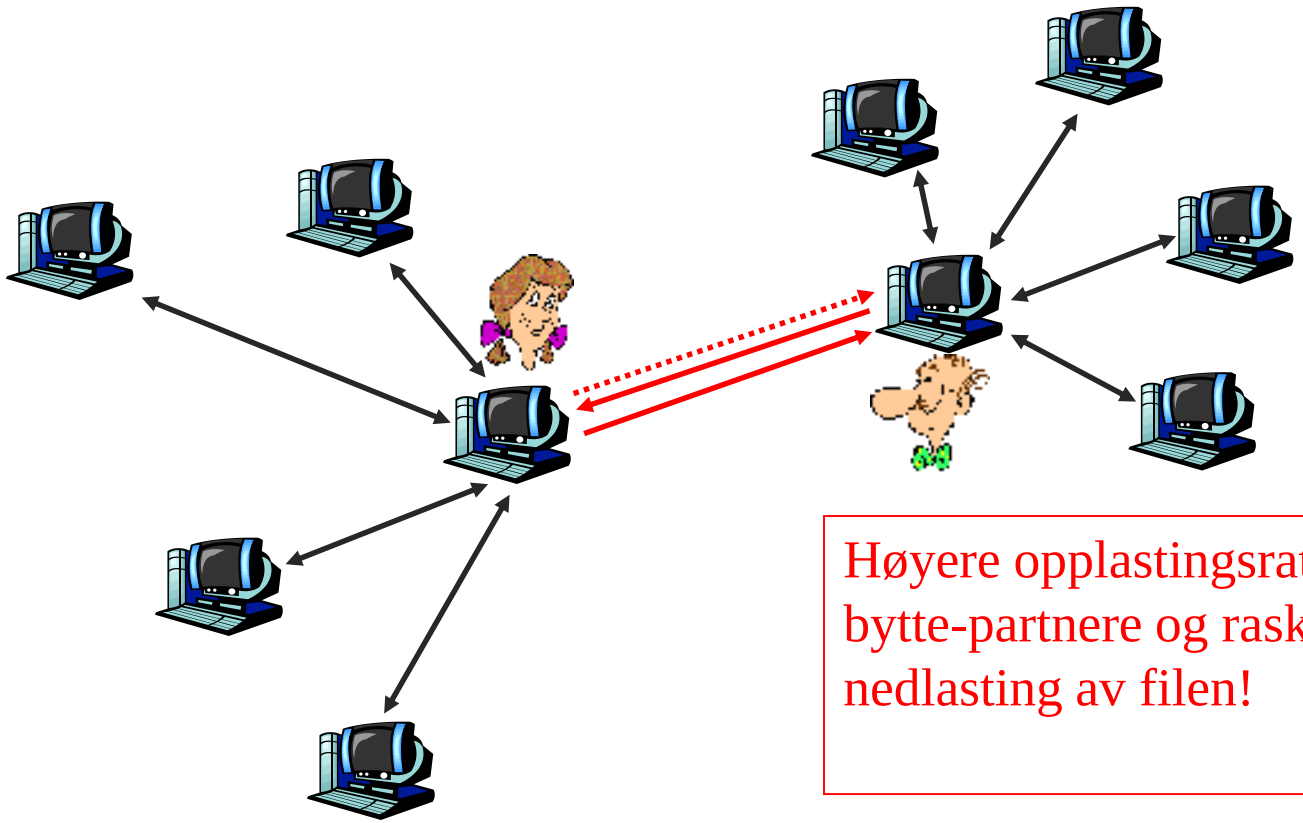
- Anne sender chunk'er til de fire naboene som for tiden sender henne chunks med høyest *bitrate*
  - re-vurderer topp 4 hvert 10. sekund
- hver 30. s: velger tilfeldig en annen peer, og begynner å sende "chunks"
  - nyvalgt peer kan bli en av de topp 4
  - "optimistisk nedstruping"

# Like-for-like (tit-for-tat)

(1) Anne "optimistisk nedstruper" Basse

(2) Anne blir en av Basses topp-fire leverandører; Basse gjør som Anne

(3) Basse blir en av Anne sine topp-fire leverandører



Høyere opplastingsrate gir bedre bytte-partnere og raskere nedlasting av filen!

# DHT (Distribuert Hash Tabell)

- DHT er en teknikk for å opprettholde, og videreformidle peer-listene der Peers selv fungerer som Tracker-servere
- System for innmelding, utmelding og oppslag i «Tracker-databasen» basert på «nabosladder» og IP-adresser.
  - Hash-funksjonen tilordner ett tall til hvert nøkkel-verdi-parr
  - Lagres hos den Peer hvis adresse er nærmest hash-verdien.

# HVA SKAL VI KUNNE NÅ?

# Skal kunne (1)

- TCP/IP-modellen
  - Funksjonelle nivåer og hvilke oppgaver som løses på dem
- Klient/Tjener vs P2P
  - Viktigste forskjeller
  - Fordeler og ulemper…
- HTTP
  - Meldingsutveksling
  - Typer spørringer og svar
  - Tilstandsløshet og konsekvenser av det
- SMTP
  - Meldingsutveksling og syntax
  - Forskjell på SMTP-kommandoer og Epost-header
- MIME

# Skal kunne (2)

- DNS
  - Oppbygging av systemet
    - Rotservere
    - TLD
    - Sonefiler
  - Typer Resource Records
    - A, AAAA, MX, NS, PTR, CNAME, ..
    - Bruk av `nslookup`
- `Portnummer:`
  - `SMTP:  25  (TCP)`
  - `HTTP:  80  (TCP)`
  - `HTTPS: 443 (TCP)`
  - `DNS:  53  (UDP)`