

Denne forelesningsøkten vil bli tatt opp og lagt ut i emnet i etterkant.

Hvis du ikke vil være med på opptaket:

 ^ Start Video	La være å delta med webkameraet ditt.
 ^ Unmute	La være å delta med mikrofonen din.
To: Marianne Sundby (Privately) Type message here...	Still spørsmål i Chat i stedet for som lyd. Hvis du ønsker kan spørsmålet også sendes privat til foreleser.



Høyskolen
Kristiania

TK1100

Digital teknologi

5. FORELESNING

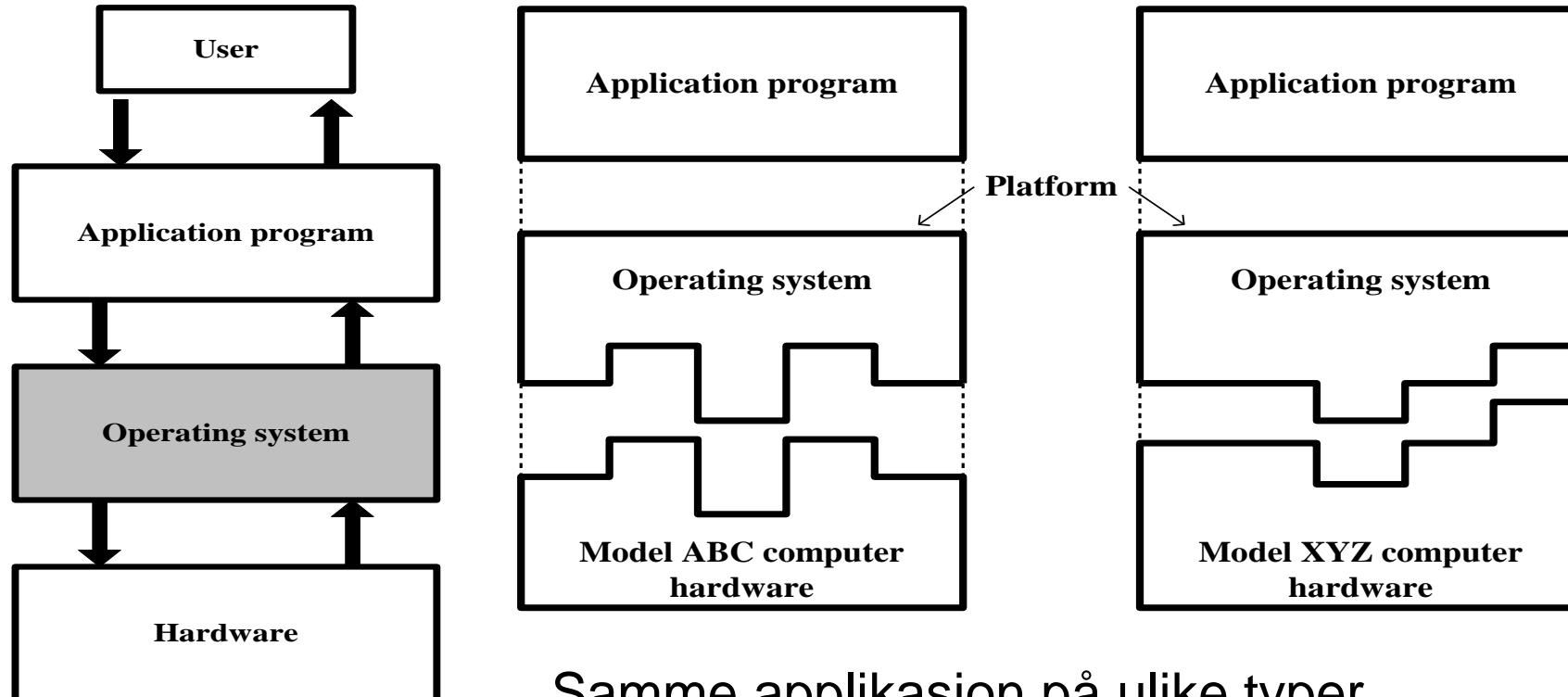


I dag

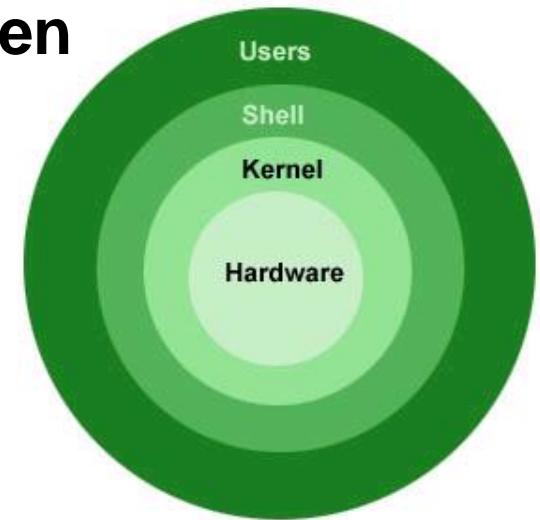
- Fokusere på å få en *oversikt*
- Lære **begrepene** som benyttes i litteratur og dokumentasjon
- Øving
 - Teori og begreper, samt litt historie
 - Lære seg kommando-linje interface på eget OS, hvordan manøvrere seg og hvordan gjøre enkle oppgaver i kommando-linje
 - Annet dere trenger; åpning av arkiv filer og installasjon av programvare

Hva er et operativsystem?

Operativsystemet er programvare som ligger mellom brukeren/programmereren og maskinvaren



Samme applikasjon på ulike typer maskinvare (OS = “plattform”)

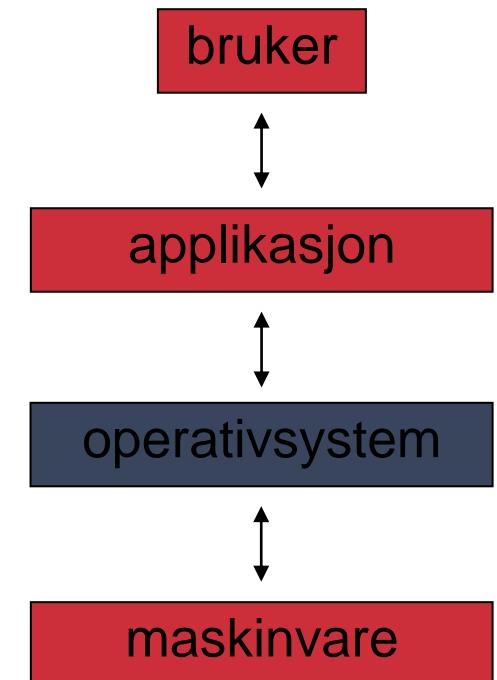


Hva er et operativsystem (OS)?

- “An operating system (OS) is a collection of programs that acts as an intermediary between the hardware and its user(s), providing a high-level interface to low level hardware resources, such as the CPU, memory, and I/O devices. The operating system provides various facilities and services that make the use of the hardware convenient, efficient, and safe”

Lazowska, E. D.: Contemporary Issues in Operating Systems , in: Encyclopedia of Computer Science, Ralston, A., Reilly, E. D. (Editors), IEEE Press, 1993, pp.980

- Det er en **utvidet/abstrakt maskin** (top-down view)
 - Skjuler de “grisete” detaljene i HW
 - Tilbyr brukeren og programmereren en **virtuell maskin** som det er enklere å programmere/bruke
- Det er en **ressursadministrator** (bottom-up view)
 - Hvert program får **tid** på ressursen
 - Hvert program får **plass** på ressursene (CPU, Minne,...).

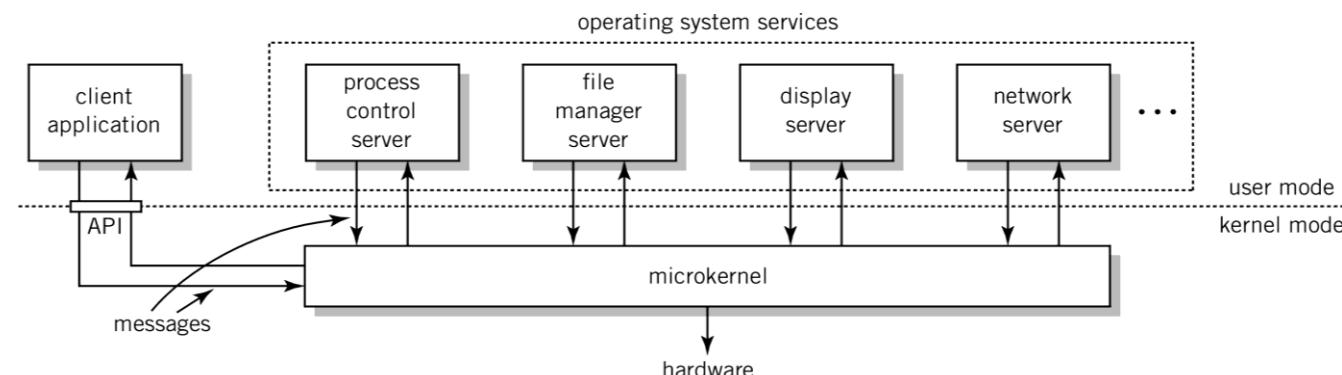
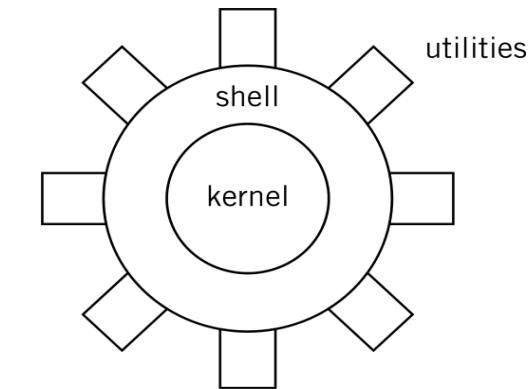


Abstraksjon

- Å abstrahere vil si å ta bort (uvesentlige) detaljer.
- For eksempel: En **fil** vil (oftest) bestå av metadata (data om data)
 - Et navn
 - når opprettet, sist endret, o.l.
 - hvor stor (KiB)
 - hvilke sektorer på disken data er fysisk plassert på
 - ... og selve innholdet (data/instruksjoner)
- Filene organiseres i **kataloger** (directories/mapper)
 - en annen abstraksjon, som også er **en type fil** som inneholder metadata og adresser til andre filer
- **Fysisk**/konkret er den bare magnetiserte områder i adresserte sektorer på en disk/CD/minnepinner
- For brukeren fremstår denne abstraksjonen som symboler h@n kan klikke på.
- For programmeren tilbyr OS **systemkall** som gjøre at h@n kan opprette, åpne, lukke, posisjonere seg innenfor, lese fra/skrive til

OS Organisering

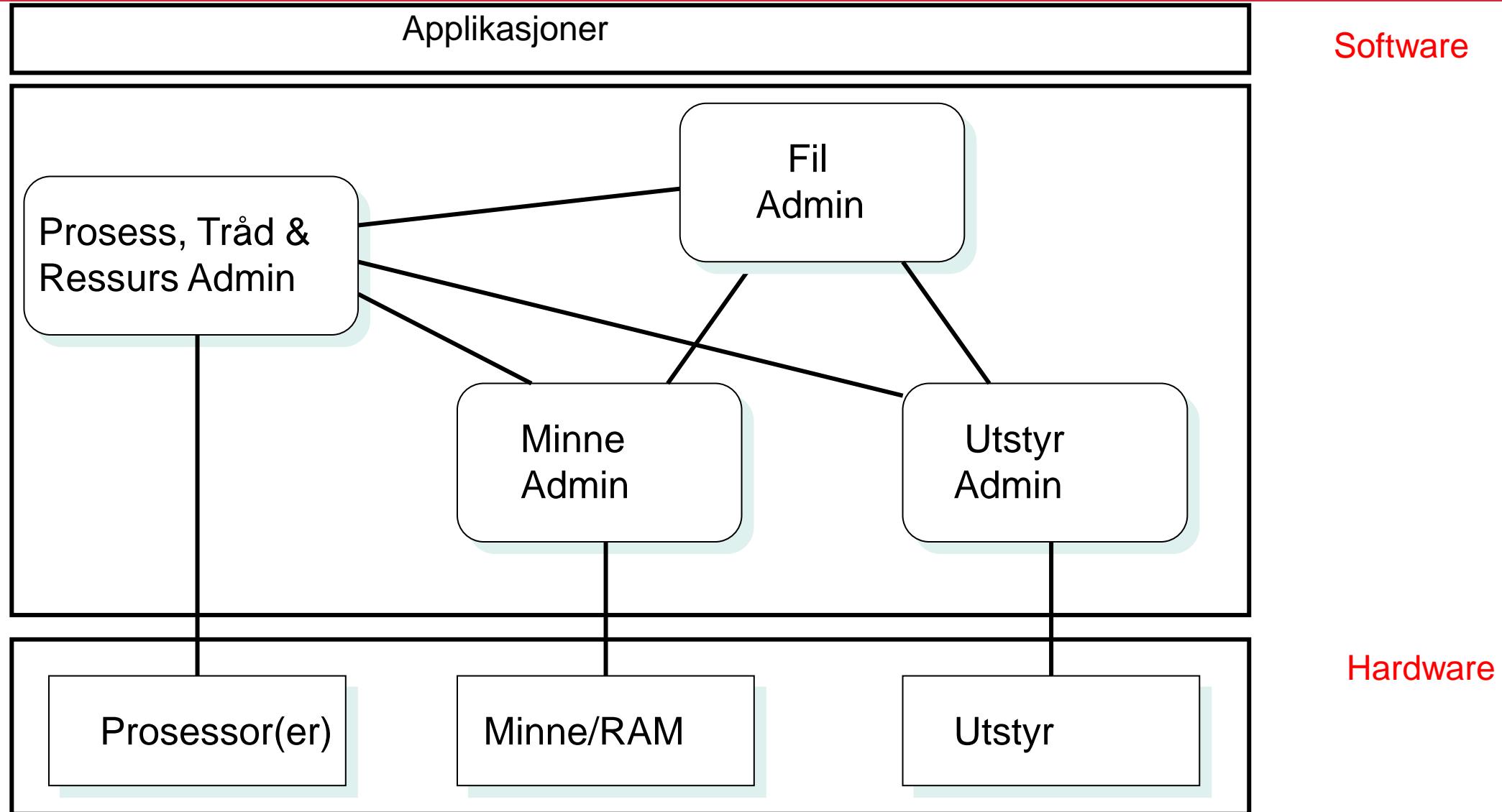
- Flere mulig tilnæringer, ingen standard.
- **Monolithic kernel** (“the big mess”):
 - Skrevet som en samling av funksjoner som er linket sammen til ett objekt.
 - Vanligvis effektivt (ingen grenser som krysses i kjernen)
 - Store, komplekse, kræsjer rimelig lett
 - UNIX, Linux, Windows NT, OSX (i praksis, men MACH i starten)
- **Micro kernel**
 - Kjerne med minimal funksjonalitet (administrere interrupt, minne, prosessor)
 - Andre tjenester implementeres som server prosesser i brukermodus i henhold til en klient-tjener-modell.
 - Mye meldingsutveksling (ineffektivt)
 - lite, modulært, utvidbart, portablet, ...
 - MACH, L4, Chorus, ...



Funksjoner i operativsystemer

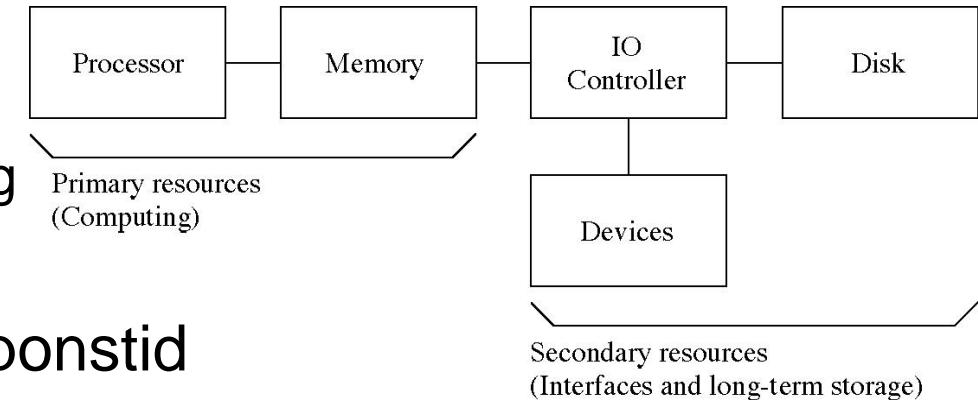
- Brukergrensesnitt (skall!)
- Applikasjons-kjøring
 - Tilbyr API (Application Programming Interface)
 - Tilbyr SPI (System Programming Interface)
- Håndtering av ressurser
 - Prosesser, hukommelse, eksterne lager, I/O-enheter
- Håndtering av maskinvare
 - Drivere!
- Håndtering av nettverk
- Sikkerhet
 - Oftest tett knyttet opp til **filsystemet**
- **Ikke alle** anvendelser av datamaskiner trenger et OS

Analyse av OS «kjerne»



Operativsystemet - hovedfunksjoner

- Oppstart av maskinen
- Intern behandling
 - GUI, «**auditing**», sikkerhet, feilbehandling
- Prosess behandling
 - Oppstart, fjerning, scheduling, responstid
- Minne behandling
 - Paging, segmentering, virtuelt minne, delt minne
- Disk behandling
 - Filsystem
- Device/Utstyr behandling
 - Drivere, interrupt behandling, spoolere
- Nettverk behandling
- Service-programmer (utilities)
 - Filbehandler, backup, defragmentering



Programmerer-perspektiv: Kjernen («KERNEL»)

- **Kjernen** starter opp og konfigurerer hardware
- **Kjernen** administrerer prosesser, tråder og ressurser

Begreper

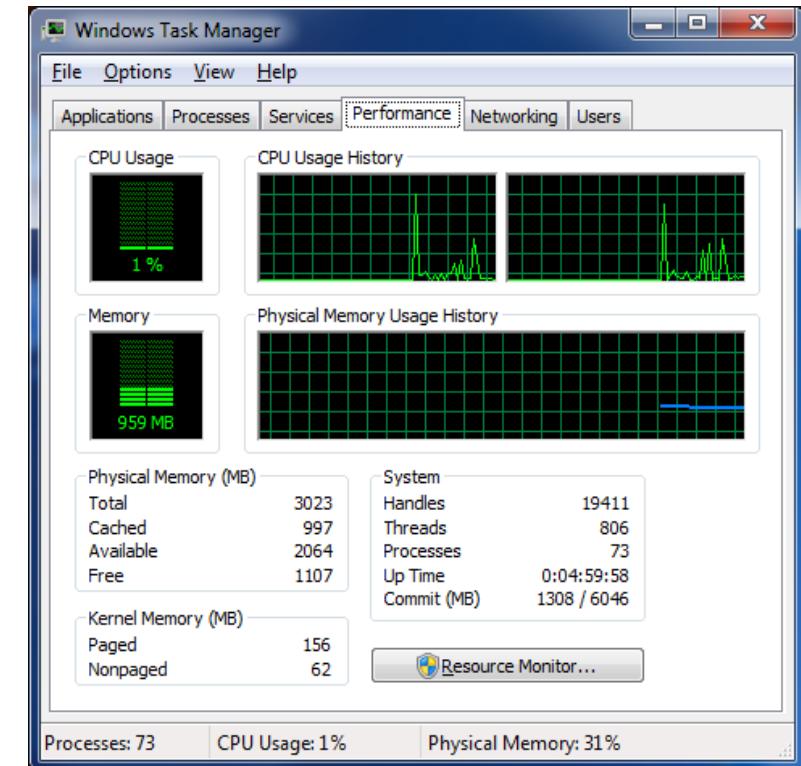
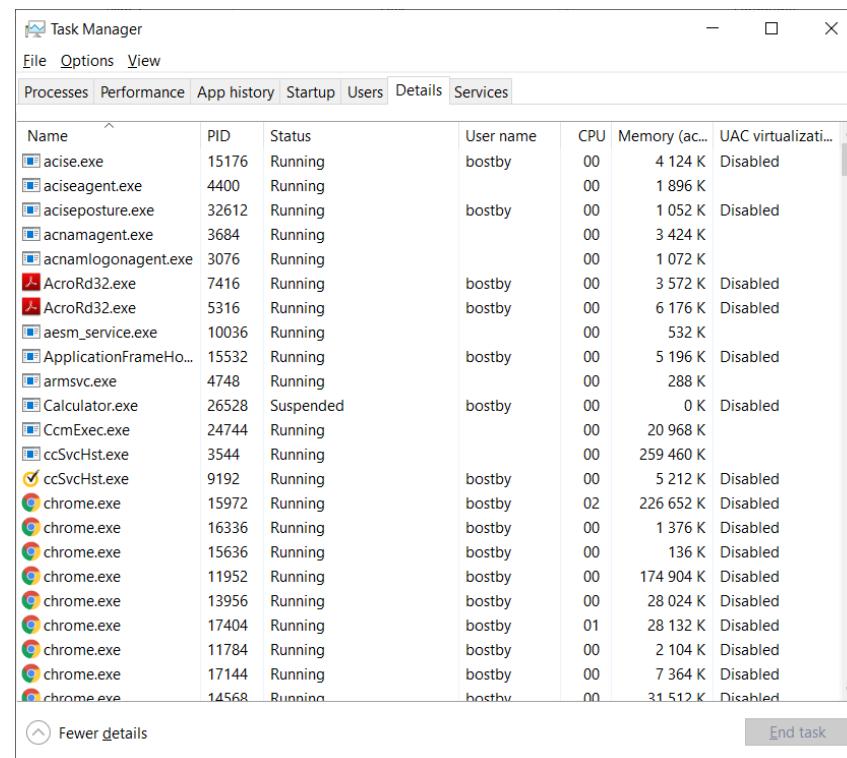
- **Prosess**
 - Et program under kjøring med tilhørende ressurser
- **Tråd**
 - "Thread of control" – selve kjøringen av instruksjoner (en og en...)
- **Ressurs**
 - Alt et program trenger for å kjøre ferdig.
 - CPU, RAM, I/O, ...

Bruker- vs kjernemodus

- For å oppnå sikkerhet og beskyttelse gir de fleste **CPUer** muligheten til å **kjøre instruksjoner** enten i **applikasjons**- eller **kjerne-modus**
- **Vanlige applikasjoner** og mange OS-tjenester og kjører i “**user mode**” (Intel/AMD “ring 3”)
 - Kan ikke aksessere HW, utstyrssdriver direkte , må bruke en API
 - Kun adgang til **minnet som OSet har tildelt**
 - Begrenset instruksjonssett
- **OS** kjører i **kjernemodus** (“ring 0”)
 - Tilgang til **hele minnet**
 - **Alle instruksjoner** kan kjøres
 - Ingen sikring fra HW

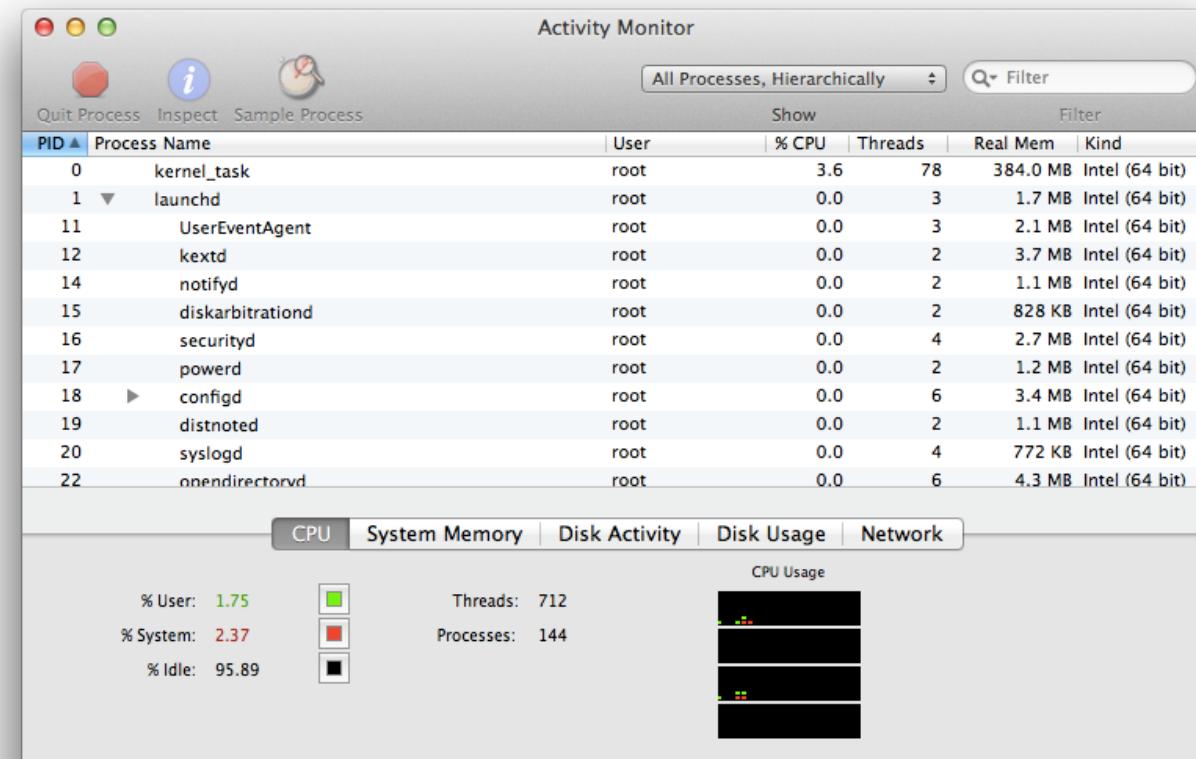
Prossesser og tråder: Taskmanager

- Windows – Taskmanger
 - Ctrl-Shft-Esc
 - Lar deg inspisere prosesser og deres bruk av ressurser



OSX – Activity Monitor

- Viser (sorterte) oversikter over prosesser, tråder, ressurser.
- Kan også bruke bash-verktøyene...



Linux: ps, top, mmfl

- Linux /OSX – ps, top og /proc

- ps – kommando

- top – applikasjon

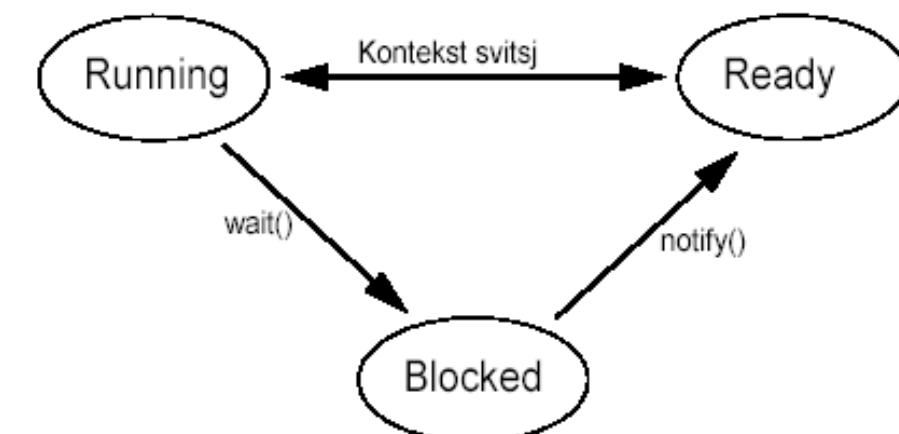
- /proc – del av ”filsystemet”

```
top - 17:14:56 up 138 days, 2:02, 3 users, load average: 0.00, 0.00, 0.00
Tasks: 72 total, 2 running, 70 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0%us, 0.0%sy, 0.0%ni, 96.7%id, 3.3%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2059632k total, 1587720k used, 471912k free, 342932k buffers
Swap: 2096472k total, 72k used, 2096400k free, 855740k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	15	0	10348	632	536	S	0.0	0.0	0:01.58	init
2	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.04	ksoftirqd/0
4	root	10	-5	0	0	0	S	0.0	0.0	3:27.07	events/0
5	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khelper
22	root	11	-5	0	0	0	S	0.0	0.0	0:00.00	kthread
26	root	10	-5	0	0	0	S	0.0	0.0	0:00.26	kblockd/0
27	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	kacpid
186	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	cqueue/0
189	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	khubd
191	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kseriod
259	root	10	-5	0	0	0	S	0.0	0.0	0:14.39	kswapd0
260	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	aio/0
466	root	11	-5	0	0	0	S	0.0	0.0	0:00.00	kpsmoused
496	root	10	-5	0	0	0	S	0.0	0.0	0:30.63	mpt_poll_0
497	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	scsi_eh_0

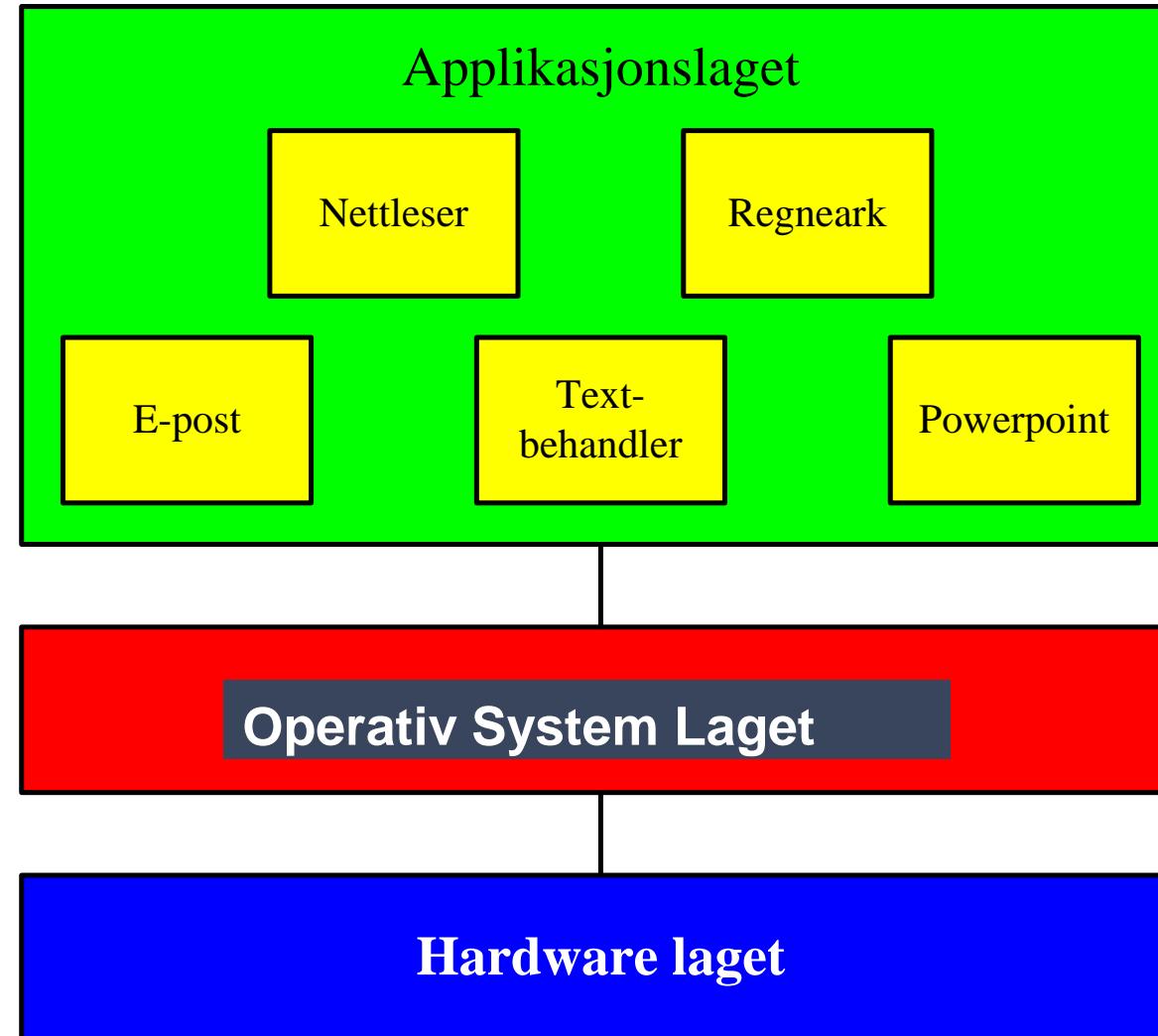
Operativsystemet - multitasking

- Datamaskinen kan kjøre flere prosesser «**samtidig**»
- CPUen kan bare behandle en instruksjon fra en prosess til enhver tid (på hver prosessorkjerne)
- Operativsystemet må derfor holde styr på hvilken prosess som skal få benytte CPUen
- CPU-bytte mellom prosesser heter **context switching**
 - Running: Kjører nå
 - Ready: Aktivisert og venter i kø
 - Blocked: Venter på å bli aktivisert



Mange Samtidige Oppgaver

- Bedre utnyttelse
 - mange samtidige prosesser
 - Utfører ulike oppgaver
 - Bruker ulike deler av maskinen
 - mange samtidige brukere
- Utfordringer
 - “samtidig” tilgang
 - beskyttelse/sikkerhet
 - rettferdighet
 - ...



Operativsystemet - minne

- Programmer som kjøres ligger i minnet
 - Operativsystemet er også et program!
- Vanlig RAM-minne er forbedret på flere måter
- Cache er en meget raskt minne (SRAM) som brukes som mellomlager
 - Brukes både mellom CPU og RAM og for enkelte devicer
- Hvis RAM-minnet er for liten, kan noe av innholdet midlertidig legges på disk («**swapping**»)

Kjøreplan (schedule)

- OS bruker en *kjøreplan* for å bestemme hvilken tråd som skal få lov å kjøre på prosessoren
- Kontekst-svitsjing tar tid
- Det finnes to overordnede kjøreplan-prinsipper
 - Ikke-preemptiv (frivillig)
 - Tråden som er *running* setter seg selv til *ready* (yield-instruksjon)
 - Høflighets-metoden
 - Preemptiv
 - Et klokkestyrt avbrudd sørger med jevne mellomrom for at den tråden som er *running* settes til *ready*
 - Power-metoden
- NT, OSX og Linux har **preemptive kjøreplaner** basert på prioritets-køer og Round Robin ("Ta den ring") algoritmen

Ulike typer eksekverbare filer

- Ulike OS benytter ulike formater for kjørbare programfiler
 - Windows: Portable Executable (PE)
 - Linux: ELF
 - OSX: Mach-O
- De benytter også forskjellige systemkall og standard biblioteker

Prosesssen (tråden) sitt Minne

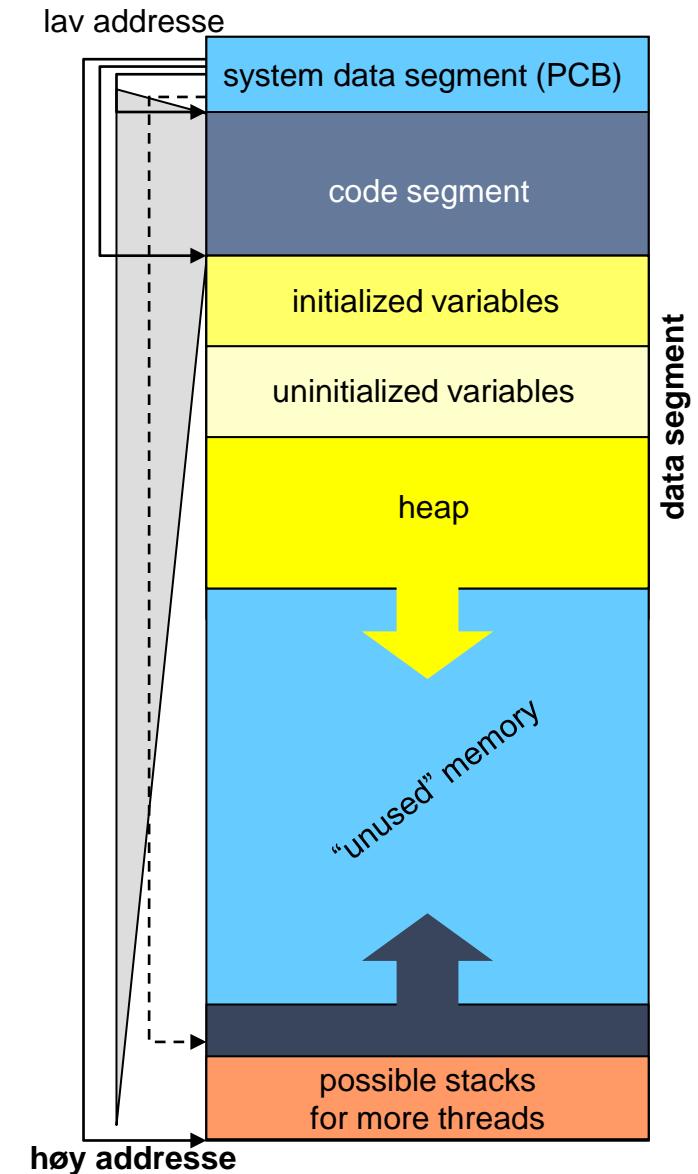
På Intel arkitekturen partisjonerer en oppgave (task) sitt tildelte minne

- et **TEXT (code)** segment
 - Lest fra program fil for eksempel av `exec`
 - vanligvis read-only
 - Kan deles av flere tråder
- et **DATA** segment
 - initialiserte globale variabler (0 / NULL)
 - uinitialiserte globale variabler
 - heap
 - dynamisk minne f.eks., alokert med `malloc`
 - vokser oppover
- et **STACK** segment
 - Variabler i en funksjon
 - Lagrede registertilstander (kallende funksjons EIP)
 - Vokser nedover
- **system data segment(PCB)**
 - segment pekere
 - pid
 - program and stack pekere
 - ...
- Flere stacker for trådene

```

8048314 <add>:
8048314: push %ebp
8048315: mov %esp,%ebp
8048317: mov 0xc(%ebp),%eax
804831a: add 0x8(%ebp),%eax
804831d: pop %ebp
804831e: ret
804831f <main>:
804831f: push %ebp
8048320: mov %esp,%ebp
8048322: sub $0x18,%esp
8048325: and $0xfffffffff0,%esp
8048328: mov $0x0,%eax
804832d: sub %eax,%esp
804832f: movl $0x0,0xfffffffffc(%ebp)
8048336: movl $0x2,0x4(%esp,1)
804833e: movl $0x4,(%esp,1)
8048345: call 8048314 <add>
804834a: mov %eax,0xfffffffffc(%ebp)
804834d: leave
804834e: ret
804834f: nop
...
...
...

```

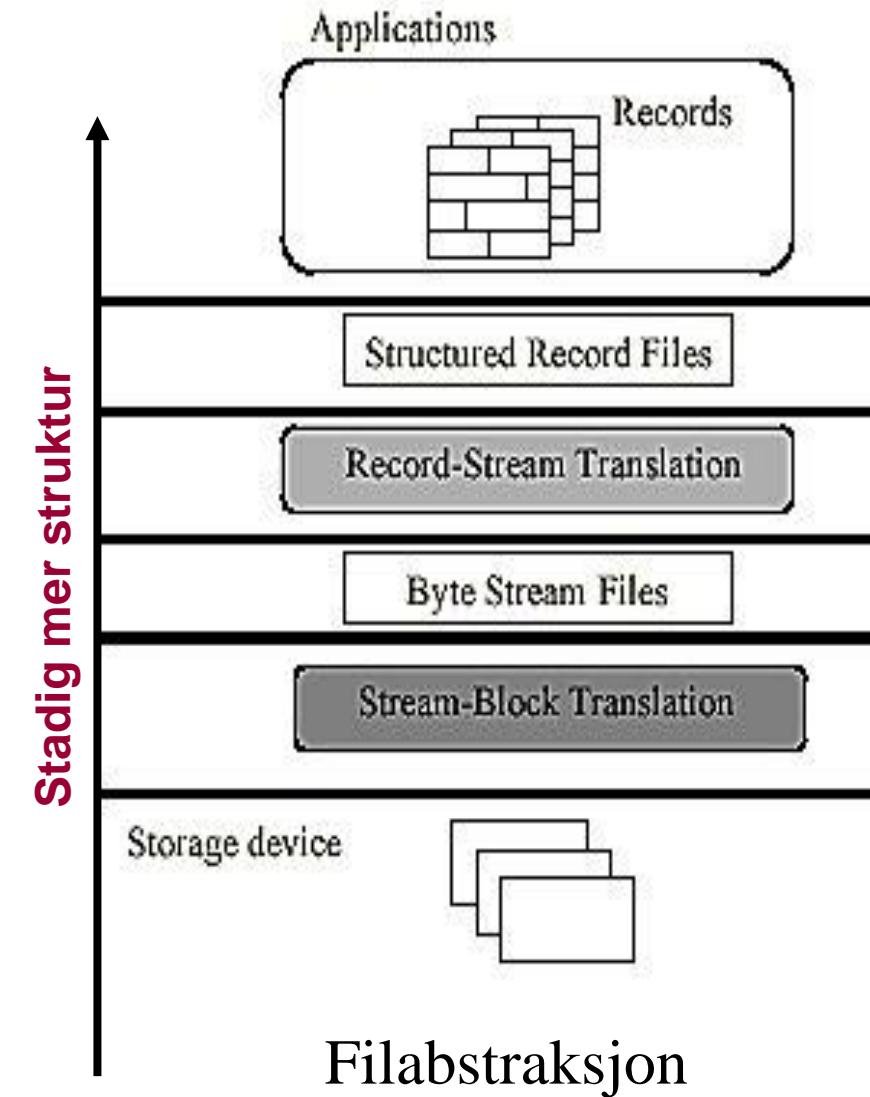


Operativsystemet - filer

- Data som ligger i en hukommelse forsvinner når strømmen slås av
- Data kan også lagres permanent
 - Disk, diskett, CD, tape,
- Et slikt lager består av mapper og filer
 - Hierarkisk struktur
 - Navn, tilgangs-rettigheter, skapelses-dato,

Filadministrasjon

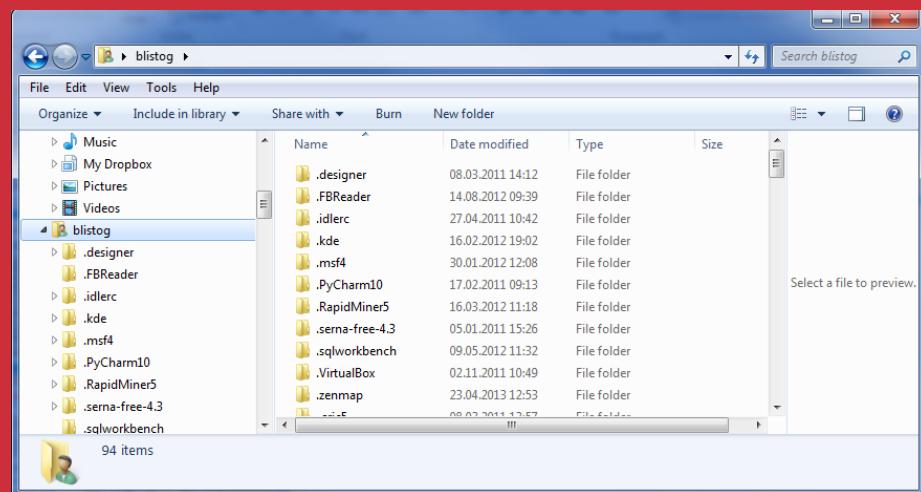
- Filadministratoren i OS'et må:
 - **Lagre** informasjon **sikkert** på utstyr (lagringsenhet: harddisk, ...)
 - Kartlegge **sammenhengen** mellom **blokklagring** på utstyret og **filabstraksjonen** (logisk "bilde")
 - Allokere/deallokere lagringsenheter
 - Administrere deling av data
 - Beskytte mot feil, crash og ondsinnede brukere
 - Samtidskontroll (lock)



Drive

- Drive er programvare som kan oversette frem og tilbake mellom OS sine krav og instruksjonsettet til **kontrolleren** på utstyret.
- Hovedårsaken til krasj er dårlig skrevne drive
 - Microsoft påstår "minst" 70%

- Viktigst i vanlig bruk:
 - 1) Starte opp programmer («prosesser»)
 - 2) Manøvrere seg rundt i filsystemet



Brukerperspektiv:

SKALLETT

C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\blistog>dir
 Volume in drive C is OS
 Volume Serial Number is 0A3E-CF8C

 Directory of C:\Users\blistog

03.10.2013 17:09    <DIR>      .
03.10.2013 17:09    <DIR>      ..
30.01.2012 13:06            1 934 .armitage.prop
08.03.2011 15:12    <DIR>      .designer
14.09.2011 20:31            389 .drjava
14.08.2012 09:39    <DIR>      .FBReader
12.02.2012 20:19            35 .gtk-bookmarks
```

home.nith.no

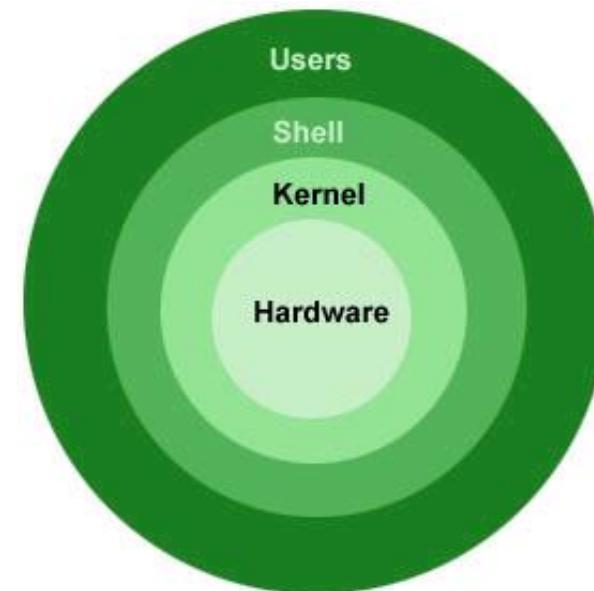
```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Hei! Jøss, er det ikke blistog, som er her for å lage krøll igjen?!
Test

-->ps ef
  PID TTY      STAT   TIME  COMMAND
 6456 pts/0    Ss    0:00  -bash USER=blistog LOGNAME=blistog HOME=/home/b
 6462 pts/0    R+    0:00  \_ ps ef TERM=xterm SHELL=/bin/bash SSH_CLIENT
 6274 pts/8    Ss    0:00  -bash USER=blistog LOGNAME=blistog HOME=/home/b
 6435 pts/8    T     0:00  \_ /usr/lib/games/nethack/nethack-console
 6440 pts/8    S+    0:00  \_ telnet towel.blinkenlights.nl TERM=xterm SH
-->whoami
blistog
-->ps
  PID TTY      TIME CMD
 6456 pts/0    00:00:00 bash
 6464 pts/0    00:00:00 ps
-->kill -9 6440
-->
-->[green square]
```

Brukerperspektiv: Skallet

- Brukeren kan gi ordre til OS via skallet (shell)
 - Starte opp programmer
 - Flytte filer
 - ...
- Windows
 - Explorer (GUI)
 - Cmd.exe (NT)
 - Powershell (neste..)
- Linux
 - F.eks. Nautilus (Gnome)
 - bash (m.fl.)
- OS X
 - Finder (GUI)
 - bash (m.fl.)



- Likheter
 - Tekst- og kommando-basert
 - Hierarkisk filstruktur
 - Tilsvarende fil-adgang (les, skriv, ...)
 - Standard I/O med piping (>, >>, <, |)

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\blistog>dir
Volume in drive C is OS
Volume Serial Number is 0A3E-CF8C

Directory of C:\Users\blistog

14.09.2011  20:31    <DIR>          .
14.09.2011  20:31    <DIR>          ..
08.04.2011  11:28    <DIR>          .android
08.03.2011  15:12    <DIR>          .designer
14.09.2011  20:31          389  .drjava
27.04.2011  10:42    <DIR>          .idlerc
21.03.2011  16:29    <DIR>          .IntelliJIdea10
15.03.2011  13:40          192  .jline-jython.history
17.02.2011  10:13    <DIR>          .PyCharm10
```

```
blistog@home.nith.no's password:  
Last login: Mon Sep 20 22:53:28 2010 from nith-vpn-nat02.osl.basefarm.n  
Hei! Jøss, er det ikke blistog, som er her for å lage krøll igjen?!  
  
~>ls -la  
total 29480  
drwxr-xr-x  28 blistog  users        4096 Jul  30  01:30 .  
drwxr-xr-x 1298 root    root        36864 Sep 23 12:01 ..  
-rwxr-xr-x  1 blistog  users         0 Sep 26  2009 a.out  
-rw-r--r--  1 blistog  users        4 Oct  6  2008 arg  
-rwx-----  1 blistog  users      10412 Sep 20 23:20 .bash_history  
-rw-r--r--  1 blistog  users      6126 Oct  9  2008 bash_igjen  
-rw-----  1 blistog  users      4934 Oct  9  2008 bash_igjen.save  
-rw-r--r--  1 blistog  users       17 Feb  6  2008 .bash_login  
-rw-r--r--  1 blistog  users       24 Sep  9  2003 .bash_logout  
-rw-r--r--  1 blistog  users       334 Feb  5  2008 .bash_profile  
-rw-r--r--  1 blistog  users       141 Feb  6  2008 .bashrc
```

Kommandoer CMD vs BASH

- display list of files
- display contents of file
- copy file
- rename file
- delete file
- delete directory
- find string
- create directory
- change working directory
- get help
- print file
- Endre rettigheter

<u>DOS</u>	<u>UNIX</u>
dir	ls
type	cat
copy	cp
ren	mv
del	rm
rd	rmdir
find	grep
md	mkdir
cd	cd
help	man
print	lpr
attrib	chmod

Kommandoer og **flagg**

- Hvordan en kommando oppfører seg kan modifiseres ved å sette ulike **flagg**
 - plasseres direkte eller etter en flaggmarkør (*/*, *-*, *--*)

```
dir /?  
ls -la  
ps aux
```

-

Kommandoer og omdirigering

- Vanligvis er output til skjerm (konsoll), men i tillegg har du noen standard omdirigeringsoperatorer

>

- Skriver output inn i en fil
- Ex: `dir > katalog.txt`, eller
`ls -la > directorylist`

>>

- Skriver output inn på **slutten** av en fil

<

- Tar **innholdet i en fil** og bruker det som argumenter til kommandoen

|

- Tar output fra en kommando og leverer den som input til neste kommando
- Ex: `dir | sort /R | more`
`ps aux | grep blistog | wc`
- «**pipe**»

Signaler

- Under både Windows og Linux/OSX kan du i CLI-skall («command line interface» = kommandolinje-grensesnitt) gi signaler til OS med noen hurtigtaster
 - Ctrl-C Aborerer kjørende prosess
 - Ctrl-D Slutt på filen (EOF)
 - Ctrl-Z Sett kjørende prosess til å sove
 - m.fl.

Hvorfor bruke kommando-skall?

- GUI er utmerket til «peke, dra, klikke» oppgaver
 - Dårlig til f.eks. automatisering pga **fattig semantikk**
 - Kommando-skall tilbyr oftest et fullstendig programmeringspråk (kalt batch/bash)
- Servere
 - De fleste web-servere kjører Apache på Linux, og settes ikke opp med GUI da det ville være dårlig ressursbruk.
 - Microsoft har også begynt å satse mer på skall enn GUI for servere.. (**Powershell**)

- Hardware har styrt hva OS tilbyr/gjør;
- Ny hardware har ofte kommet til ut fra OS behov.

HISTORIKK

(Trenger ikke pugge dette, men spennende og nyttig å ha vært gjennom)

Operativsystemet - tidlig utvikling

- Frem mot 1960-tallet var det ikke bruk for operativsystem. Datamaskinen ble brukt av en person eller en jobb om gangen
- Ved **batch-kjøring** måtte maskinen ha en kø-ordner
- Ved **time-sharing** (multitasking) måtte maskinen holde rede på flere prosesser på en gang
- Bruk av masselager krevde et organisert filsystem
- Virtuelt minne krevde bedre styring mot filsystemet
- Stadig nytt eksternt utstyr krevde forenklet I/O-kontroll
- Brukere var datafolk

Operativsystemet - videre utvikling

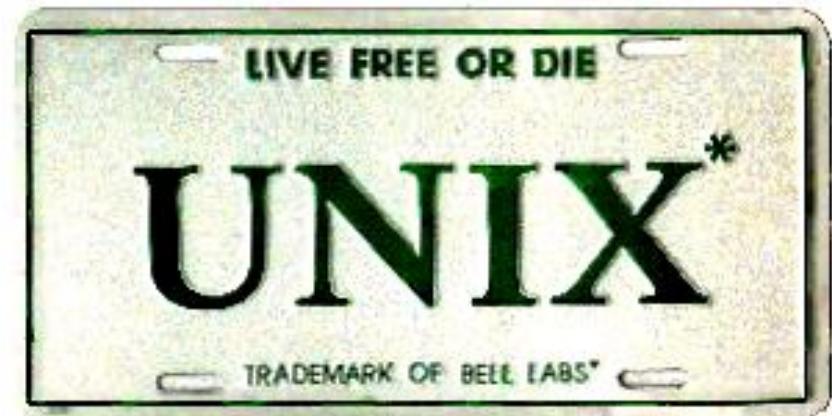
- Komplisert oppstart-prosedyre
- Flere interaktive brukere på samme maskin
- Behov for generelle service-programmer for brukerne
- Maskiner i nettverk
- Beskytte maskiner og brukere mot maskiner og brukere
- PCer gjennomgikk samme utvikling som større maskiner allerede hadde
- Økende tilbud av applikasjoner på maskin og nett
- Tyngden av brukerne uten databakgrunn

- Parallelle systemer
 - Mer enn en CPU i samme maskinen
 - Kan kjøre flere jobber samtidig
- Distribuerte systemer
 - Flere maskiner jobber sammen
 - Deling av jobber og ressurser
- Sanntid systemer
 - Tidskritisk responstid, styrt av interrupt
- Nettverk
- Mobiltelefoner o.l.

- **UNIX**
 - Skrevet i C (1971), tekstbasert, portabelt, flerbruker
 - Linux som moderne (PC) utgave
 - Stor frihetsgrad både fordel og ulempe
- **MS-DOS**
 - Skrevet i maskinkode (1980) av Microsoft for IBM
 - Tekstbasert, enbruker
 - Senere bygget Windows 3.0-11 GUI oppnå DOS
 - Liten frihetsgrad

- DOS hentet mye fra UNIX og kan derfor synes å være relativt likt
 - CP/M: Gary Kildall, 1973 for 8080 og 8" diskett (DR-DOS)
 - 86-DOS: Tim Paterson, 1980 for 8086
 - **FAT** filsystem
 - MS-DOS: Microsoft for IBM, 1981, basert på 86-DOS som de kjøpte for 75.000 dollar
- Hoved-forskjellene ligger ikke så mye i kommandoene men i frihet, struktur og portabilitet

- Forskjeller
 - DOS var en-bruker, UNIX var fler-bruker
 - DOS støttet bare enkel-prosessering, UNIX støtter multi-prosessering
 - DOS hadde kommando-tolker (command.com -> cmd.exe), UNIX bruker skall (shell, bash, csh, tcsh,...)
 - DOS/NT kjøres på Intel-kompatible prosessorer, UNIX er mye mer portabel
 - DOS/NT brukte bat og cmd-filer, UNIX bruker skall-skript



- Startet som Multics på MIT 1965
 - Første time-sharing OS
- Videreutviklet hos Bell 1965-1969
 - Ken Thompson, Dennis Ritchie,.....
- Overført fra PDP-7 til PDP-11 1970
 - Unics → Unix
 - Skrevet i C
 - Dannede basis for portabilitet
- Linux - Linus Torvalds 1991



Forenklet
tidskart!

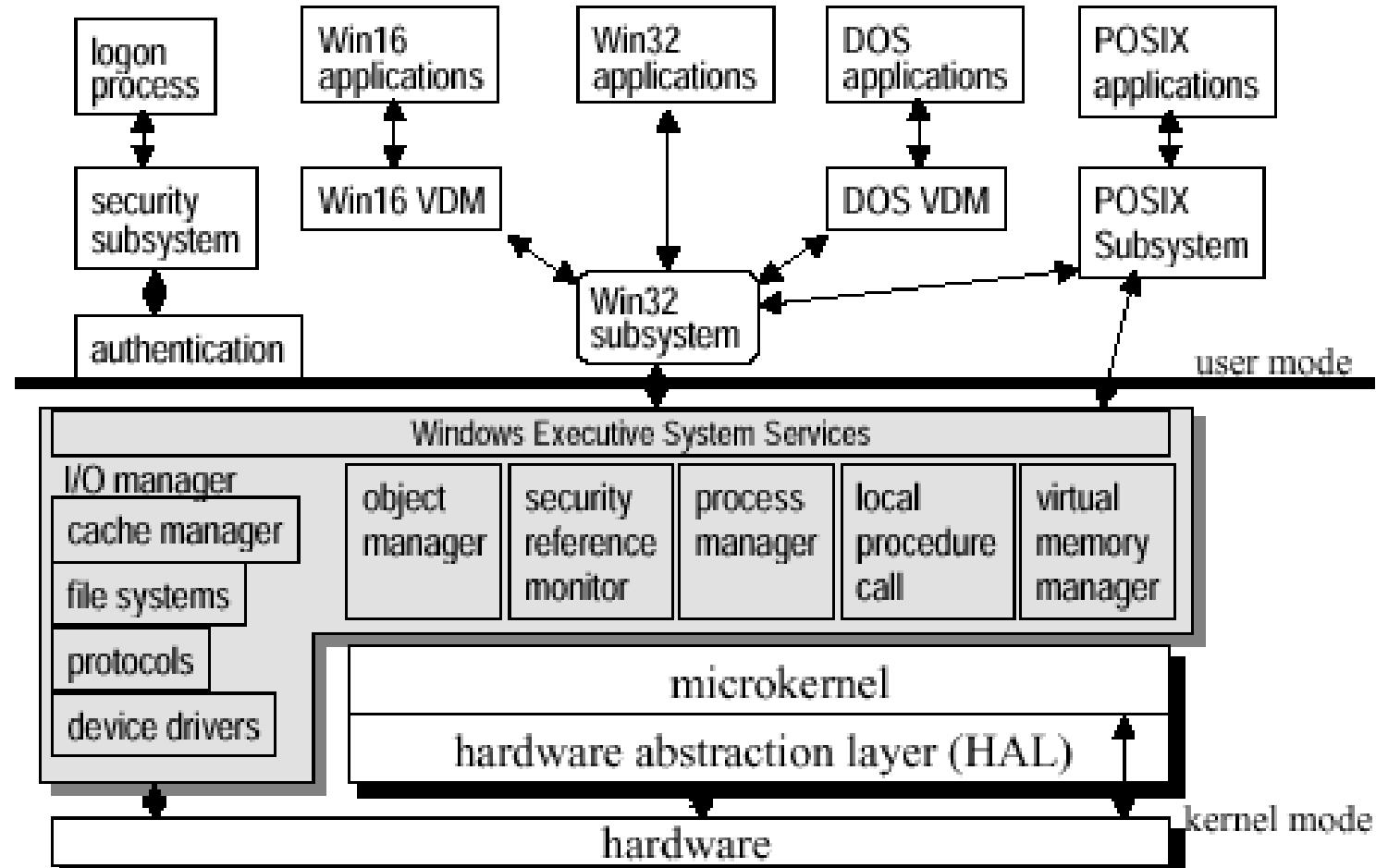
MULTI-TASKING

- Alle moderne OS tilbyr multitasking
- Flere prosesser kjører «samtidig»,
 - «samtidig» betyr at de får tildelt tidsrom på CPU for å kjøre sin(e) tråd(er)
 - eller kjøres på hver sin prosessorkjerne
 - Synchroniserings-problemer
 - OS besørger kjøreplan («scheduling»)
 - OS sørger for minne-isolasjon og –samarbeid mellom prosesser/tråder



- Windows 1.0, 2.0, 3.0 og 3.1 er grafiske grensesnitt for DOS
- Windows 95/98 inneholder DOS som en service
- Windows NT var en mer portabel utgave av Windows
 - Nytt kjerne-design fra bunnen av. Ikke basert på DOS
 - Modulært, 32 bit system for mange samtidige omgivelser
 - Robuste mekanismer for kritiske prosesser
 - Kjører alle Windows-applikasjoner med samme interface
- I Windows XP møttes Windows 98 og Windows 2000
- Vista tilfører ny sikkerhets-modell og grensesnitt
- Windows 7 og Server 2008 er fortsatt samme kjerne (NT)
- Windows 8 og 10 er neste trinn, men involverer ingen/små endringer i kjernen (Windows Vista er internt «Windows 6.0», Windows 10 er internt «Windows 6.3»...)

- System struktur



- Registry
 - Sentralisert database for system konfigurering informasjon
 - Videreføring av DOS og Windows INI-filer
 - Brukes av forskjellige Win enheter
 - Recognizer: Gjenkjenner hardware ved oppstart
 - Setup: Verktøy for system konfigurering
 - Kernel: CPU konfigurering og balansering av system belastning
 - Drivers: Informasjon om initialisering av hardware

- Registry

- Hierarkisk oppbygning
 - HKEY_LOCAL_MACHINE
 - HKEY_CLASSES_ROOT
 - HKEY_CURRENT_USER
 - HKEY_USERS
 - HKEY_CURRENT_CONFIG
 - HKEY_DYN_DATA
- Oppdateringer valideres ikke mot andre deler av registry
- Backup og restore er følsomt

REGEDIT4

```
[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\explorer\Advanced\Folder\StartMenuScrollPrograms]
"Type"="checkbox"
"Text"="Multi-Column Start Menu"
"HasKeyRoot"="dword:80000001"
"RegPath"="Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced"
"ValueName"="StartMenuScrollPrograms"
"CheckedValue"="dword:00000000"
"UncheckedValue"="dword:00000001"
"DefaultValue"="dword:00000000"
"HelpID"="update.hlp"
```

- NT File System (**NTFS**)
 - Stor forbedring i forhold til DOS's FAT
 - Bedre egenskaper og ytelse
 - Data gjenvinning, sikkerhet, store filer, komprimering
 - Metadata lagret i Master File Table (MFT)
 - Data om data: Hvordan, når, av hvem og formatering
 - NTFS er ikke bare byte-strømmer
 - MFT inneholder opplysninger om dataene
 - Filreferanser er mer enn bare adresser
 - 48 bit filnummer (peker), 16 bit sekvensnummer (multidisk)
 - Logging av transaksjoner
 - NTFS videreutvikles ikke og venter på en arvtager

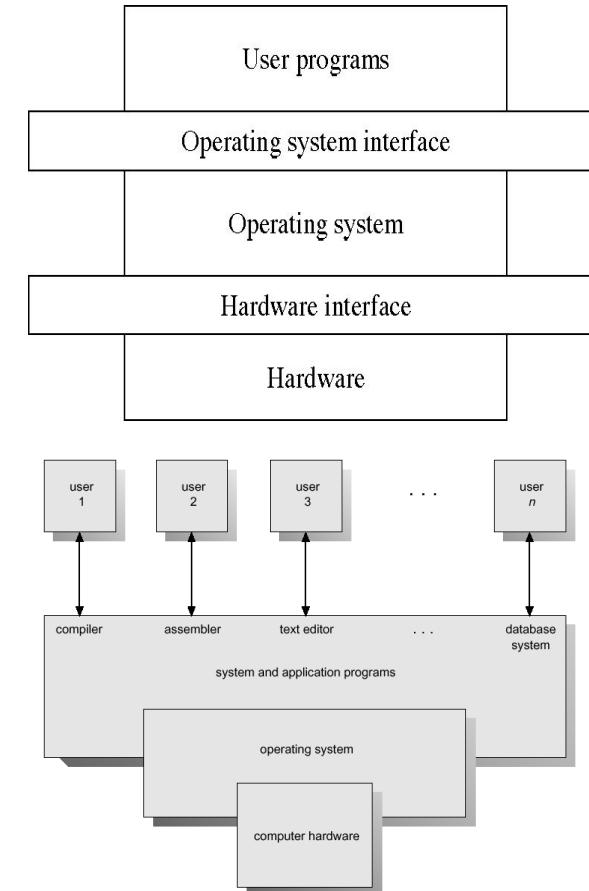
- Apple startet også med et «rent» DOS (Disk Operating System) 1976-1984
 - Ikke multitask, 1-nivå filsystem («uten kataloger»)
- MacIntosh introdusert i 1984
 - **Finder** som GUI-«skall»
 - Nytt filsystem etter hvert : **Hierarchical File System**
 - Toolbox for GUI-app utvikling

- Videreutvikling av NeXTSTEP
 - Basert på **Mach mikrokjerne**, FreeBSD (Unix) og I/O Kit driver-pakken
 - **Kjernen** kalles Darwin/**XNU** (X is Not Unix)
 - Mye Open Source takket være FreeBSD bakgrunnen

Oppsummering

Operativsystemet – generelle krav

- Operativsystemet skal altså organisere de tilgjengelige ressursene og administrere dem på best mulig måte både for maskiner og brukere
- Brukerne vil konsentrere seg om applikasjoner uten å tenke på maskinenes software- og hardware- finurligheter
- Brukerne vil helst bli presentert for et miljø som er uavhengig av foreliggende hardware
- Opplærings-terskelen skal helst være så lav som mulig



Hva skal vi kunne?

- Definere prosess, tråd og ressurs
- Forklare hvordan OS abstraherer Hardware og dermed tilbyr et enklere programmerings- og bruker-grensesnitt
- Kunne forklare OS rolle som ressurs-administrator: Prosess/tråd-adm, Minne-adm, Fil-adm, Utstyr-adm.
- Vite hvilke problemer som må løses for å få til multitasking
- Kjenne minne-layouten og kjøremønsteret til et vanlig program på en Intel-type prosessor
- Forklare forskjellen på User- og Kernel-modus på CPU

ØVINGSOPPGAVE

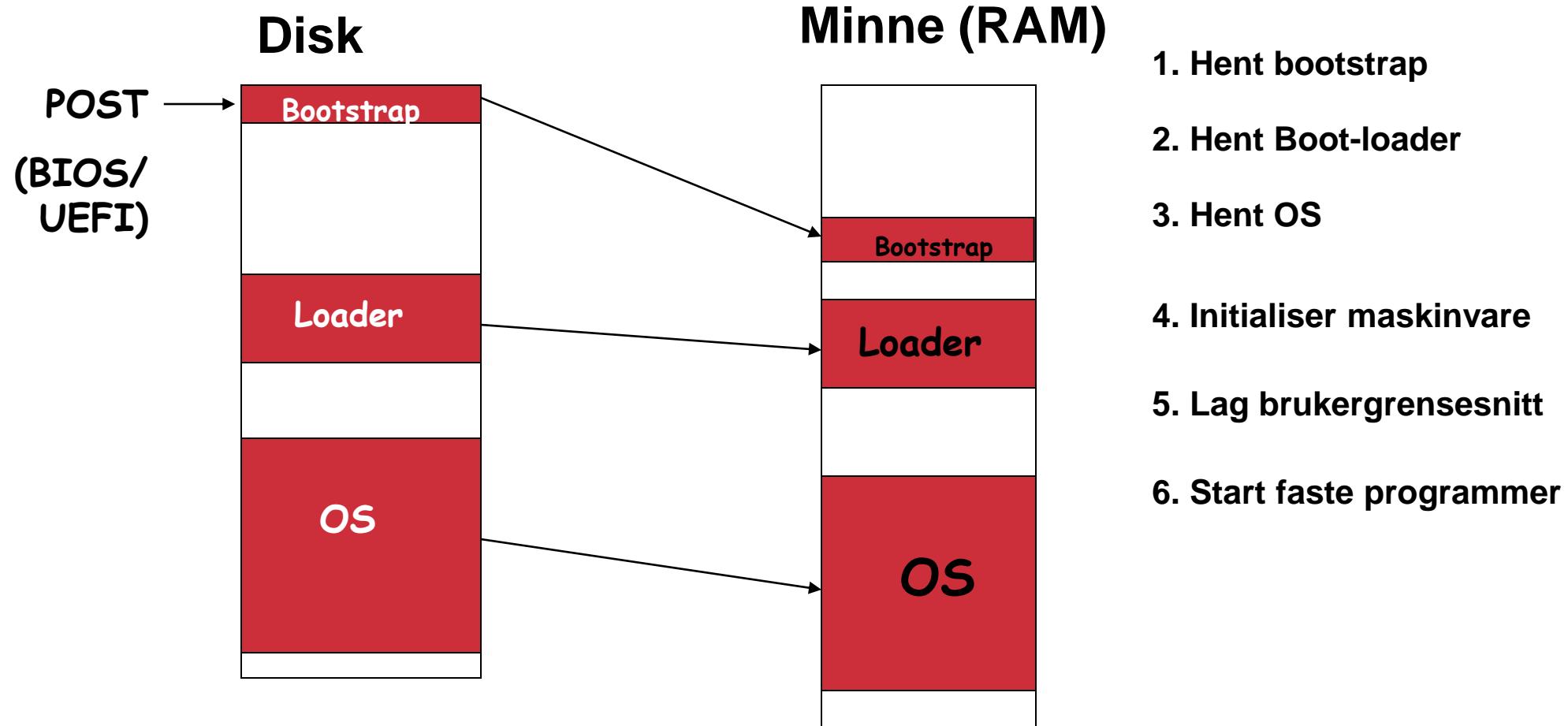
- Avanserte operativsystem operasjoner
 - TK1100_F05_Ø05_OS_Advanced.pdf
 - Arkiv filer; ZIP, TAR.GZ, krypterte arkiver
 - Installere programvare (.MSI/.EXE på Windows, .DMG på Mac, TAR.GZ på Linux)
 - Bruke CMD (shell) i operativsystem
- **Sett av flere timer til disse øvingsoppgavene, og BRUK veilederne, for de fleste er dette mer avansert PC bruk enn dere er vant til – og praktiske oppgaver i resten av semesteret baserer seg på dette!**
- **Praktiske oppgaver kommer på eksamen**
- TK1102_F05_OS_Øvingsoppgaver.pdf

For valgfritt egenstudie

For de som ønsker å lære noen emner mer i dybden for å forstå det bedre er det her samlet noen ekstra temaer relatert til dagens undervisning, det må forventes en del egenarbeid for å forstå disse emnene.

Det vil ikke komme spørsmål på eksamen fra disse, og dette er altså ikke ansett for å være en del av pensum.

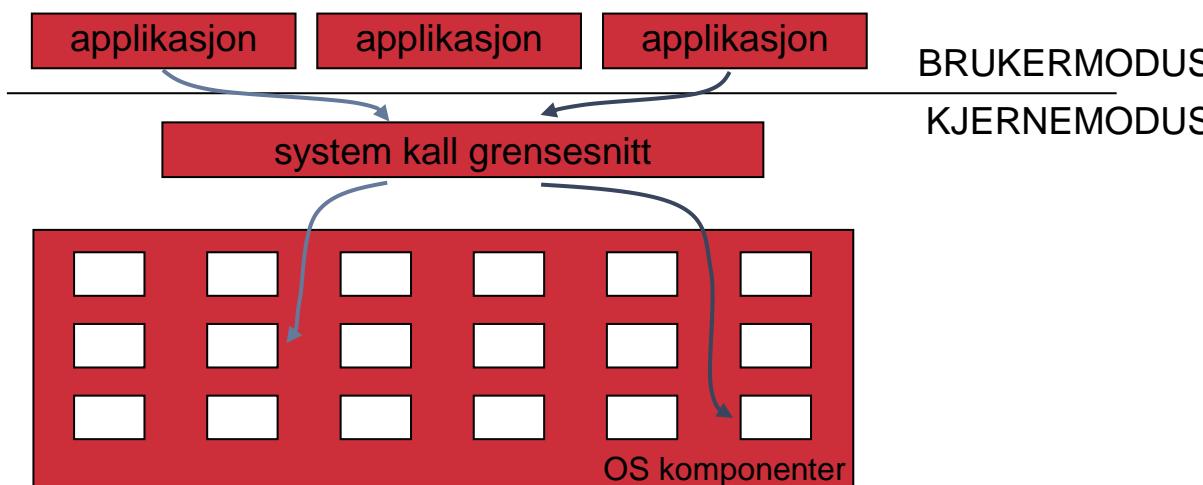
Oppstart (bootstrapping)



Systemkall

Eksempel fra Linux:

- **Grensesnittet** mellom OS og brukerne defineres med en mengde **systemkall**
- Å utføre et systemkall er tilsvarende et vanlig metodekall, men systemkallet kjører kjerne-kode:

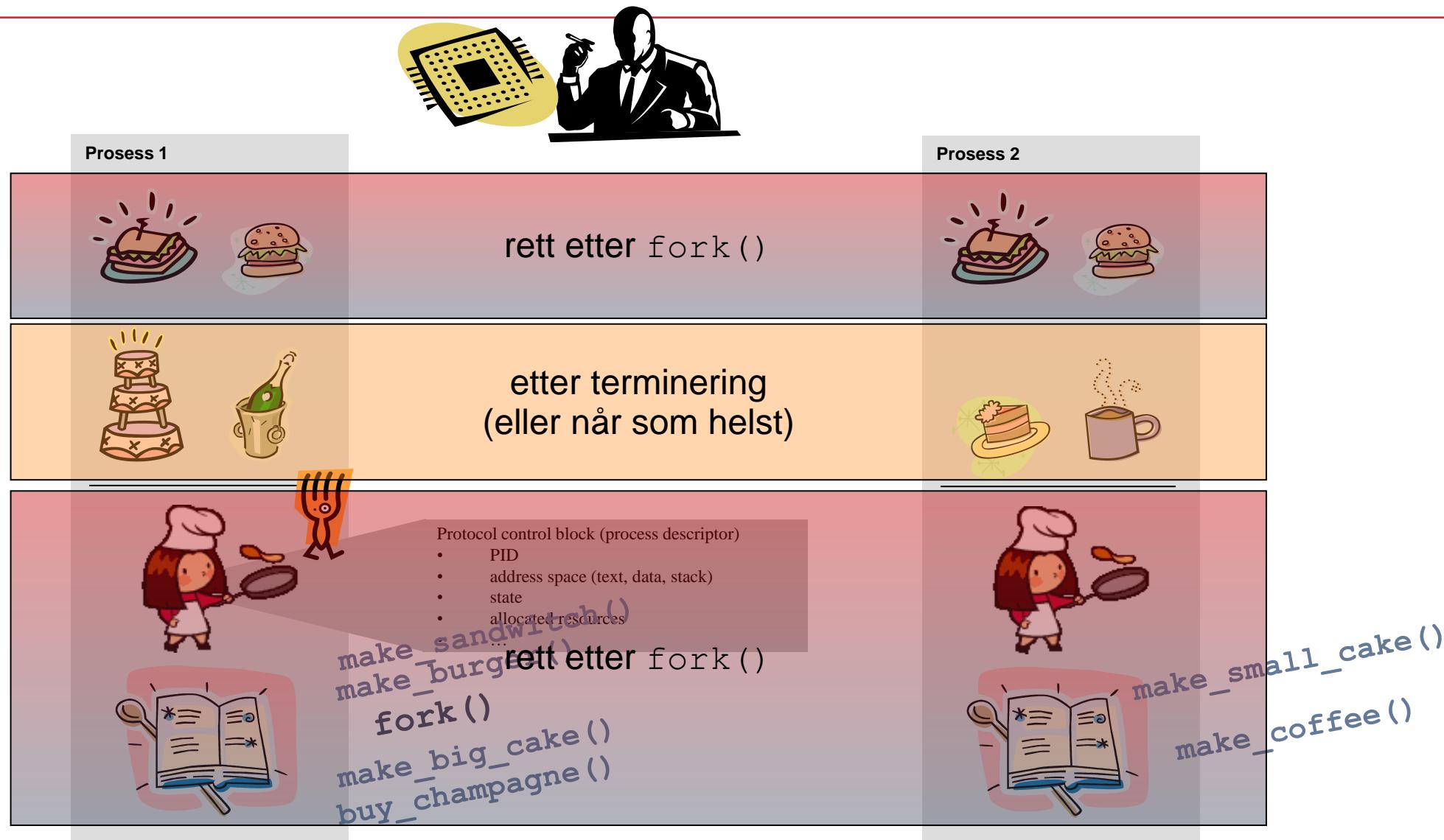


```

sys_acct(const char *name)
sys_acct(const char *filename)
sys_capget(cap_user_header_t header, cap_user_data_t dataptr)
sys_capset(cap_user_header_t header, const cap_user_data_t data)
sys_exit(int error_code)
sys_wait4(pid_t pid,unsigned int * stat_addr, int options, struct rusage * ru)
sys_waitpid(pid_t pid,unsigned int * stat_addr, int options)
sys_futex(void *uaddr, int op, int val, struct timespec *utime)
sys_sysinfo(struct sysinfo *info)
sys_getitimer(int which, struct itimerval *value)
sys_setitimer(int which, struct itimerval *value,
sys_sync(void); /* it's really int */
sys_syslog(int type, char * buf, int len)
sys_nice(int increment)
sys_sched_setscheduler(pid_t pid, int policy,
sys_sched_setparam(pid_t pid, struct sched_param *param)
sys_sched_getscheduler(pid_t pid)
sys_sched_getparam(pid_t pid, struct sched_param *param)
sys_sched_setaffinity(pid_t pid, unsigned int len,
sys_sched_getaffinity(pid_t pid, unsigned int len,
sys_sched_yield(void)
sys_sched_get_priority_max(int policy)
sys_sched_get_priority_min(int policy)
sys_sched_rr_get_interval(pid_t pid, struct timespec *interval)
sys_ni_syscall(void)
sys_setpriority(int which, int who, int niceval)
sys_getpriority(int which, int who)
sys_reboot(int magic1, int magic2, unsigned int cmd, void * arg)
sys_setregid(gid_t rgid, gid_t egid)
sys_setgid(gid_t gid)
sys_setreuid(uid_t ruid, uid_t euid)
sys_setuid(uid_t uid)
sys_setresuid(uid_t ruid, uid_t euid, uid_t suid)
sys_getresuid(uid_t *ruid, uid_t *euid, uid_t *suid)
sys_setresgid(gid_t rgid, gid_t egid, gid_t sgid)
sys_getresgid(gid_t *rgid, gid_t *egid, gid_t *sgid)
sys_setfsuid(uid_t uid)
sys_setfsgid(gid_t gid)
sys_times(struct tms * tbuf)
sys_setpgid(pid_t pid, pid_t pgid)
sys_getpgid(pid_t pid)
sys_getpgrp(void)
sys_getsid(pid_t pid)
sys_setsid(void)
sys_getgroups(int gidsetsize, gid_t *grouplist)
sys_setgroups(int gidsetsize, gid_t *grouplist)
sys_newuname(struct new_utsname * name)
sys_sethostname(char *name, int len)
sys_gethostname(char *name, int len)
sys_setdomainname(char *name, int len)
sys_getrlimit(unsigned int resource, struct rlimit *rlim)
sys_old_getrlimit(unsigned int resource, struct rlimit *rlim)
sys_setrlimit(unsigned int resource, struct rlimit *rlim)
sys_getrusage(int who, struct rusage *ru)
sys_umask(int mask)

```

Prosess opprettes (Unix) – fork ()



Virtuelt minne

- = **paging** **og swapping**
- Programmet kjøres instruksjon for instruksjon
 - Ikke alle instruksjoner trenger å være til stede i minnet, bare de som skal kjøres
- Abstraher minnebruken!
 - Programmet selv tror det har hele minnet (alle adressene) tilgjengelig.
 - Programmet kompileres med interne (**logiske**) minneadresser...
 - Programmet deles opp i sider (**pages**) som bare lastes inn i minnet dersom adressen siden inneholder refereres til.
 - Referanse til en page som ikke er lastet i minnet utløser en **PAGE FAULT**
 - CPU har en **MMU (Memory Management Unit)** som **oversetter** mellom program-interne (**logiske**) adresser og faktiske **fysiske adresser** i RAM (page table)
- Dersom minnet blir for fullt kan minne-sider legges ut på disk (swapping).

Kompilering

```
#include <stdio.h>
int main() {
    printf("Hello World");
    return 0;
}
```

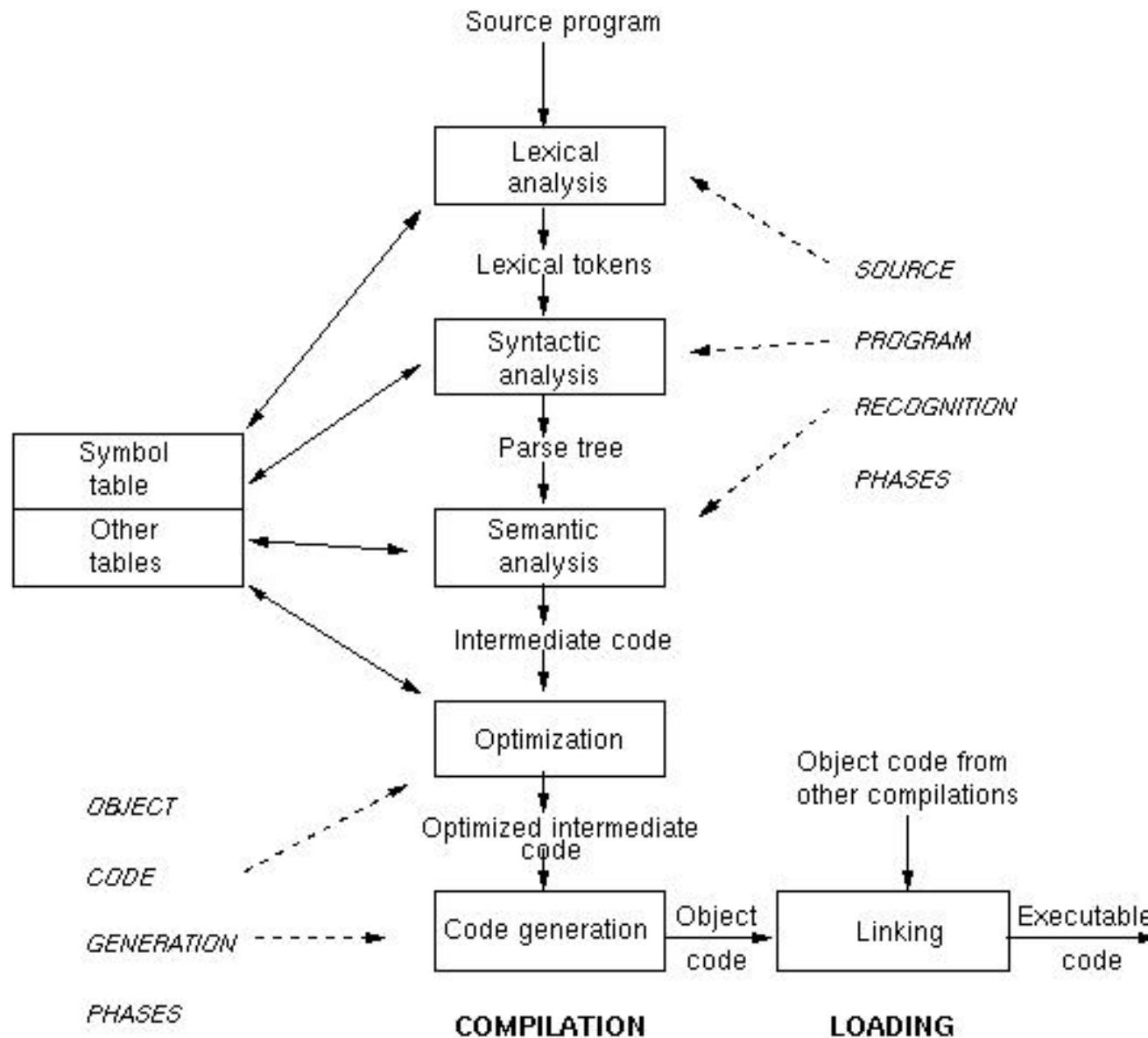
i C

```
MOV AH, 09
MOV DX, 109
INT 21
INT 20
DB "Hello World!","$"
```

Assembly

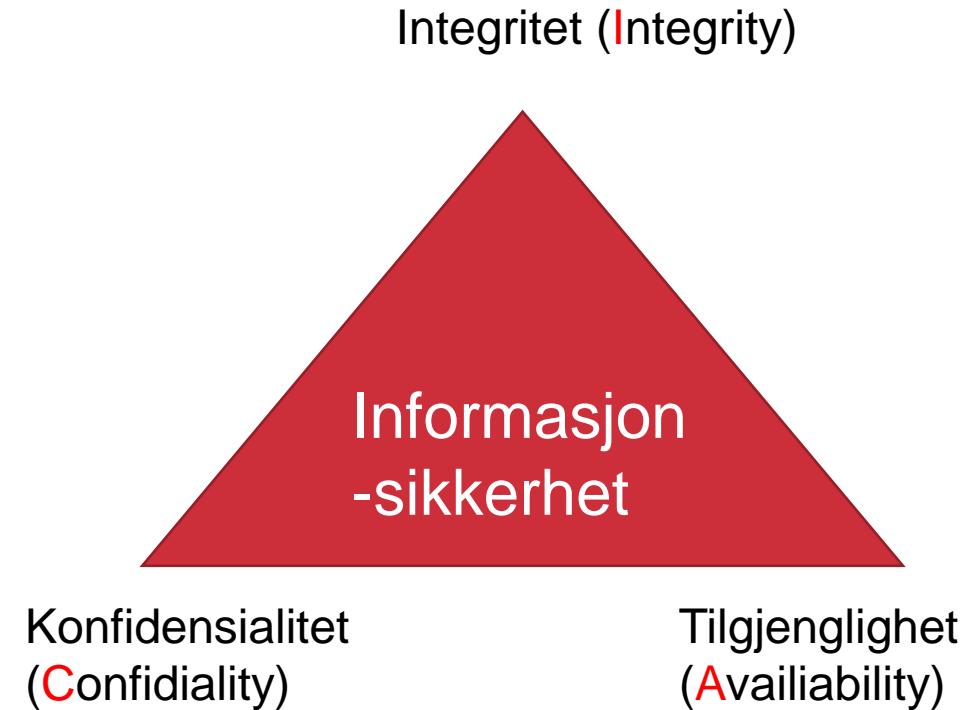
```
B4 09 BA 09 01 CD 21 CD
20 48 65 6C 6C 6F 20 57
6F 72 6C 64 21 24
```

```
...Hello W
orld!$.....4.T.
```



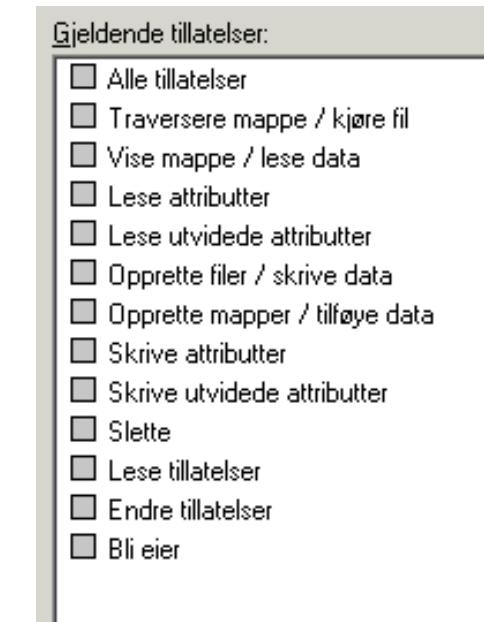
NIVÅER for sikkerhet og tiltak

- Data
 - Access Control Lists, kryptering (av filesystem), ...
- Applikasjon
 - Patching, sertifikater, Anti-virus, ...
- Vertsmaskin
 - OS, autentisering og autorisering, brukeradm, group policies
- LAN (Internt nett)
 - IP-segmentering, IPSec, IDS, ...
- Grense (Perimeter)
 - IDS, Firewall, VPN, NAT, ...
- Fysisk sikring
 - Vakt, låser og overvåking
- Policy, prosedyrer, bevissthet
 - Brukeropplæring



Sikkerhet og filsystemet

- Både Linux/OSX og Windows benytter ulike typer **ACL** (**Access Control Lister**)
- Linux filsystemene deler adgangsrettigheter (**execute**, **read**, **write**) ut fra grupperingen
 - **user**
 - **group**
 - **all**
 - Kataloger (d) er bare en spesiell type bruker
 - Kun brukeren root har alle rettigheter/er administrator
- Windows (NTFS) har et mer finmasket og komplisert system
Vanlige brukere er ofte satt opp som administratorer også



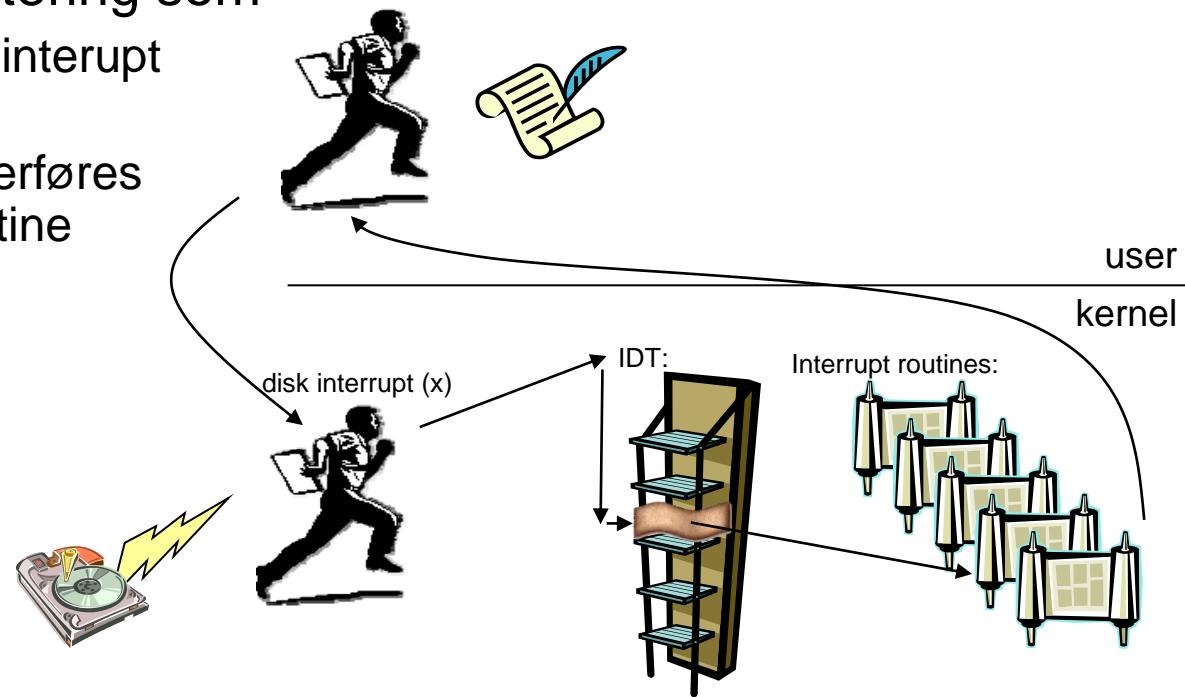
```
-rw-r--r-- 1 blistog users 256 Sep 15 2008 TUBE.COM
-rw-r--r-- 1 blistog users 148480 Sep 11 2008 UTF-8AA.jpg
drwxr-xr-x 3 blistog users 4096 Nov 18 2008 xhtml
```

Operativsystemet – I/O kontroll

- Alt utstyr som er koblet til datamaskinen oppfattes som ressurser av operativsystemet
 - Skjerm, tastatur, mus, CD, høytalere,
- Operativsystemet må vite hvordan datamaskinen skal kommunisere med alt utstyret (**drive****r**e)
 - Plug and play
- Operativsystemet skal presentere utstyret på en enkel og grei måte, også for programmering
 - **UNIX/Linux**: "Lat som" **alt er filer** som kan leses/skrives mot. (lastbare moduler)
 - **Windows**: "Lat som" **alt er objekter**. (komplisert objekt-rom, mange nivåer av drive

Interrupt (og Exception) Håndtering

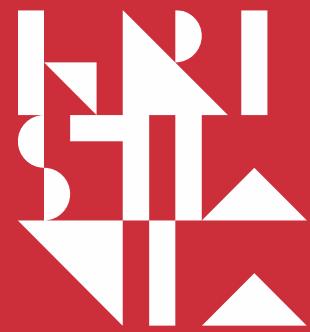
- IA-32 arkitekturen har en IDT med pekere til 256 ulike interrupt and exceptions
 - 32 (0 - 31) forhåndsdefinert og reservert
 - 224 (32 - 255) bruker/OS-definert
- Hvert interrupt har en en peker i Interrupt tabellen (IDT) og en unik index verdi som gir håndtering som
 - 1. prosess kjører, det kommer et interrupt
 - 2. bevar kjøretilstand, kontroll overføres og finn interrupt-håndteringsrutine
 - 3. Eksekver interrupt-håndtering
 - 4. hent tilbake avbrutt prosess
 - 5. fortsett eksekvering



Denne forelesningsøkten vil bli tatt opp og lagt ut i emnet i etterkant.

Hvis du ikke vil være med på opptaket:

 ^ Start Video	La være å delta med webkameraet ditt.
 ^ Unmute	La være å delta med mikrofonen din.
To: Marianne Sundby (Privately) Type message here...	Still spørsmål i Chat i stedet for som lyd. Hvis du ønsker kan spørsmålet også sendes privat til foreleser.



Høyskolen
Kristiania

TK1100

Internett og WWW
6'te forelesning

Status

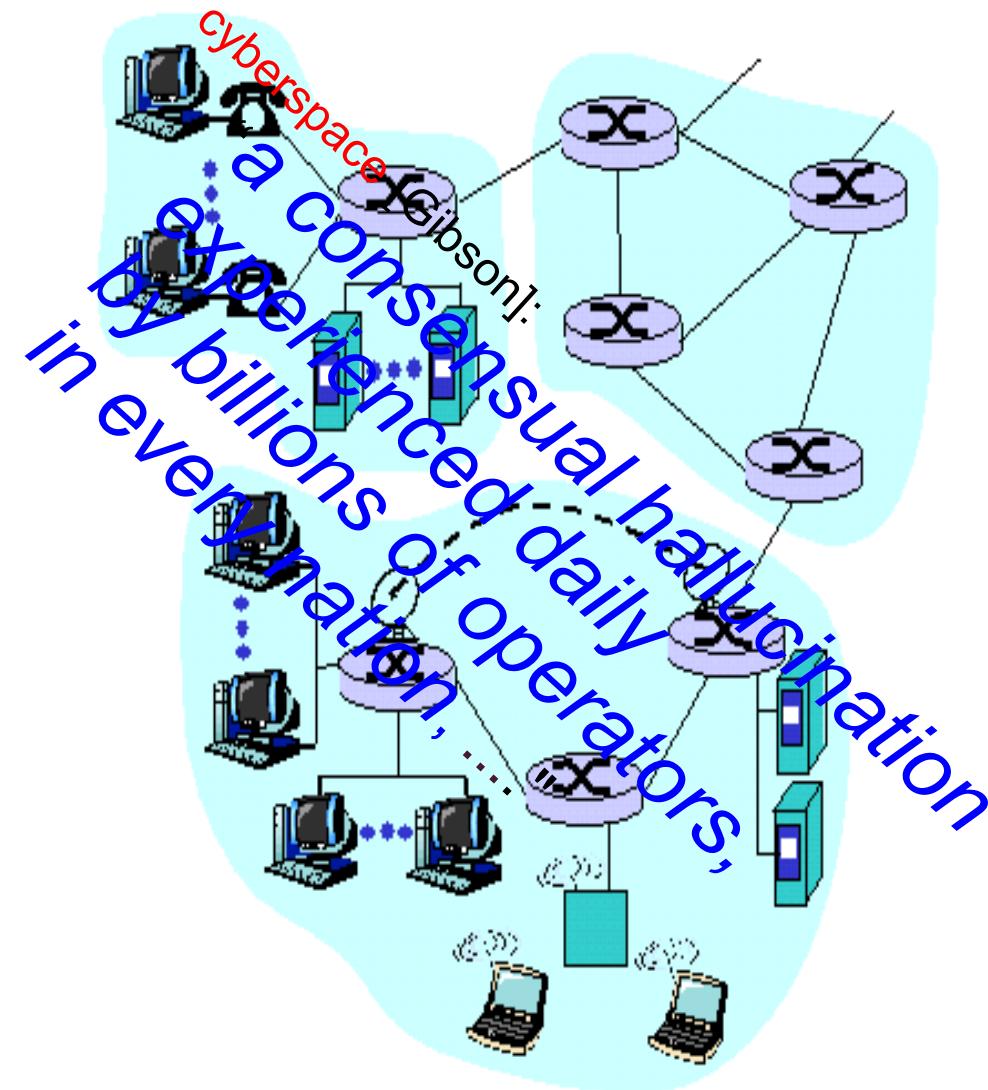
- Vi har gått gjennom:
 - Oppbygging
 - Von Neumann modellen
 - Hardware
 - Logiske porter
 - CPU
 - Hovedkort og busser
 - Periferiutstyr
 - Software
 - Datatyper (binære)
 - Instruksjoner (typer)
 - Filformater (noen få)
 - Operativsystem
 - Oppgave
 - Oppbygging
 - Virkemåte

- Virkemåte:
 - Digital representasjon
 - Tall (hel- og flyt-)
 - Text (kodeskjema)
 - Bilde (bitmap+vektor)
 - Sekvensiell kjøring av instruksjoner
 - Adressering i RAM
 - Aritmetiske og logiske operasjoner
 - Shell

- **Så langt** har vi sett på hvordan enkeltmaskiner fungerer på ulike nivåer
 - Fra logisk port
 - Opp til systemprogramvare
 - I **Programmering** ser dere på hvordan man lager **programmer** som kjører på hardware
 - I **Databaser** ser dere på hvordan **data** organiseres og gjøres brukbare
 - I **Web prosjekt** kommer dere til å se på hvordan **presentere** informasjon i **web-sider**
 - men hvordan får vi **distribuert** dem – hvordan får en bruker sett på dine web-sider?
- **I dag**
 - Hvordan kan maskiner **samarbeide** over **datanett**?
 - Svaret er **protokoller** og **standarder**.

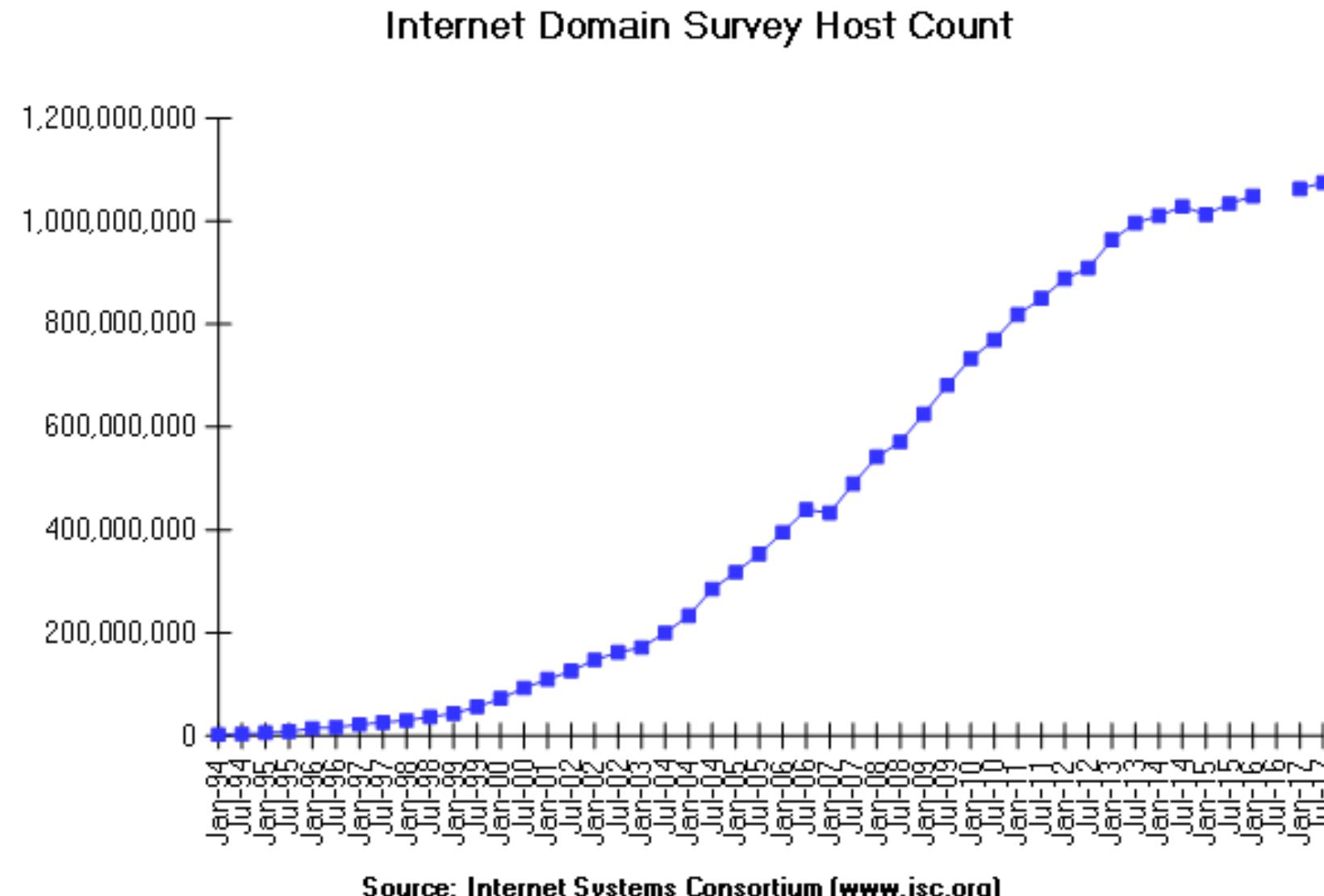
Introduksjon

- Hva er Internett?
 - Milliarder av computere i nett
 - Nettet tilgjengelig for "alle"
 - Enkle brukergrensesnitt
 - Ikke fullt så enkelt bak kulissene
 - Et sett med **standarder** for nettverks-kommunikasjon som sammenkopler ulikartede LAN og WAN ("TCP/IP-stacken")

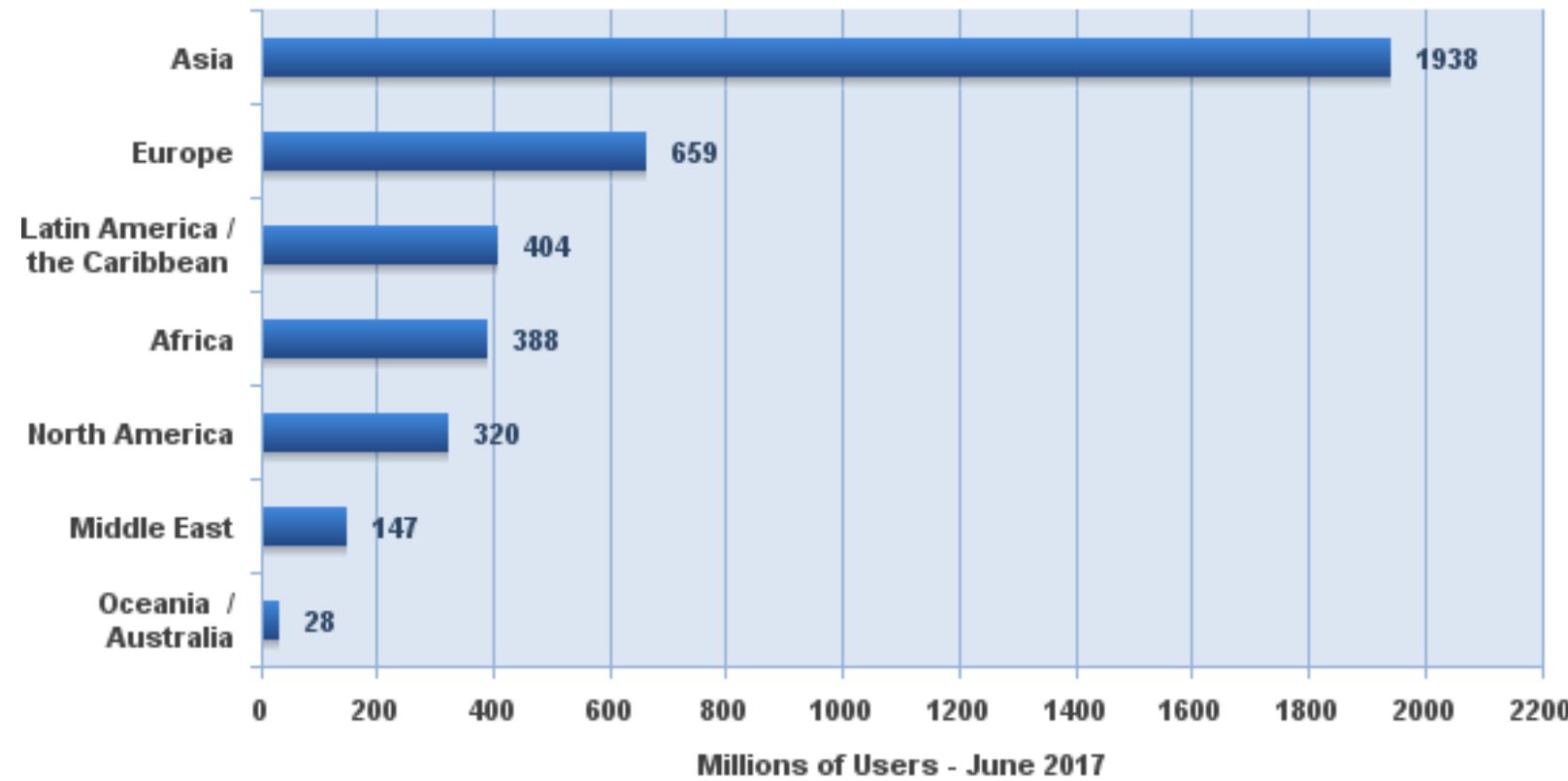


- Teknisk **Infrastruktur** som kopler sammen ulike nettverk ved hjelp av TCP/IP-suiten av **protokoller**
- **WWW** er IKKE det samme som Internett!!!
 - Uansett om det har blitt vanlig språkbruk i Norge og andre steder.
 - WWW er **en applikasjon** levert med HTTP

Antall domener



Internet Users in the World by Geographic Regions - June 30, 2017



Source: Internet World Stats - www.internetworldstats.com/stats.htm

Basis: 3,885,567,619 Internet users estimated in June 30, 2017

Copyright © 2017, Miniwatts Marketing Group

- ... er normer og krav til tekniske systemer
- Nedskrevet i et dokument som beskriver hvilke krav man må oppfylles for å tilfredsstille standarden.
- *De Jure* (etter lov) vs *De Facto* (faktisk)
 - Standarder som følges ...
 - fordi «alle» finner det fordelaktig kalles **De Facto**.
 - fordi det kreves (typisk i lov) kalles **De Jure**
 - Eksempel:
 - MP3 i stedet for CD Wav på nettet (De facto)
 - HTML frem til 1995 (De facto -> De jure)
 - Microsoft Word DOC (De facto -)

RFC'er som
standardiserings-
prosess

- Request for Comment

IANA, RIPE m.fl. adm.
(DNS-)NAVN og (IP-
)NUMMER

Internet Society (ISOC)

Provides leadership in issues relating to the Internet and serves as the organizational home of groups responsible for Internet standards.

Internet Engineering Task Force (IETF)

Develops and disseminates the standards and protocols that define the architecture and software of the Internet.

Internet Architecture Board (IAB)

Defines the overall architecture of the Internet, providing guidance and broad direction to the IETF.

Internet Engineering Steering Group (IESG)

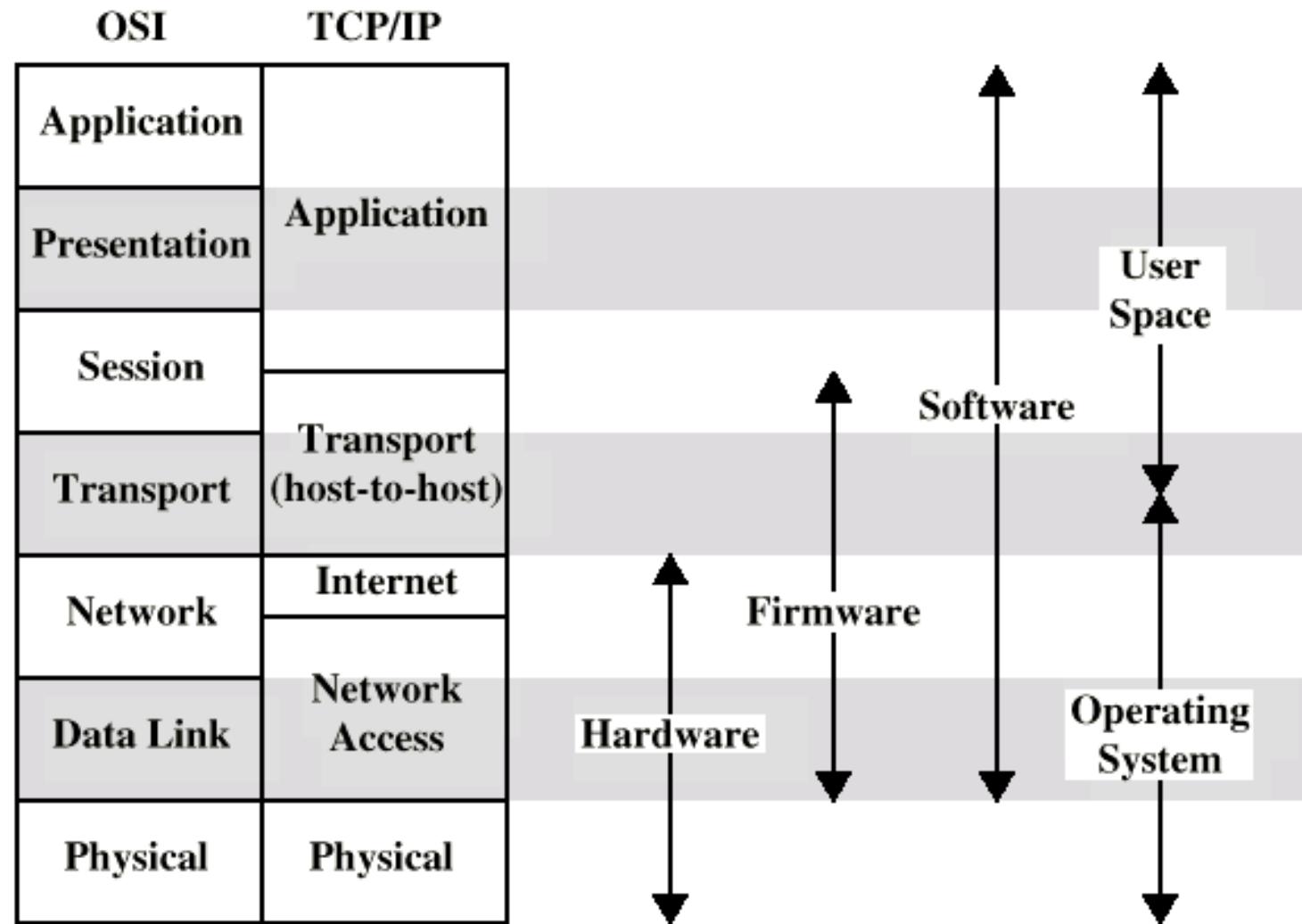
Manages the technical activities of the IETF and oversees the process of standards development.

Internet Research Task Force (IRTF)

Organizes research groups on topics related to Internet protocols, applications, architecture, and technology.

OSI vs TCP/IP

- Hvilke modeller (arkitektur) brukes?



Kronologi

Internett historikk: Forspill

1961

- Kleinrock: **Pakkeswitching** som prinsipp

1964

- Baran: Pakkeswitching i militære nett

1967

- ARPAnet (Advanced Research Project Agency) unnfangenet

1969

- Første **ARPAnet** node operativ

1972

- ARPAnet demonstrert med 15 noder, NCP, Mail
- **Norge** tilknyttes

Internett historikk: 70-tallet

1970

- ALOHAnet på Hawaii

1973

- Metcalfe: Ethernet

1974

- Cerf & Kahn: Nettverksarkitektur, Internetts "fedre": «Ende-til-ende prinsippet»

1976

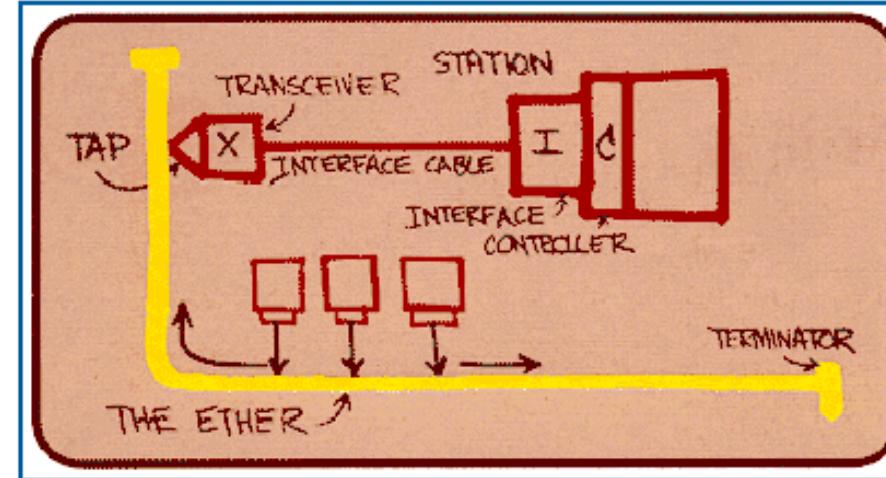
- UNINETT blir etablert for å levere Internett til norske universitet

1977-79

- Proprietære arkitekturen, SNA, XNA
- Pakkeswitching på fast lengde (forløper til ATM)

1979

- ARPAnet har 200 noder



Cerf and Kahn's internetworking prinsipper:

- minimalisme, autonomi – krever ingen indre endringer for å kople sammen netteverk
- **Best effort** service model
- **Tilstandsløse** routere
- **Desentralisert** kontroll

definerer dagens Internet arkitektur

Internett historikk: 80-tallet

1982

- **SMTP** protokoll (**email** sending) definert

1983

- **TCP** blir off. standardprotokoll for ARPAnet

1988

- TCP **trafikk-kork kontroll**, **IPv4**, UDP

1989

- 100 000 verter i totalt nettverk

Internett historikk: 90-tallet

1991

- Berners-Lee: HTML og HTTP

1994

- Mosaic, senere Netscape

1994

- ARPAnet «nedlagt», NSFnet overtar, kommersialiseringen starter

2000

- 600 millioner Internett-brukere
- Mer enn 1 milliard tilgjengelige sider på WWW

2008

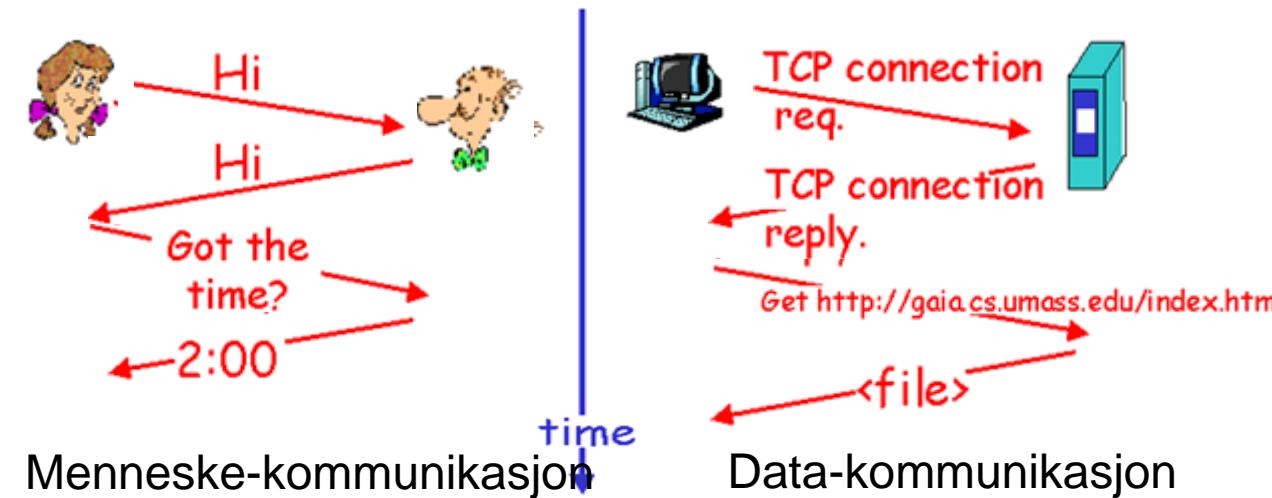
- 1,2 milliarder brukere fordelt på 500 millioner hosts

Protokoller (og andre begrep)

Protokoll

Hva er en protokoll?

- Brukes til kommunikasjon mellom "like" funksjoner
- Må snakke det samme "språk"
- Funksjoner
 - Bruker-applikasjoner
 - E-mail
 - Terminaler
- Systemer
 - Computere
 - Terminaler
 - Sensorer



Protokoll

- Nøkkel-elementer for en **protokoll**
 - **Syntaks**
 - "ord, ordstilling og grammatikk"
 - Data-formater
 - Signal-nivåer
 - **Semantikk**
 - "betydning"
 - Kontroll-informasjon
 - Feil-behandling
 - **Timing**
 - Hastighets-tilpasning
 - Sekvens-tilordning

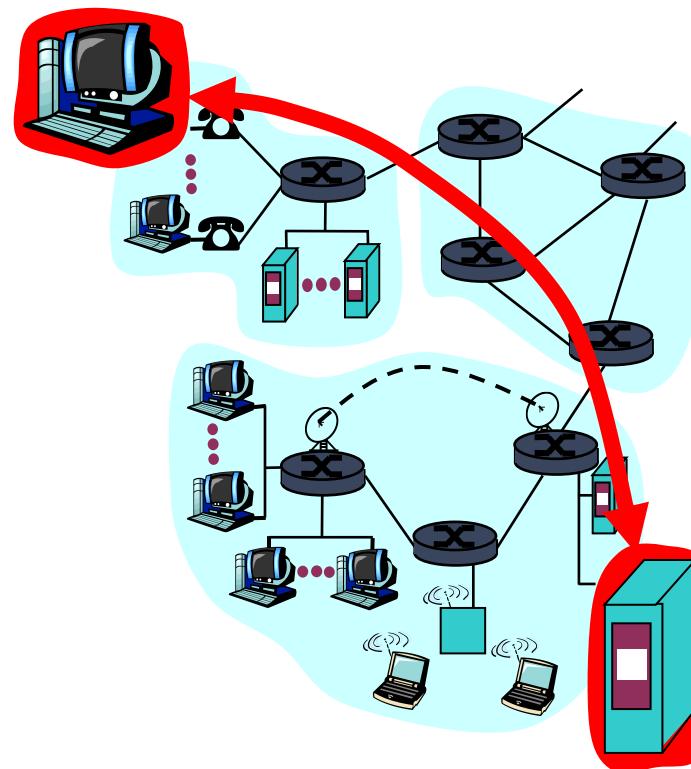
Nettverkets ytterkanter

Ende-systemer (verter = hosts)

- Kjører **applikasjoner**
 - WWW, telnet, email

Tjenestemodeller:

- **Klient-tjener** modell
 - Forespørslar kommer fra klient som mottar service fra tjener
 - F.eks. WWW browser/tjener
- Peer-peer (P2P) modell
 - Alle computere er likeverdige
 - F.eks. BitTorrent, Skype,...

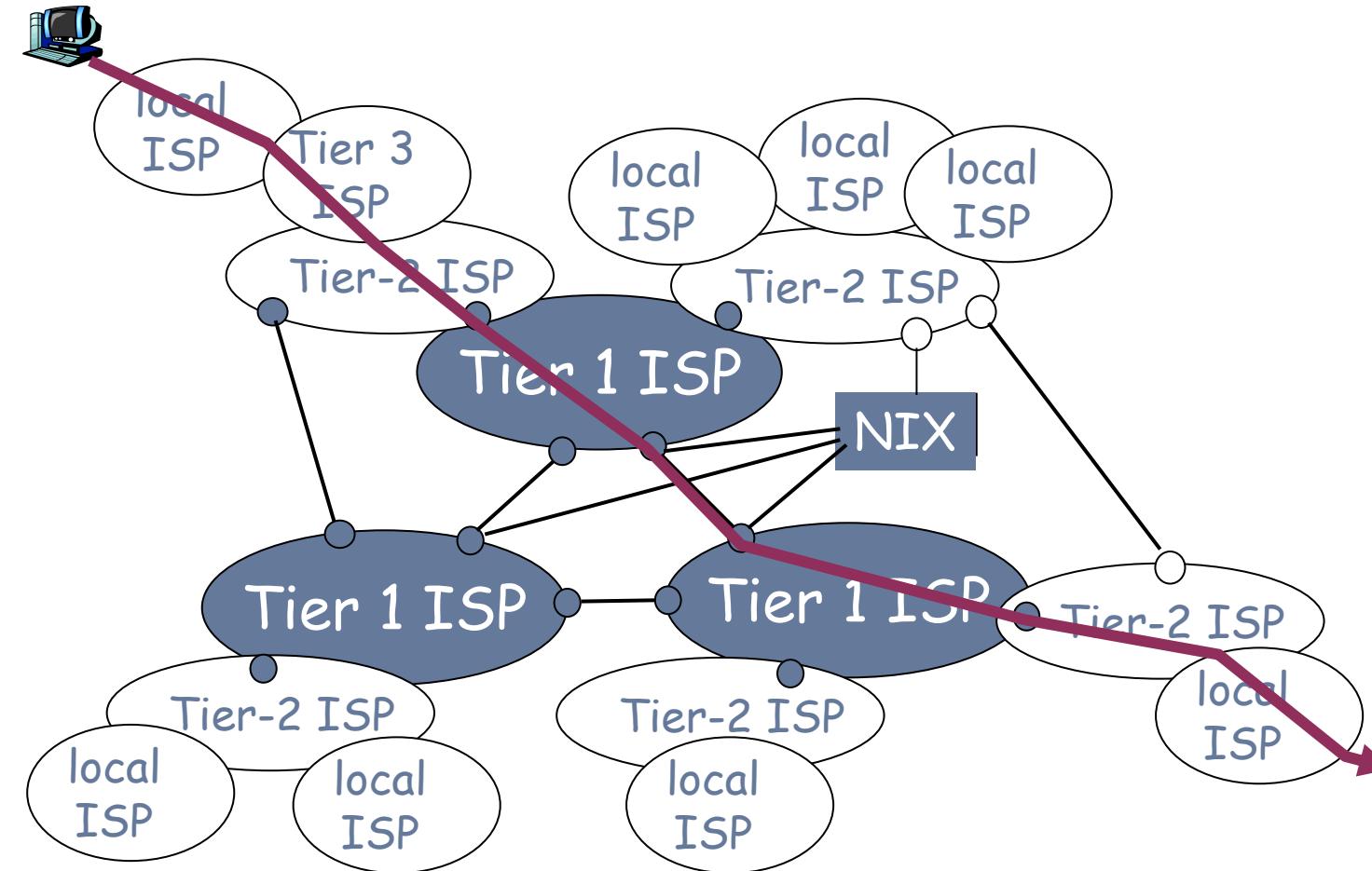


Nettverkets ytterkanter

- **Forbindelses-orientert** service
 - Oppretter kontakt (**handshake**) før data overføres
 - Etablerer overførings-**tilstand** i endesystemene
- **TCP** (Transmission Control Protocol)
 - Internetts mest brukte forbindelses-service
 - Pålitelig og ordnet dataoverføring
 - Service ved tap av data
 - Flytkontroll mot oversvømmelse
 - Reduserer farten på avsender ved stor trengsel på nettet (metningskontroll)

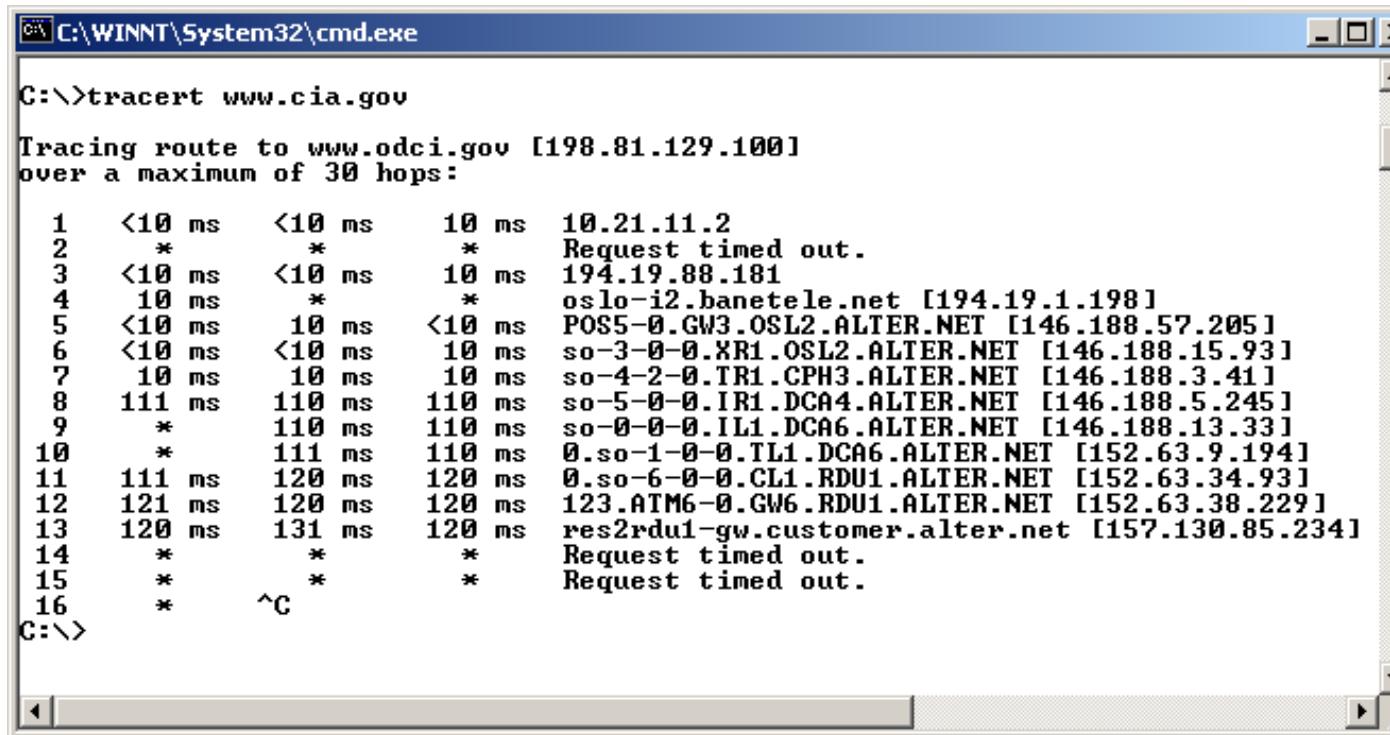
Nettverk av nettverk!

- Internett var opprinnelig laget for å kople sammen ulike typer lokalnettverk
- En datapakke passerer altså (ofte) gjennom mange ulike typer nettverk!



TRACEROUTE APPLIKASJONEN

- Viser hvilke **routere** en pakke går gjennom
- Kan måle forsinkelsene mellom nodene med applikasjonen traceroute (Windows: tracert)
- Kan bli stoppet av brannmurer o.l.



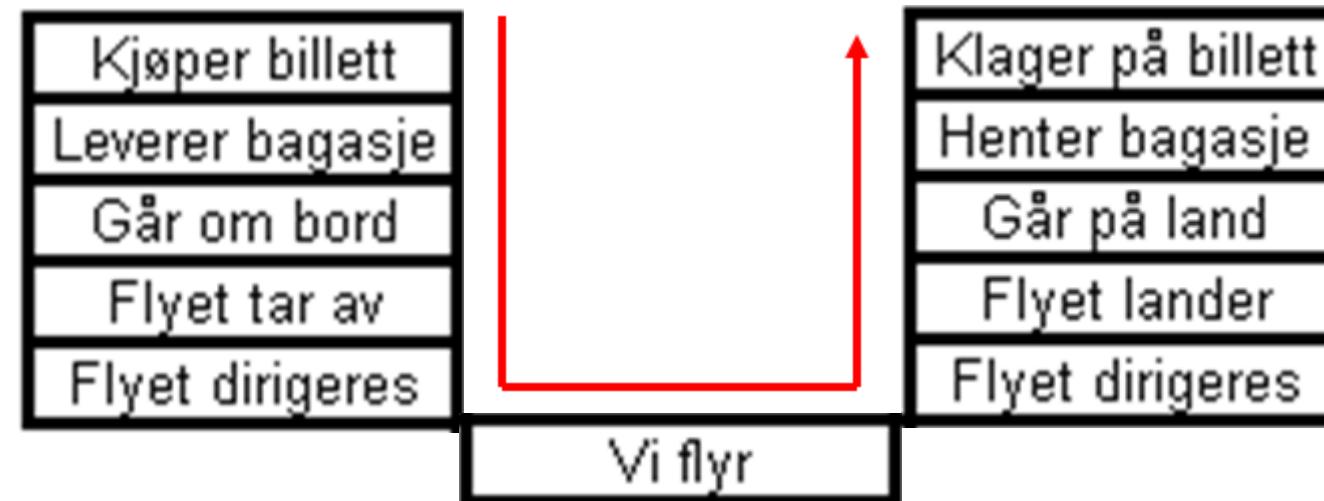
```
C:\>tracert www.cia.gov

Tracing route to www.odci.gov [198.81.129.100]
over a maximum of 30 hops:
1 <10 ms <10 ms 10 ms 10.21.11.2
2 * * * Request timed out.
3 <10 ms <10 ms 10 ms 194.19.88.181
4 10 ms * * oslo-i2.banetele.net [194.19.1.198]
5 <10 ms 10 ms <10 ms POS5-0.GW3.0SL2.ALTER.NET [146.188.57.205]
6 <10 ms <10 ms 10 ms so-3-0-0.XR1.0SL2.ALTER.NET [146.188.15.93]
7 10 ms 10 ms 10 ms so-4-2-0.TR1.CPH3.ALTER.NET [146.188.3.41]
8 111 ms 110 ms 110 ms so-5-0-0.IR1.DCA4.ALTER.NET [146.188.5.245]
9 * 110 ms 110 ms so-0-0-0.IL1.DCA6.ALTER.NET [146.188.13.33]
10 * 111 ms 110 ms 0.so-1-0-0.TL1.DCA6.ALTER.NET [152.63.9.194]
11 111 ms 120 ms 120 ms 0.so-6-0-0.CL1.RDU1.ALTER.NET [152.63.34.93]
12 121 ms 120 ms 120 ms 123.ATM6-0.GW6.RDU1.ALTER.NET [152.63.38.229]
13 120 ms 131 ms 120 ms res2rdui-gw.customer.alter.net [157.130.85.234]
14 * * * Request timed out.
15 * * * Request timed out.
16 * ^C
```

Oppdeling av nettverk

- Nettverket består av mange deler
 - Verter
 - Routere
 - Linker til forskjellige media
 - Applikasjoner
 - Protokoller
 - Programvare
 - Maskinvare
- Hvordan skal vi få organisert alt dette?
 - Splitt og hersk!
 - Del opp i «abstraksjonsnivåer» ut fra funksjonalitet

Ex: Organisering av flytur



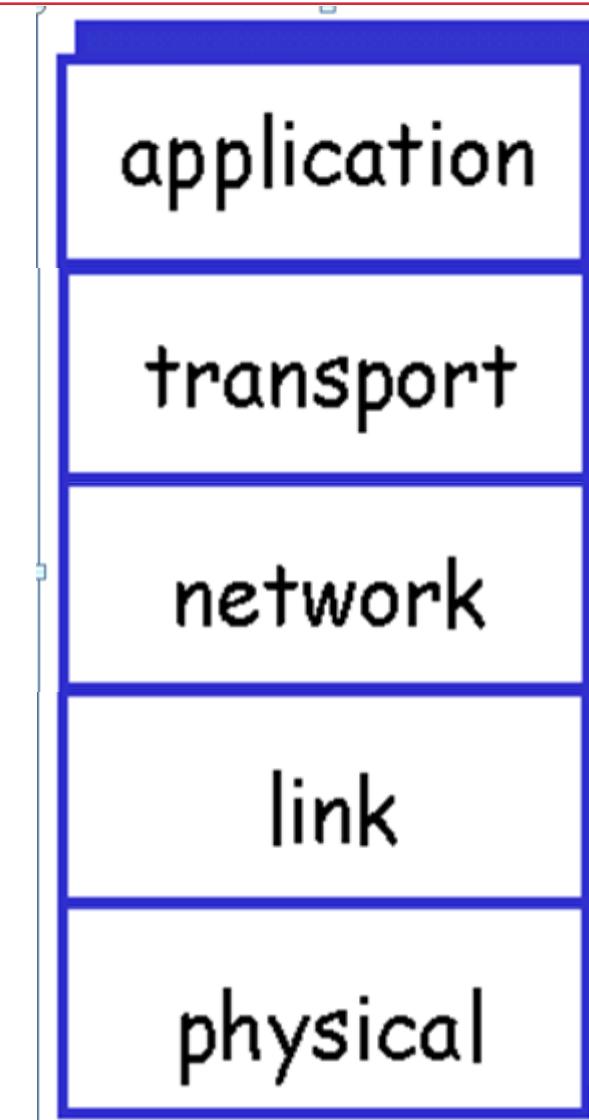
- **Vi abstraherer:**
 - Hva er felles funksjonalitet?
 - Vi definerer regler (protokoller) for hvordan ting skal foregå på hvert trinn/nivå



Organisering på Internett

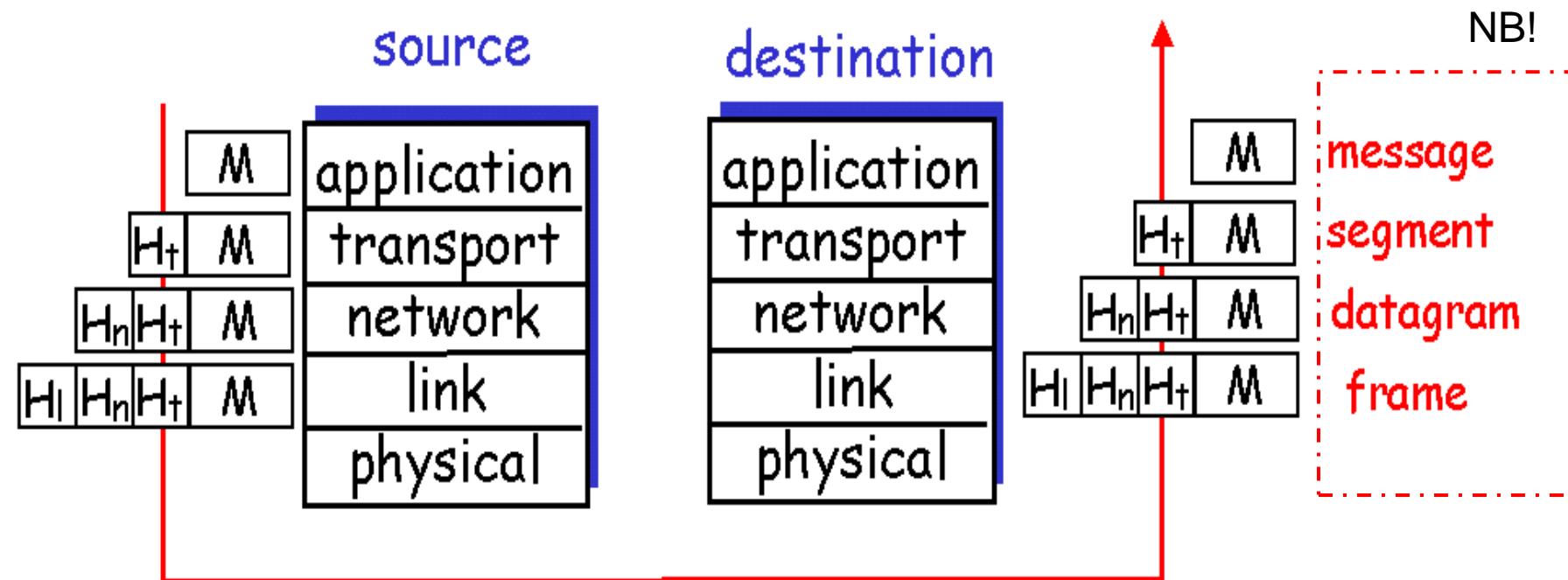
Internet protocol stacken

- Lag for nettverksapplikasjoner
 - FTP, SMTP, HTTP, ...
- Lag for **transport** mellom verter/prosesser
 - TCP, UDP,..
- Lag for **nettverks-ruting** ende-til-ende
 - IP, ICMP, RIP..
- Lag for overføring nabo-til-nabo
 - Ethernet
- Lag for fysisk overføring
 - Kabling, plugger, signalnivå



Inn/ut- pakking av data

- Avsender: Hvert lag tar data fra laget ovenfor
 - Legger til informasjon (header), lager ny dataenhet
 - Leverer nye data til laget nedenfor
- Mottaker prosesserer data i motsatt rekkefølge



- Internett var designet for et minimum av **pålitelighet** og med svært lite tanke på **sikkerhet**
 - Har blitt den viktigste spredningsvektoren for **malware**
- Nettverkssikkerhet
 - Hvordan beskytte kommunikasjon mot å bli avlyttet, utnyttet eller "krasjet"
 - Virus, ormer, trojanske hester, spyware, spam,...
 - Denial of Service Angrep (DOS)

Kritiske tjenestenivåer for applikasjoner

- **Tap av data**
 - Noen applikasjoner **tåler litt** tap av data
 - Audio, video
 - Andre må ha 100% pålitelig dataoverføring
 - Filoverføring
- **Båndbredde/bitrate (bps)**
 - Noen applikasjoner må ha en viss båndbredde
 - Multimedia
 - Andre kan bruke båndbredden dynamisk
 - Filoverføring
- **Timing**
 - Noen applikasjoner tåler ikke mye tidsforsinkelse/latens
 - Sanntidsprosesser, spill

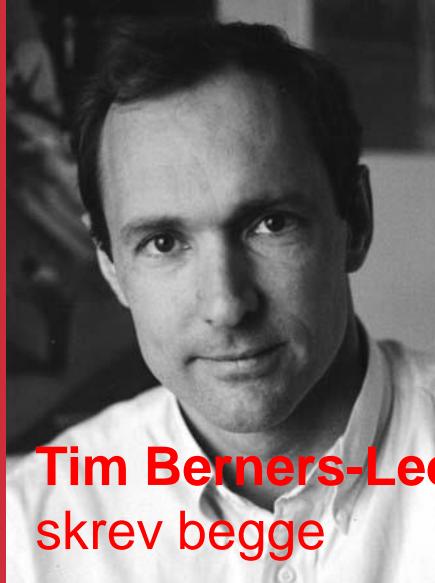
BITRATE («båndbredde»)

- Båndbredde heter egentlig **bitrate** på norsk
 - Båndbredde er en relatert størrelse som måles i Hz (signalbehandling)
 - Uttrykk for den **teoretiske** dataoverførings-kapasiteten på en **link**
 - $1 \text{ Mbps} = 1\ 000\ 000 \text{ b/s}$
 - Merk at dette ikke er **Mi**, men «**m**»
 - Cirka 108 KiB per sekund
- **Latens** = «responstid» er ofte like viktig i forhold til **opplevelse** av ytelse...
- Pakketap er også en viktig faktor for **opplevelse** av ytelse, mer om det senere, men tips; sitter du på en fiber linje med høy hastighet og opplever treghet i online spill? Trekk en fysisk eternett kabel istedefor å bruke Wifi ☺

<http://info.cern.ch/hypertext/WWW/TheProject.html>

var den første «hjemmesiden»
publisert 6. August 1991

WORLD WIDE WEB



- Noen sentrale protokoller og begreper

Klient/tjener

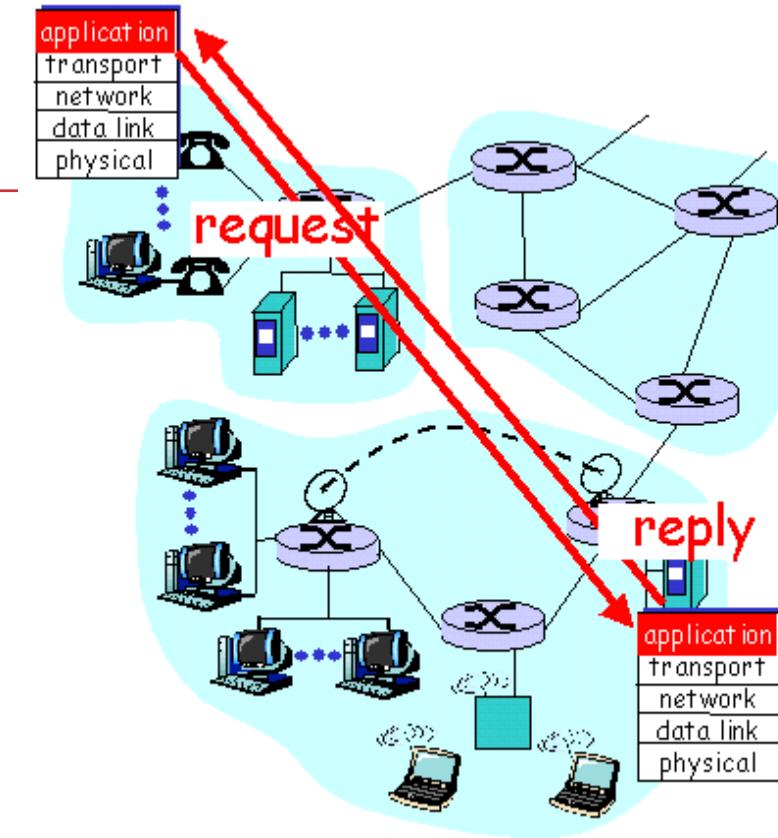
- Typisk oppsett i et nettverk

Klient

- Tar initiativet
- Ber om en service/tjeneste fra tjeneren
- På web er klienten i **browseren**

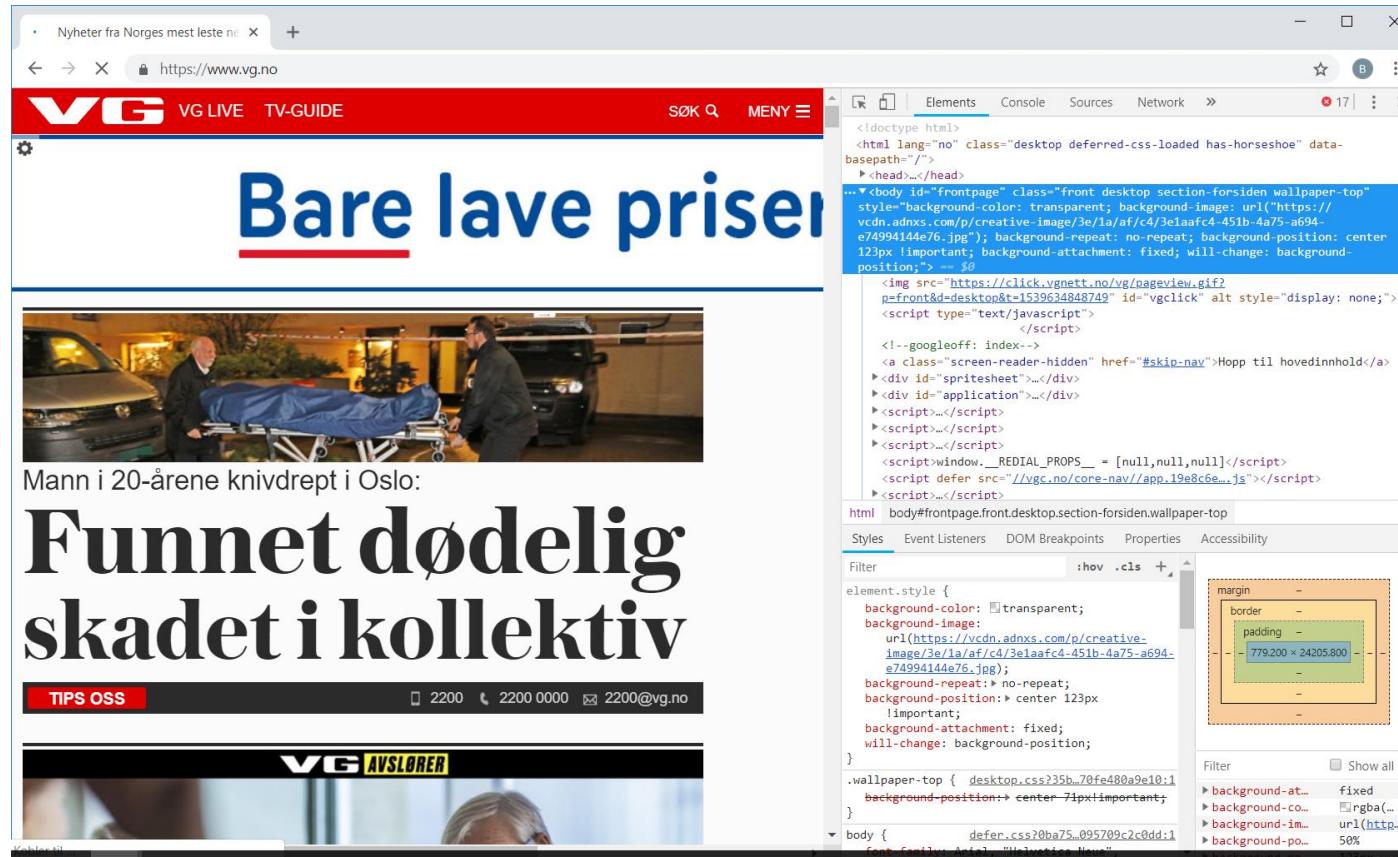
Tjener

- Leverer etterspurt service til klienten
- På web er dette **web-serveren**



- Du **ser** svært lite av hva som skjer «under panseret» i browseren.
- Det du oppfatter som «en side» er resultatet av mengdevis av enkeltutvekslinger av forespørsler og svar
 - F.eks. er hvert enkelt bilde i siden resultatet av en separat forespørsel (GET) og svar (200 OK + fil innhold)
- For å kunne feilsøke må vi hele tiden huske hva som egentlig foregår...

Chrome: Ctrl-Shft-I (Developer Tools)



The screenshot shows a news article from VG.no. The headline reads "Bare lave priser" (Low prices) and the sub-headline "Mann i 20-årene knivdrept i Oslo" (Man in his 20s stabbed in Oslo). Below the headline is a photograph of a man on a stretcher. The footer has a "TIPS OSS" button and contact information: 2200, 2200 0000, and 2200@vg.no. The developer tools Network tab is open, showing the loading of a CSS file for the wallpaper: `https://cdn.adnxs.com/p/creative-image/3e/1a/af/cd/3e1aaafc4-451b-4a75-a694-e74994144e76.jpg`. The Styles tab shows the CSS for the `.wallpaper-top` class.

```

<!doctype html>
<html lang="no" class="desktop deferred-css-loaded has-horseshoe" data-basepath="/">
  <head></head>
  <body id="frontpage" class="front desktop section-forsiden wallpaper-top" style="background-color: transparent; background-image: url("https://cdn.adnxs.com/p/creative-image/3e/1a/af/cd/3e1aaafc4-451b-4a75-a694-e74994144e76.jpg"); background-repeat: no-repeat; background-position: center 123px !important; background-attachment: fixed; will-change: background-position;">
    
    <script type="text/javascript">
      </script>
      <!--googleoff: index-->
      <a class="screen-reader-hidden" href="#skip-nav">Hopp til hovedinnhold</a>
    <div id="spritesheet"></div>
    <div id="application"></div>
    <script></script>
    <script></script>
    <script></script>
    <script>window.__REDIAL_PROPS__ = [null,null,null]</script>
    <script defer src="//vgc.no/core-nav/app_19e8c6e...js"></script>
    <script></script>
  </body>

```

```

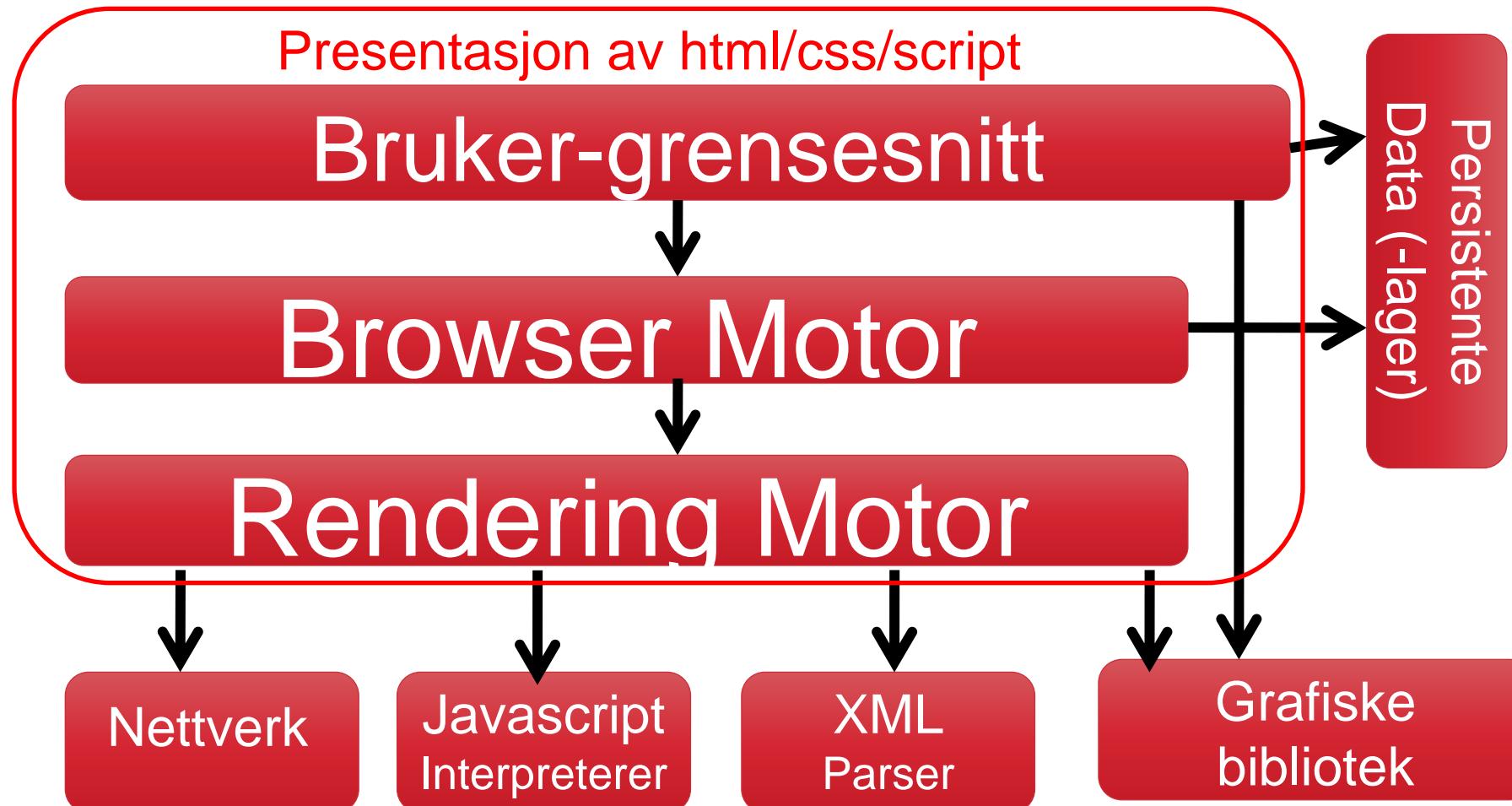
.wallpaper-top { desktop.css?35b_70fe480a9e10:1
  background-position: center 71px !important;
}
body { defer.css?0ba75_095789c2c0dd:1
  font-family: Arial, "Helvetica Neue", Helvetica, sans-serif;
}

```

- Ved sjekke under **Network** får du se mer av hva som har foregått før siden endelig ble rendret i browseren

Hva er en browser/nettleser?

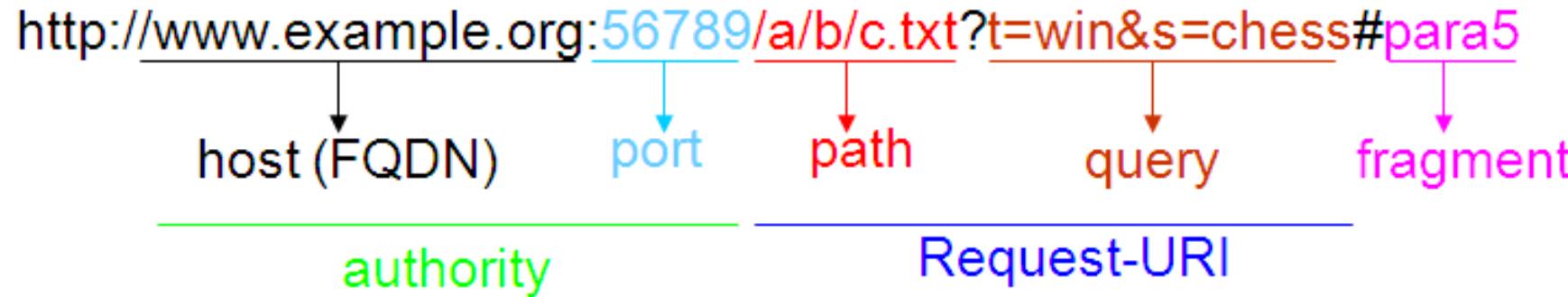
- Programvare som kjører klient-delen av en web-applikasjon



Service i **transportlag** protokoller

- **Transportlaget** sørger for kommunikasjon mellom prosesser som kjører på vertsmaskiner («hosts»)
- **TCP**
 - Forbindelsesorientert
 - Pålitelig transport
 - Flytkontroll
 - Metnings-kontroll
 - **Ikke** timing kontroll eller minimumsgaranti for båndbredde
- **UDP**
 - Lettvektsprotokoll uten garantier

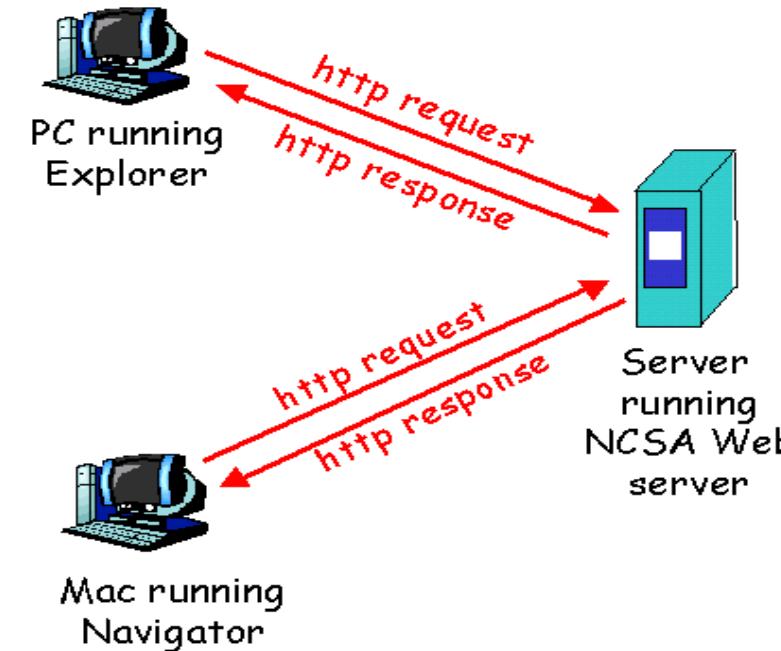
- Web-side
 - Består av "objekter", adresseres av en URI
- Vanligvis har web-siden
 - En base HTML side (index.html), flere objektreferanser
- URI (url) består av
 - protokoll://bruker:passord@vertsnavn:port/filsti/filnavn#anker?parametre
(protocol://user:pwd@host:port/path?parameters#anchor)
 - http://test.kristiania.no/TK1100/ressursnavn.txt
- Bruker-agenten på web er en **browser**
 - Netscape, Internet Explorer, Mozilla
- Tjeneren på web kalles **web-server**
 - Apache, MS IIS



- 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") ()

HTTP (HyperText Transfer Protocol)

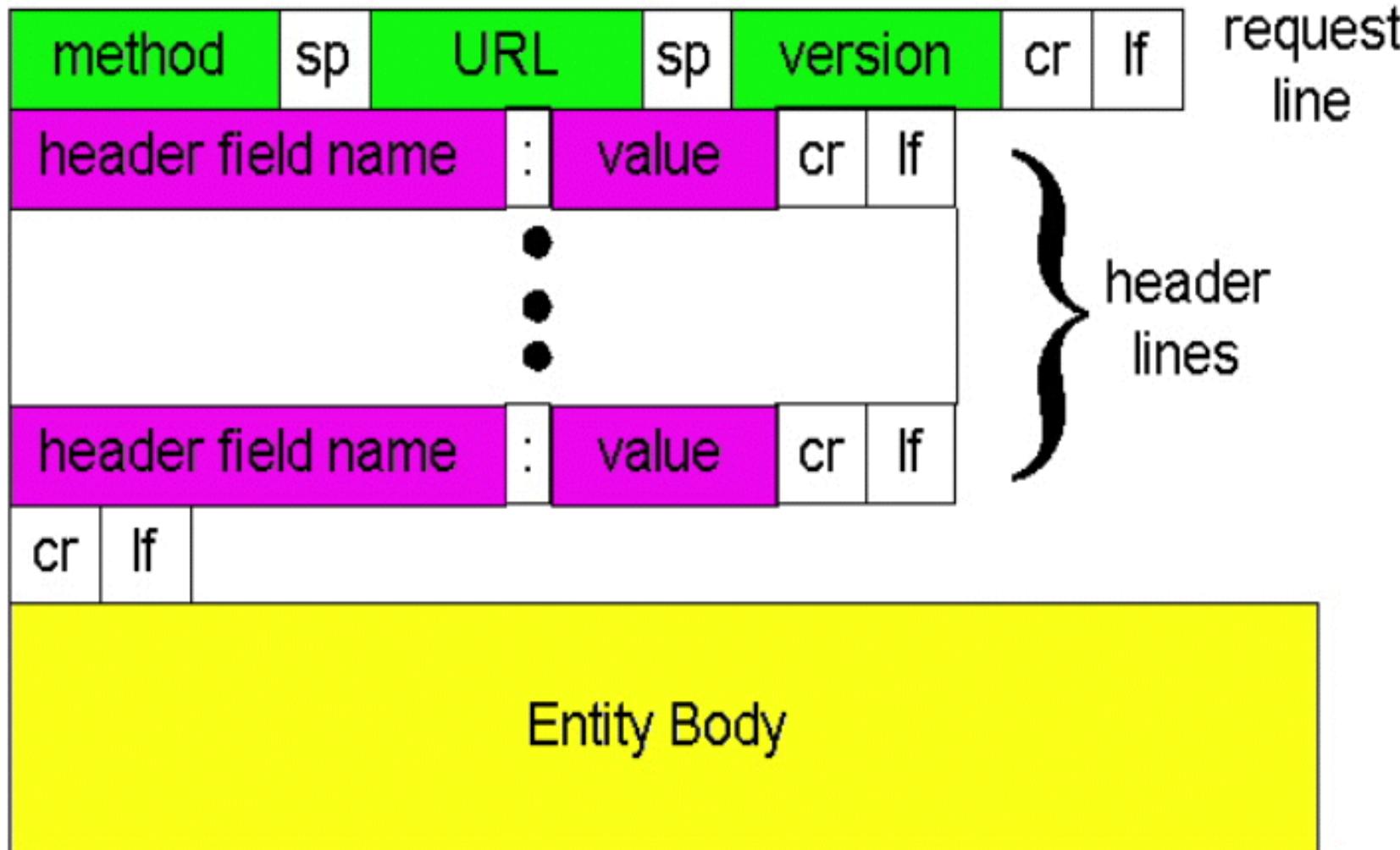
- Webens applikasjons-protokoll
- Klient/tjener modell
 - Klienten spør etter, mottar og viser web "objekter"
 - Tjeneren sender objekter på etterspørsel



- Bruker **TCP** transport service
 - Klient setter opp TCP-forbindelsen (lager socket) til tjeneren, port **80**
 - Tjeneren godtar forbindelsen til klienten
 - Klient og tjener utveksler infomasjon
 - TCP-forbindelsen lukkes
- HTTP oppbevarer ingen informasjon om tidligere forespørsler
(tilstandsløs = stateless)

HTTP meldingsformat: spørring

- Meldings**headere** er kodet i 7 bit **ASCII**-format



HTTP 1.0 meldingsformat

- Spørring

request line
(GET, POST,
HEAD commands)

header
lines

```
GET /somedir/page.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language: fr
```

Carriage return
line feed
indicates end
of message

(extra carriage return, line feed)

- Svar

status line
(protocol
status code
status phrase)

header
lines

```
HTTP/1.0 200 OK
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 .....
Content-Length: 6821
Content-Type: text/html
```

data, e.g.,
requested
html file

data data data data data ...

Typer metoder

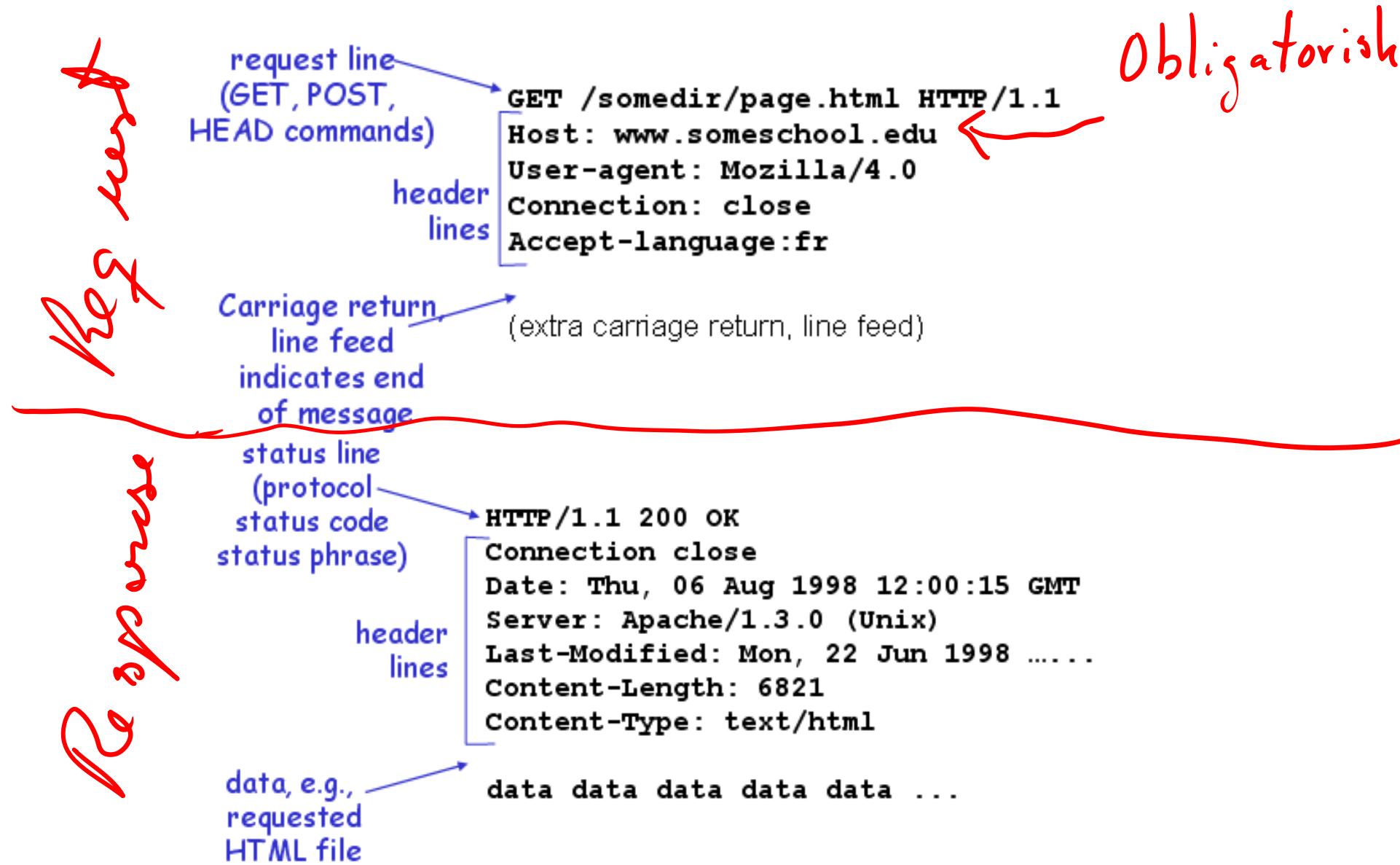
HTTP/1.0

- 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

HTTP 1.1 Meldingsformat



Manuell kjøring



Sett opp TCP-forbindelse
med WWW-server

GET /~blistog/index.html HTTP/1.0

HTTP/1.1 200 OK
 Date: Mon, 13 Oct 2014 11:46:19 GMT
 Server: Apache/2.2.22 (Debian)
 Last-Modified: Tue, 01 Apr 2014 10:16:38 GMT
 ETag: "1ae41f-90c-4f5f876a40d80"
 Accept-Ranges: bytes
 Content-Length: 2316
 Vary: Accept-Encoding
 Cache-Control: no-store, no-cache, must-revalidate, pre-check=0, post-check=0, max-age=0
 Connection: close
 Content-Type: text/html

<?xml version="1.0" encoding="UTF-8"?>
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 xhtml1/DTD/xhtml1-strict.dtd">
 <!-- Dette er HTML-filen jeg benytter som eksempel i TK110 -->
 3.org/1999/xhtml">
 <head>
 <meta http-equiv="Content-type" content="application/xhtml+xml;

GET kommando
sendes serveren

Svar header

Svar data

- Ligger i første linjen på svarmeldingen
- Eksempler:
 - 200 OK**
 - spørring vellykket, objektet kommer senere i meldingen
 - 301 Moved Permanently**
 - etterspurt objekt flyttet, ny adresse senere i meldingen
 - 400 Bad Request**
 - spørring ikke forstått av tjeneren
 - 404 Not Found**
 - etterspurt dokument/fil ikke funnet på denne tjeneren
 - 505 HTTP Version Not Supported**

System:

- Tre siffers statuskode
 - 1xx = Informational**
 - 2xx = Success**
 - 3xx = Redirection**
(alternate URL is supplied)
 - 4xx = Client Error**
 - 5xx = Server Error**

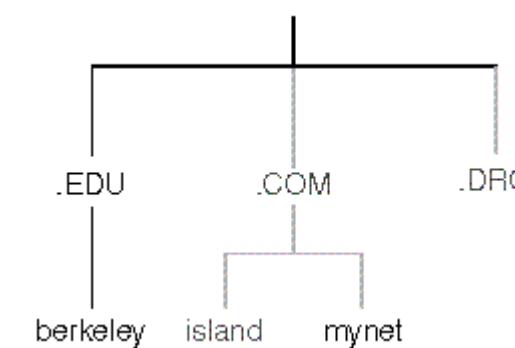
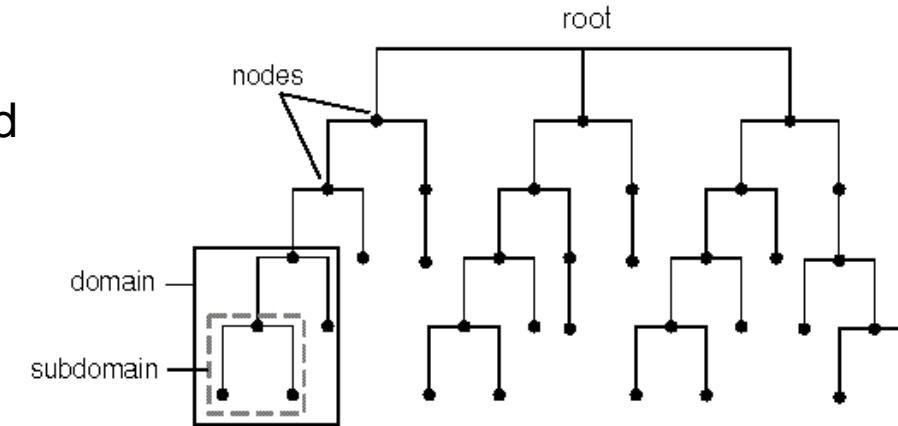
- Mai 2015
- Helt tilbake kompatibel med 1.1
- Hentet mye fra Googles SPDY prosjekt.
- Binær, i stedet for textbasert
- Bedrer ytelse (brukeropplevelse) ved
 - parallelisering
 - komprimering
 - lar serverne proaktivt «dytte» responser inn i klient-cache
- Støttes foreløpig av ca 48% av webserverne (det vi kaller en sakte utrulling...)

Domain Name System

- Mennesker
 - Navn, person-nummer
 - "Ola Nordmann", 161165 42796
- Internett
 - IP-adresse, Navn
 - 10.11.1.21, www.kristiania.no
- Løser dette på Internett med DNS
 - Distribuert database på mange navne-tjenere
 - Protokoll i applikasjons-laget for å knytte navn og IP-adresser

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
- 13 hoved navne-tjenere



- Navne-tjenere cacher DNS kartlegging
- Lagring i cache forsvinner etter en tid (timeout)
- Mekanismer for innmelding og oppdatering utvikles hos **IETF** (Internet Engineering Task Force)
 - Dynamisk oppdatering
 - Sikkerhet
 - Mmm
- Navn (og nummere) forvaltes av IANA med underorganisasjoner
 - Europa: RIPE
 - Norge (.no): NorID

Distribuert database lagrer RR (resource records)

RR format: navn, verdi, type, ttl

- Type=**A**
 - Navn=vertsnavn, verdi=IP-adresse
- Type=**NS**
 - Navn=domene, verdi=IP-adresse til navne-tjener
- Type=**CNAME**
 - Navn=alias, verdi=virkelig navn
- Type=**MX**
 - Navn=alias, verdi=post tjener

Win: nslookup

```
C:\Windows\system32\cmd.exe - nslookup
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\blistog>nslookup
Default Server: obelix.ad.nith.no
Address: 158.36.131.10

> ?
Commands:  (identifiers are shown in uppercase, [] means optional)
NAME      - print info about the host/domain NAME using default server
NAME1 NAME2 - as above, but use NAME2 as server
help or ?  - print info on common commands
set OPTION - set an option
  all        - print options, current server and host
  [no]debug  - print debugging information
  [no]d2     - print exhaustive debugging information
  [no]defname - append domain name to each query
  [no]recurse - ask for recursive answer to query
  [no]search  - use domain search list
  [no]vc     - always use a virtual circuit
domain=NAME - set default domain name to NAME
srchlist=N1[/N2/.../N6] - set domain to N1 and search list to N1,N2, etc.
root=NAME   - set root server to NAME
retry=X     - set number of retries to X
timeout=X   - set initial time-out interval to X seconds
type=X      - set query type (ex. A,AAAA,A+AAAA,ANY,CNAME,MX,NS,PTR,SOA,SRV)
querytype=X - same as type
class=X     - set query class (ex. IN (Internet), ANY)
[no]msxfr   - use MS fast zone transfer
  ixfrver=X - current version to use in IXFR transfer request
server NAME - set default server to NAME, using current default server
lserver NAME - set default server to NAME, using initial server
root        - set current default server to the root
ls [opt] DOMAIN [> FILE] - list addresses in DOMAIN (optional: output to FILE)
  -a        - list canonical names and aliases
  -d        - list all records
  -t TYPE   - list records of the given RFC record type (ex. A,CNAME,MX,NS,PTR etc.)
view FILE   - sort an 'ls' output file and view it with pg
exit        - exit the program
```

nslookup
kan også
brukes i
OSXog Linux,
men der
foretrekkes

dig

```
Command Prompt
C:\Users\Bengt>nslookup www.vg.no
Server: dnscache01.get.no
Address: 84.208.20.110

Non-authoritative answer:
Name: www.vg.no
Addresses: 2001:67c:21e0::16
          195.88.54.16
          195.88.55.16

C:\Users\Bengt>.
```

```
C:\>ipconfig /all

Windows IP-konfigurasjon

    Vertsnavn . . . . . : NITH-blistog
    Primær DNS-suffiks . . . . . : 
    Nodetype . . . . . : Hybrid
    IP-ruting aktivert . . . . . : Nei
    WINS Proxy aktivert . . . . . : Nei
    Sokeliste for DNS-suffiks . . . . . : oslo.nith.no

Ethernet-kort eth0:
    Tilkoblingsspesifikt DNS-suffiks : oslo.nith.no
    Beskrivelse . . . . . : Broadcom NetXtreme 57xx Gigabit
    Fysisk adresse . . . . . : 00-13-72-94-FF-78
    DHCP aktivert . . . . . : Ja
    Automatisk konfigurasjon aktivert: Ja
    IP-adresse . . . . . : 10.21.5.228
    Nettverksmaske . . . . . : 255.255.252.0
    Standard gateway . . . . . : 10.21.4.1
    DHCP-server . . . . . : 10.21.4.131
    DNS-servere . . . . . : 10.21.4.131
                                10.21.21.101
    Primær WINS-server . . . . . : 10.21.4.131
    Leasingavtale mottatt . . . . . : 26. oktober 2008 17:25:30
    Leasingavtale utgår . . . . . : 26. oktober 2008 19:25:30
```

- Viser IP-nettverksparametre som de har blitt satt på (NT) arbeidsstasjon
 - For å endre nettverksparametere må du bruke `netsh`
 - Linux/OS X: `ifconfig`
 - ”kraftigere” da du også kan endre oppsett med samme

netstat -an

```
C:\>netstat -an
```

Aktive tilkoblinger

Prot.	Lokal adresse	Ekstern adresse	Tilstand
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1028	0.0.0.0:0	LISTENING
TCP	0.0.0.0:3389	0.0.0.0:0	LISTENING
TCP	10.21.5.228:139	0.0.0.0:0	LISTENING
TCP	10.21.5.228:1418	64.233.183.18:443	ESTABLISHED
TCP	10.21.5.228:1445	158.36.191.141:443	TIME_WAIT
TCP	10.21.5.228:1457	64.233.183.18:443	ESTABLISHED
TCP	127.0.0.1:1035	0.0.0.0:0	LISTENING
TCP	127.0.0.1:1416	127.0.0.1:1417	ESTABLISHED
TCP	127.0.0.1:1417	127.0.0.1:1416	ESTABLISHED
TCP	127.0.0.1:1455	127.0.0.1:1456	ESTABLISHED
TCP	127.0.0.1:1456	127.0.0.1:1455	ESTABLISHED
TCP	127.0.0.1:5152	0.0.0.0:0	LISTENING
TCP	127.0.0.1:5354	0.0.0.0:0	LISTENING
UDP	0.0.0.0:161	***	
UDP	0.0.0.0:445	***	
UDP	0.0.0.0:500	***	
UDP	0.0.0.0:1025	***	
UDP	0.0.0.0:1029	***	
UDP	0.0.0.0:4500	***	
UDP	10.21.5.228:123	***	
UDP	10.21.5.228:137	***	
UDP	10.21.5.228:138	***	
UDP	10.21.5.228:5353	***	
UDP	127.0.0.1:123	***	
UDP	127.0.0.1:1030	***	
UDP	127.0.0.1:1040	***	

- Viser transportlag-tilstand ("åpne porter")

netstat -r

```
C:\>netstat -r

Rutingstabell
=====
Grensesnittliste
0x1 ..... MS TCP Loopback interface
0x2 ...00 13 72 94 ff 78 .... Broadcom NetXtreme 57xx Gigabit Controller -
iport for pakkeplanlegger
=====
=====
Aktive ruter:
Nettverksmål Nettverksmaske Gateway Grensesnitt Metrikk
      0.0.0.0      0.0.0.0  10.21.4.1  10.21.5.228  20
      10.21.4.0  255.255.252.0  10.21.5.228  10.21.5.228  20
      10.21.5.228  255.255.255.255  127.0.0.1  127.0.0.1  20
      10.255.255.255  255.255.255.255  10.21.5.228  10.21.5.228  20
      127.0.0.0      255.0.0.0  127.0.0.1  127.0.0.1  1
      169.254.0.0  255.255.0.0  10.21.5.228  10.21.5.228  30
      224.0.0.0      240.0.0.0  10.21.5.228  10.21.5.228  20
      255.255.255.255  255.255.255.255  10.21.5.228  10.21.5.228  1
Std. gateway: 10.21.4.1
=====
Faste ruter:
Ingen
```

- Viser lokal routing-tabell
- route-kommandoen lar deg også endre den...

Avslutning

- Definere Internett
- TCP/IP-modellen
 - Hvilke oppgaver løses på de ulike lagene
- Klient/Tjener-modellen
- HTTP
 - Oppbygningen av URI og URL
 - Request/response
 - GET og ulike typer responskoder
 - Kort om tilstandsløshet og Cookies
- DNS
 - Hvordan organisert: TLD og NS-servere
- Verktøy (!)

OBLIGATORISK innlevering /arbeidskrav

- Denne oppgaven MÅ leveres innen fristen!
- Krav for å gå opp til eksamen

Skriv et kort essay (2-4 sider) om ARPA Net

- Hvem laget det, og hvorfor
- Hvilken påvirkning har dette hatt på den moderne verdenen

Innleveringsfrist kommer på Canvas nærmere eksamen – skriv det i dag så slipper du å stresse med det når fristen blir lagt ut...

Dagens øving

- Wireshark i praksis – HTTP og DNS protokoll
 - Sist installerte dere Wireshark
 - Start Wireshark
 - Åpne en browser og gå til www.h-ck.me
 - Dette er en enkel webside som over HTTPS sender en webside med innhold «test»
 - Åpne en browser og gå til www.ikke.no
 - Dette er en enkel webside som over HTTP sender en enkel HTML side
 - Bruk PING på kommandolinje for å finne IP adresse for disse to domenene og sett filter i Wireshark på IP adresse så all «støy» blir borte og dere kun ser disse to tilgangene
 - Inspiser HTTP trafikken til www.ikke.no
 - Hva skjer med trafikken til www.h-ck.me, hvorfor er denne annerledes
- Lær dere verktøyene; ping, traceroute/tracert, netstat, ifconfig/ipconfig, dig/nslookup
- TK1104_Øvingsoppgaver_Leksjon_0x06_WWW.pdf

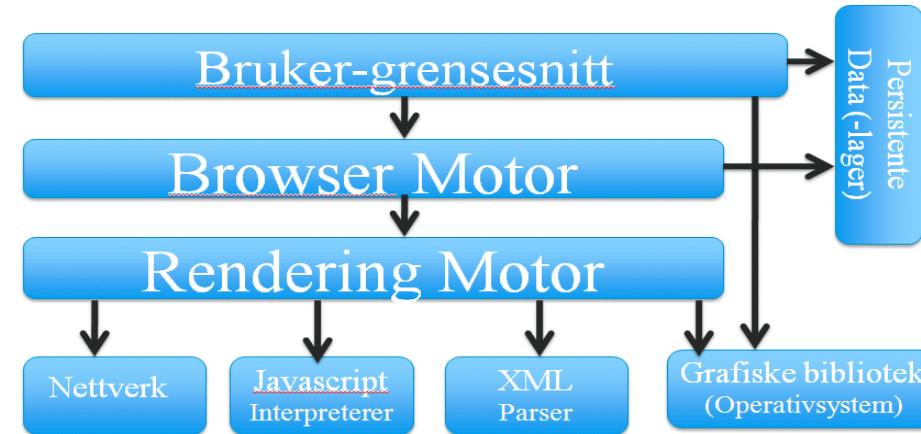
For valgfritt egenstudie

For de som ønsker å lære noen emner mer i dybden for å forstå det bedre er det her samlet noen ekstra temaer relatert til dagens undervisning, det må forventes en del egenarbeid for å forstå disse emnene.

Det vil ikke komme spørsmål på eksamen fra disse, og dette er altså ikke ansett for å være en del av pensum.

Arkitektur

- Bruker-grensesnitt (UI)
 - Toolbar, nedlastingsinfo, knapper, printing,..
- Browser motor
 - Høynivå grensesnitt mot Rendering motor
 - Laster URI, støtter "surfing": frem, tilbake, reload,..
- Rendering motor
 - Beregner side-layout og viser frem.
 - **Parser** html, css, ...
- Nettverk
 - Står for filoverføring (http, ftp, ..)
 - Oversetter mellom char-set, kan MIME
 - **Cache**

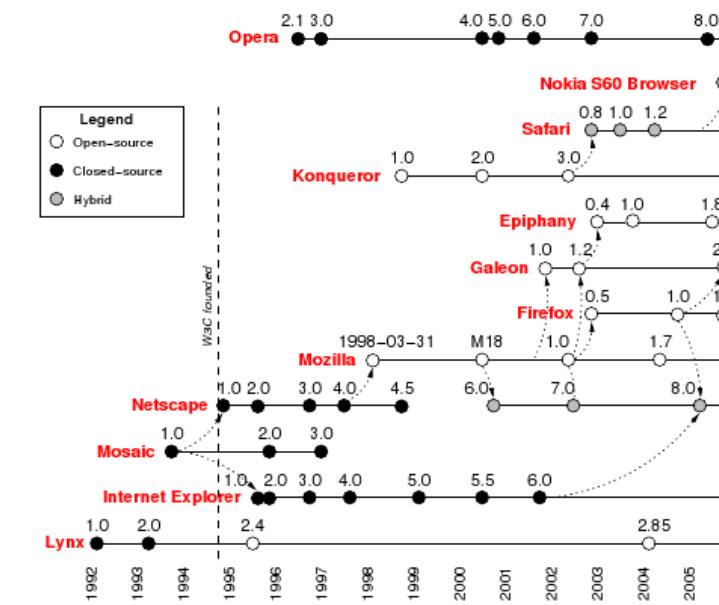
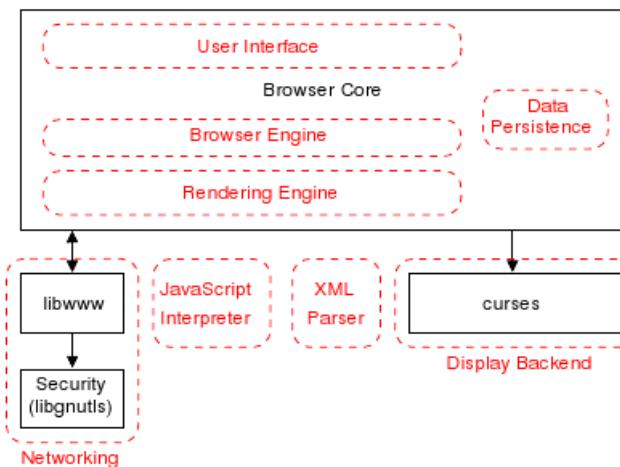
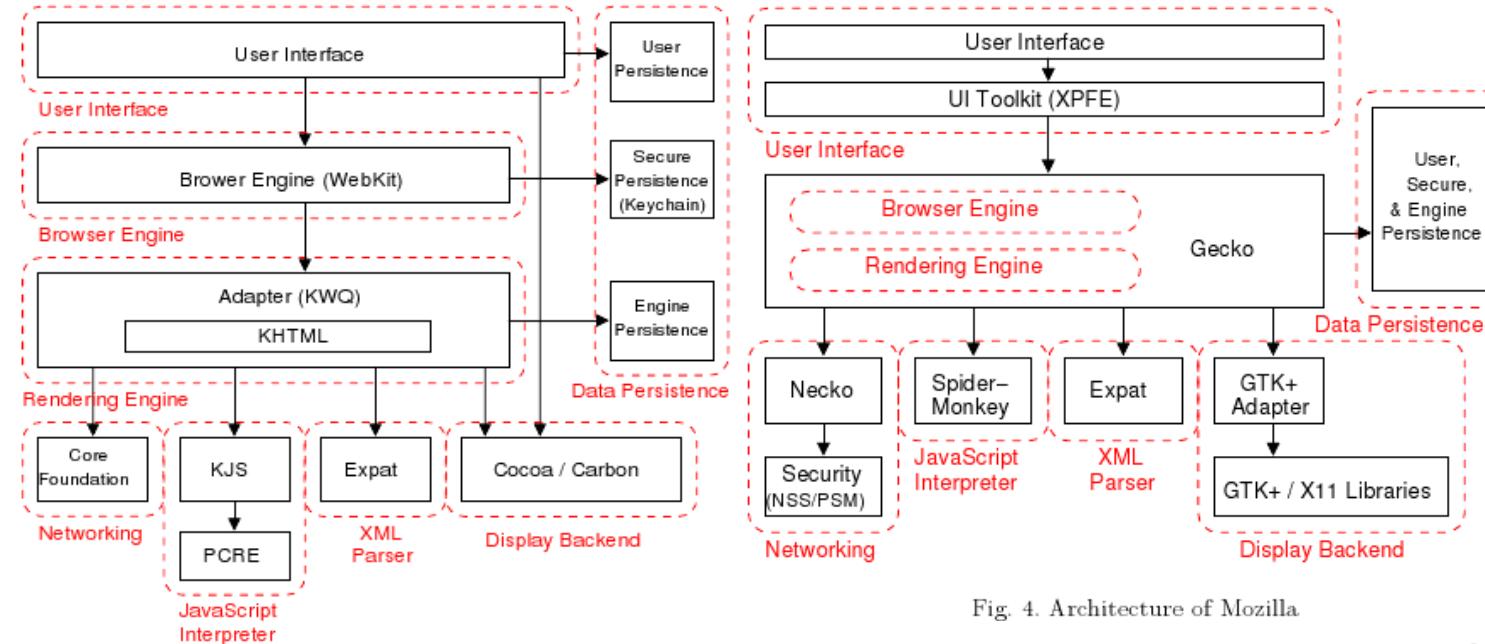


- **Javascript Interpreterer**
 - Kjører Javascript
- **XML Parser**
 - Parser html/xml til et **DOM**-tre
- **Grafikkbibliotek**
 - Tegne-, meny- og vindus-rutinene
 - Fonter
- **Persistente data**
 - Lagre bokmerker, sertifikater, personlig konfigurering

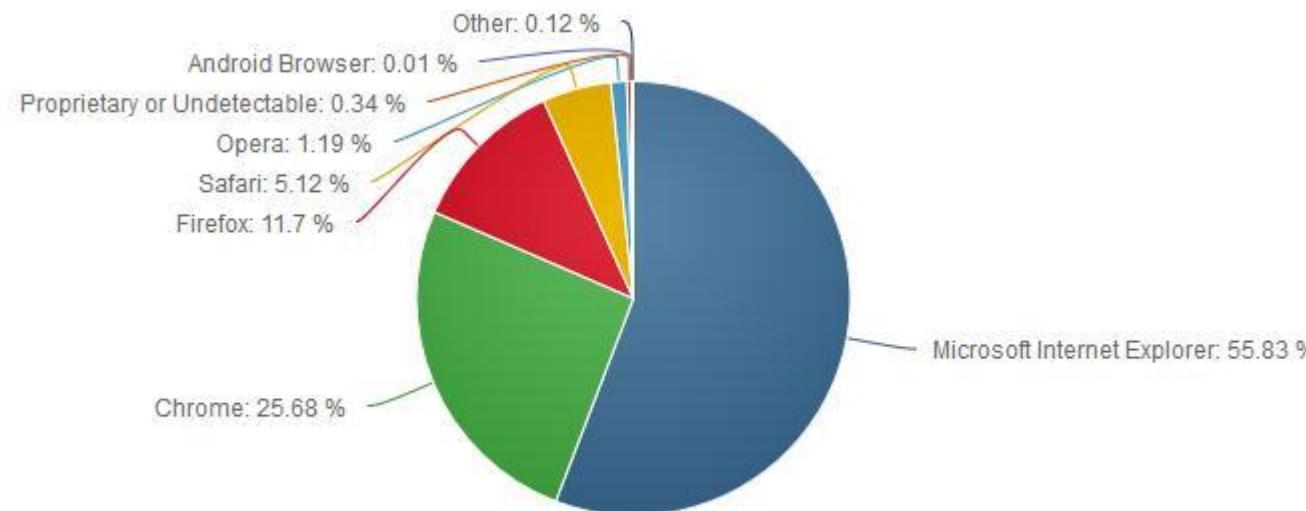
Layout motor = browser + rendering

- Rendering og browser motoren kombineres ofte i en Layout motor.
- Opera
 - Presto
 - men de gikk i 2013 over til samme som...
- Safari + Chrome
 - **WebKit**
- Firefox (Mozilla)
 - **Gecko**
- Lynx
 - Libwww + Curses

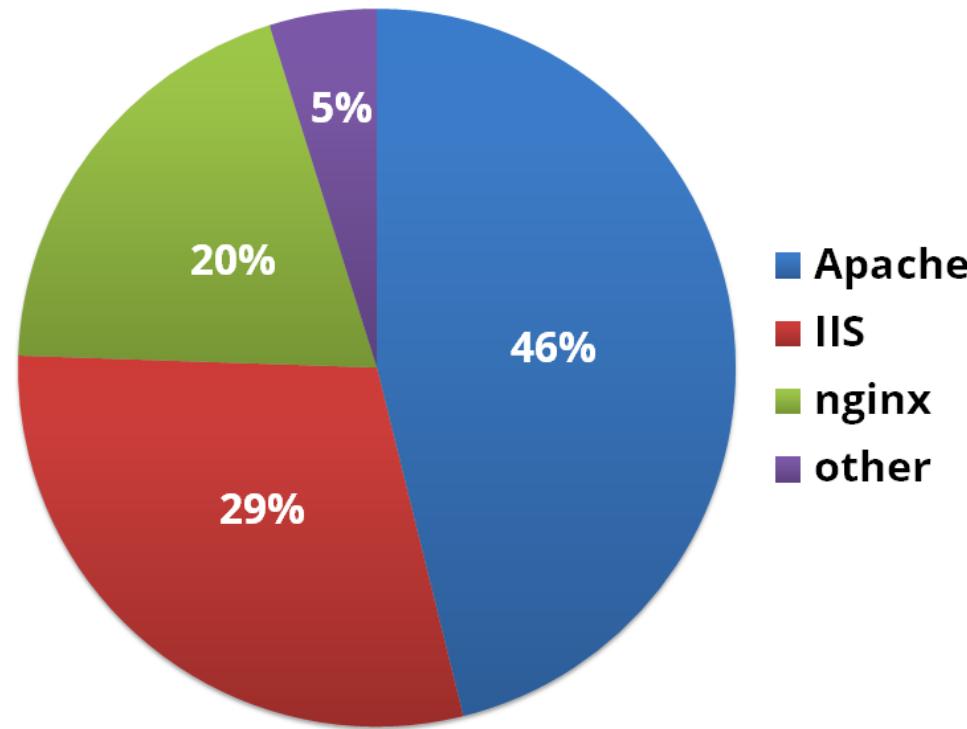
Arkitekturen (eldre design)



- Utformer og sender http-forespørsler
- Viser frem ("renderer") web-innhold ("layout-engine")
- Kjører script og plug-ins/add-ons...

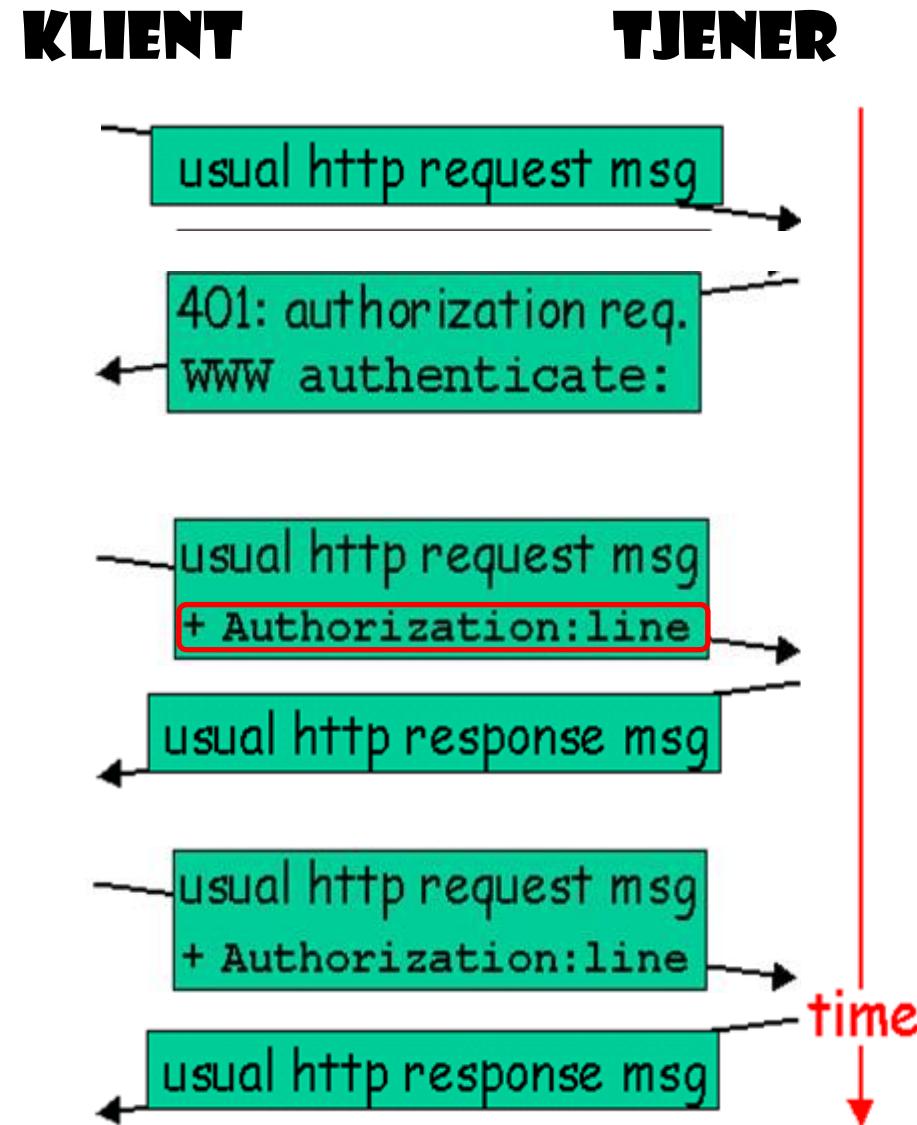


- Tar mot og betjener **http-forespørsler**

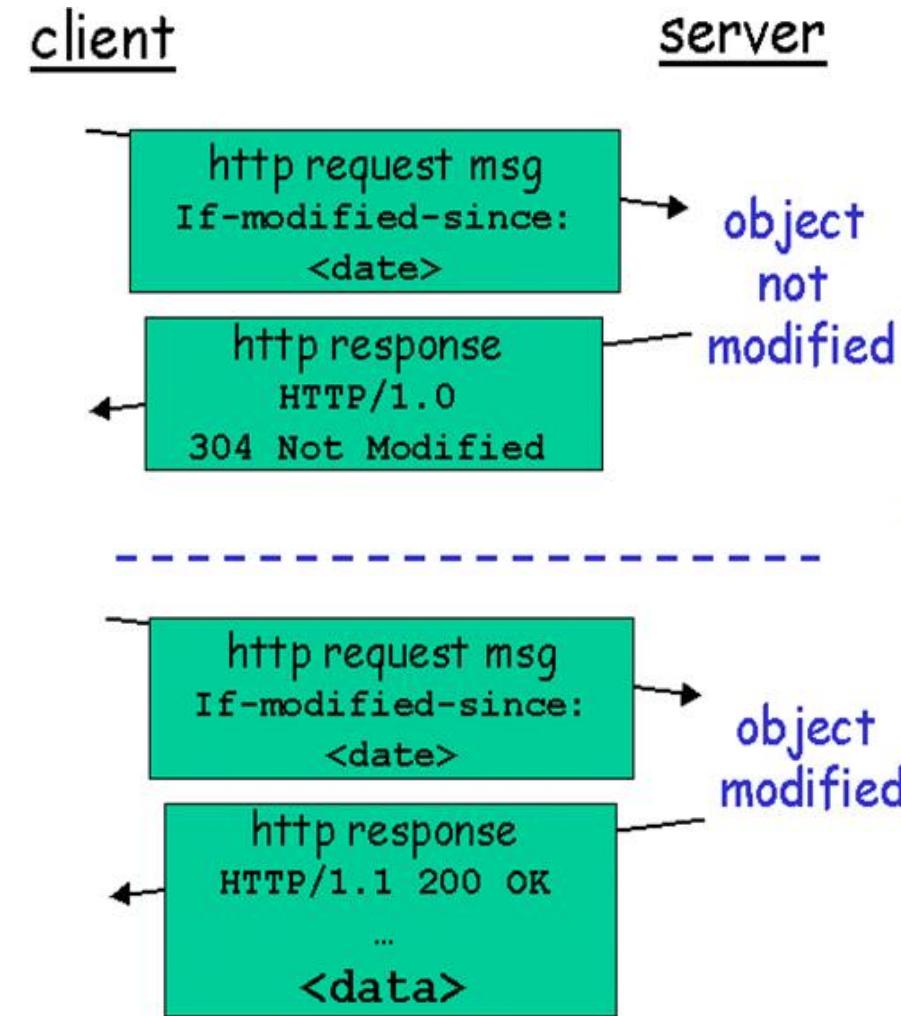


Brukertjener kommunikasjon (Ex)

- Ex: Bekrefteelse av identitet
 - Navn, passord
 - Ligger i header ved spørring
 - Responsen ligger i svar-header
 - Usikker (**Base 64** koding, ikke kryptering)
- Vanligvis (stateless) må dette gjentas ved hver spørring, men **browseren lagrer navn, passord** for brukeren slik at det bare ses en gang



- Betinget GET
 - **If-modified-since:**
 - Ikke send svar hvis klient har oppdatert versjon
 - Sjekker tidsstempel på filen
- Klient
 - Spesifiser dato for **cachet** fil
- Tjener
 - Statuskode **304** dersom ikke oppdatert
 - Header-felter som Etag og Vary har Proxy-bruk å gjøre...



- HTTP er en **tilstandsløs** protokoll
 - Fra serveren og klientens perspektiv er alle forespørsler fullstendig uavhengige av hverandre
- Mange Web-steder benytter **cookies**
- En cookie har 3/4 hoved-elementer
 - Cookie **header** linje i http-responsen
 - Cookie **header** linje i http-forespørselen
 - (Cookie fil som ligger hos klienten)
 - Database over cookies hos tjeneren
- Cookie kan
 - Bevare tilstand
 - "Huske" autorisasjoner og settinger

- Tilleggslinjer i header for **MIME** innhold
 - Tekst, bilde, audio, video, applikasjon («binære filer»)
 - Opprinnelig for epost, benyttes også av HTTP-klient/tjener
 - Angir koder for innholdstype og hvordan det er (om-kodet) under overføringen

MIME version

method used to encode data

multimedia data type, subtype, parameter declaration

encoded data

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.....
.....base64 encoded data
```

Denne forelesningsøkten vil bli tatt opp og lagt ut i emnet i etterkant.

Hvis du ikke vil være med på opptaket:

 ^ Start Video	La være å delta med webkameraet ditt.
 ^ Unmute	La være å delta med mikrofonen din.
To: Marianne Sundby (Privately) Type message here...	Still spørsmål i Chat i stedet for som lyd. Hvis du ønsker kan spørsmålet også sendes privat til foreleser.



Høyskolen
Kristiania

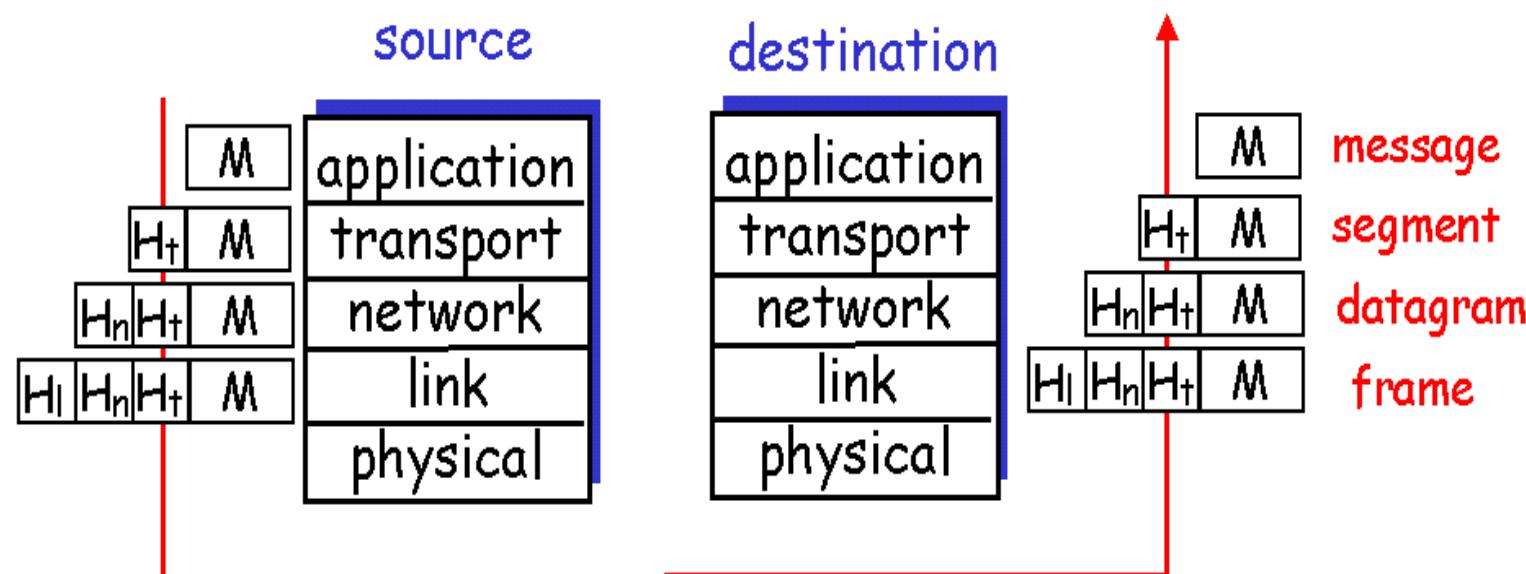
TK1100

Applikasjonslaget
7. forelesning

Sist

- Generelt
 - Internett er infrastruktur
- Linje- vs Pakke-svitsjing
 - Forsinkelser
- Lagdelt

Protokoll stack:



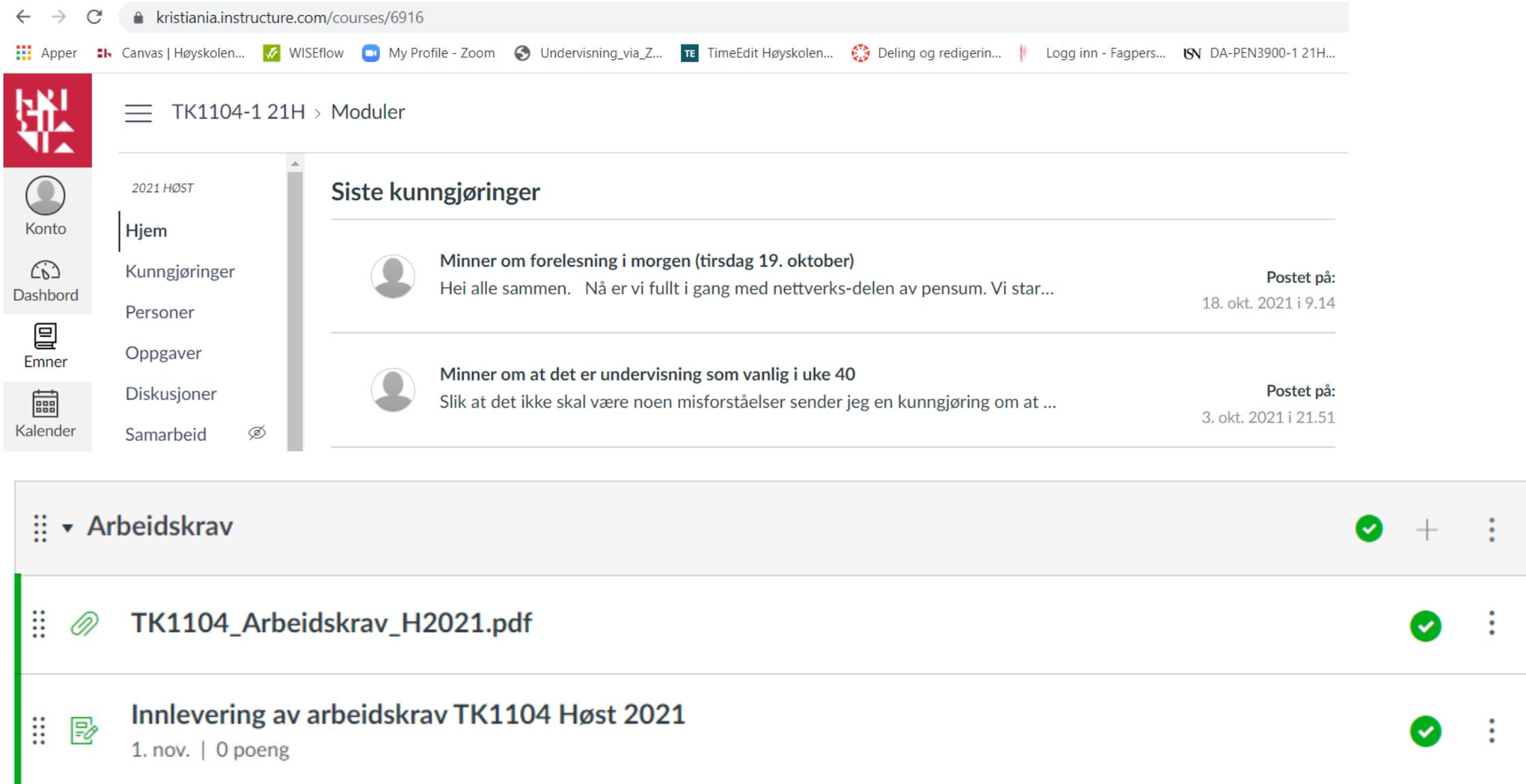
Sist (2)

- 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

Arbeidskrav – frist 1. november

← → C kristiania.instructure.com/courses/6916

Apper Canvas | Høyskolen... WISEflow My Profile - Zoom Undervisning_via_Z... TimeEdit Høyskolen... Deling og redigerin... Logg inn - Fagpers... DA-PEN3900-1 21H...



The screenshot shows the Kristiania LMS interface. The top navigation bar includes links for Apper, Canvas, WISEflow, My Profile - Zoom, Undervisning_via_Z..., TimeEdit, Deling og redigerin..., Logg inn - Fagpers..., and DA-PEN3900-1 21H... The main content area is titled 'TK1104-1 21H > Moduler'. On the left, a sidebar menu is open, showing '2021 HØST' and the following items: Hjem, Kunngjøringer, Personer, Oppgaver, Diskusjoner, Samarbeid, and a 'Kalender' item which is currently selected. The main content area is titled 'Siste kunngjøringer' and lists two posts:

- Minner om forelesning i morgen (tirsdag 19. oktober)**
Hei alle sammen. Nå er vi fullt i gang med nettverks-delen av pensum. Vi star...
Postet på: 18. okt. 2021 i 9.14
- Minner om at det er undervisning som vanlig i uke 40**
Slik at det ikke skal være noen misforståelser sender jeg en kunngjøring om at ...
Postet på: 3. okt. 2021 i 21.51

Below this, there is a section titled 'Arbeidskrav' with two items listed:

- TK1104_Arbeidskrav_H2021.pdf**
- Innlevering av arbeidskrav TK1104 Høst 2021**
1. nov. | 0 poeng

Om eksamen

Om eksamen

Det er 24 timers frist på denne hjemmeeksamen, men forventet arbeidsmengde er 4-6 timer så det er ikke meningen å «jobbe gjennom natten». Vær obs på at eksamen MÅ leveres innen fristen som er satt, og må leveres via eksamensplattformen WISEFLOW. Det vil ikke være mulig å få levert oppgaven etter fristen – det betyr at du bør levere i god tid slik at du kan ta kontakt med eksamenskontoret eller brukerstøtte hvis du har tekniske problemer.

Da dette er en hjemmeeksamen er det viktig å vise helhetlig forståelse, og oppgavene har et større preg av drøfting. Det forventes derfor utfyllende og forklarende svar på alle oppgaver. Figurer og skisser kan du velge å tegne i tekstbehandleren, eller ved å tegne på papir og laste opp bilde – husk å sette inn bilde på riktig sted i besvarelsen. (Bilder som er vedlegg, men ikke satt inn i besvarelsen anses ikke som en del av besvarelsen.)

Om eksamen

Det presiseres at studenten skal besvare eksamen selvstendig og individuelt, samarbeid mellom studenter og plagiat er ikke tillatt.

I regneoppgaver er det vesentlig at du legger vekt på å vise hvordan du kommer frem til svaret. Svar på regneoppgaver uten å vise fremgangsmåte er å betrakte som ubesvarte oppgaver.

OBS: Besvarelsen skal ikke være på mer enn 15 A4 sider, med font størrelse 12, normale marger og linjeavstand 1.0.

Om eksamen

Oppgave 1: Teori spørsmål (15%)

Oppgave 2: Tall og binære data (35%)

Oppgave 3: Praktiske oppgaver (30%)

*De praktiske oppgavene vil teste din forståelse av **verktøy** og av protokoller, de vil ikke være lik de vi har hatt i øvingene, men øvingsoppgavene vil være veldig relevante!*

Oppgave 4: “Tyngre” teori og forståelse (20%)

Om eksamen

Eksamen er Bestått / Ikke bestått.

Det kreves 40% riktig på eksamen for å bestå.

OBS: På en deloppgave som gir maksimalt 5 poeng foretas sensur slik:

- 1 poeng: Studenten har svart feil, men «inne på noe»
- 2 poeng: Veldig kort svar som kun delvis svarer på oppgaven
- 3 poeng: Et kort og riktig svar, men oppfattes som «tynt»
- 4 poeng: Et godt svar, men ikke utfyllende eller komplett
- 5 poeng: Fullgodt svar på oppgaven

Svarer du KUN på Oppgave 1 og 2 (totalt 50%), må alle deloppgavene ha «et godt svar» for å ha forhåpninger om å bestå. Moral: SVAR PÅ ALT ☺

Navn på laget	Betegnelse på overføringsenhet	Viktigste oppgaver/funksjoner Exempel på protokoller/standarder
Applikasjonslaget	Melding (Message)	Støtte nettverksapplikasjoner 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 Ex: TCP, UDP,..
Nettverkslaget	Datagram	Routing av datagram fra/til vertsmaskin gjennom nettverkskjernen Ex: IP (v4 og v6), ICMP, RIP, OSPF, BGP,...
Datalinjelaget	Ramme (Frame)	(Pålitelig) Levering av ramme fra nabo-node til nabo-node Ex: Ethernet II, FDDI, IEEE 802.11 ...
Fysisk	Bit	(Kode og) Flytte enkeltbit mellom kommunikasjonspartnere. Ex: 10BaseT, ...

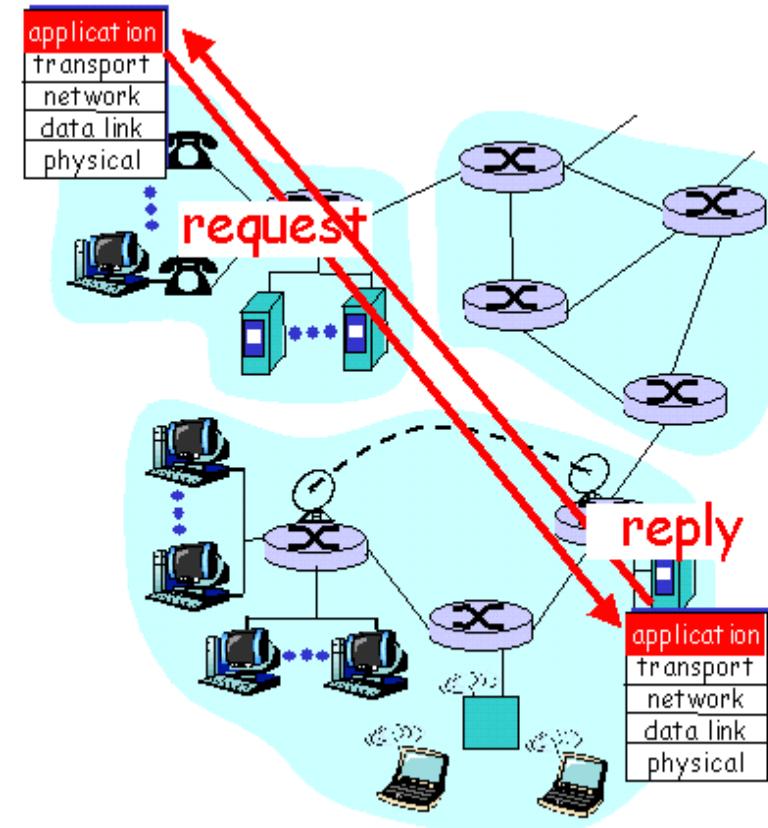
Applikasjonslaget

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)

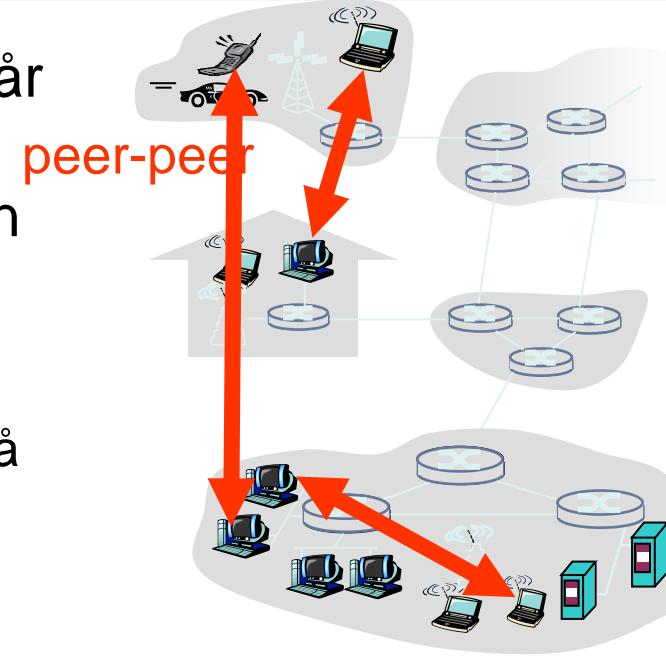
Klient/tjener

- 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 opplasting. Problematiske 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, Teams, og andre «chat tjenester»
 - Sentral tjener for å slå opp adresse til mottager – registrer seg selv der
 - der du logger på og (automatisk) oppgir kontaktinfo (IP-adresse mm)
 - kontakter sentral-register for å få adresse til samtalepartner.
 - Samtale er P2P
- Dataspill uten dedikerte servere
 - Sentral tjener for å slå opp adresse til mottager – ofte en del av et rammeverk som Steam
 - Spillet foregår P2P, eller at en av spillerne er en midlertidig «tjener»

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 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 nesten forelesning!

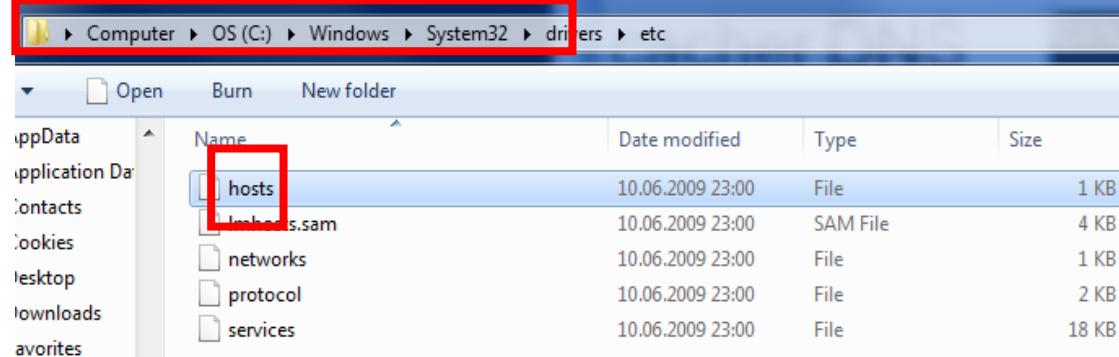
Domain Name System

DNS (Domain Name System)

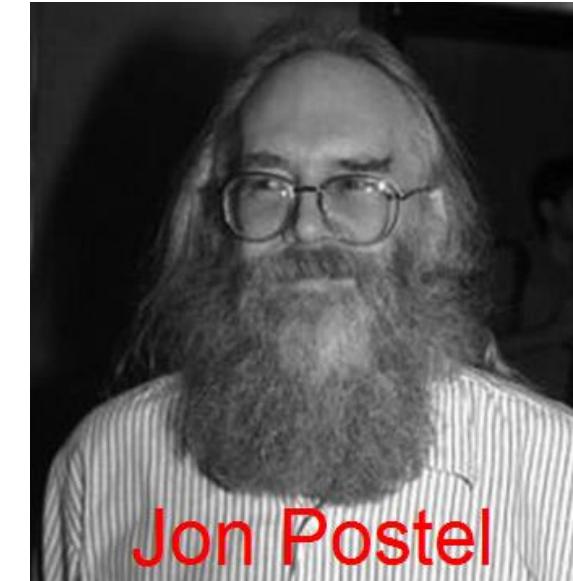
- Mennesker
 - Navn, **person-nummer**
 - "Ola Nordmann", **161165 42796**
- Internett
 - **IP-adresse**, Navn
 - **54.221.226.116**, www.kristiania.no
- Løser dette på Internett med DNS
 - Distribuert «database» på mange navne-tjenere
 - Protokoll i applikasjons-laget for å knytte navn og IP-adresser

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

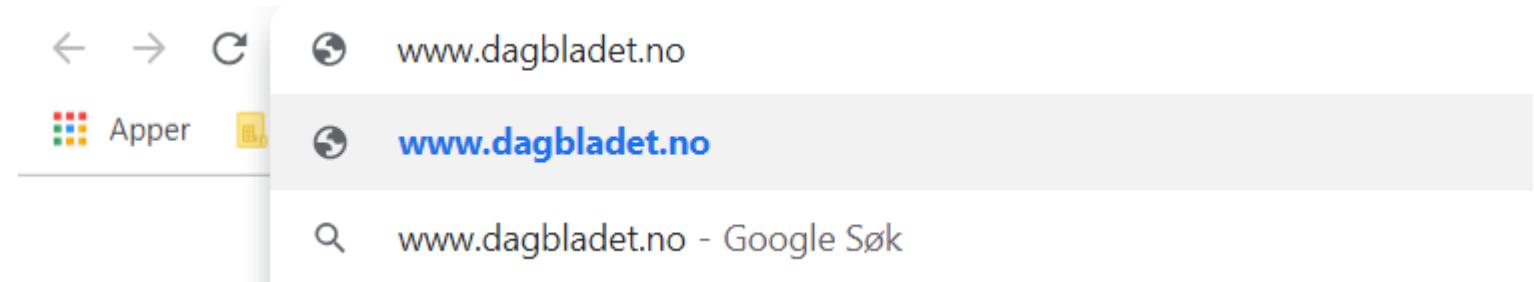


Demo av hosts

195.88.54.16

www.dagbladet.no

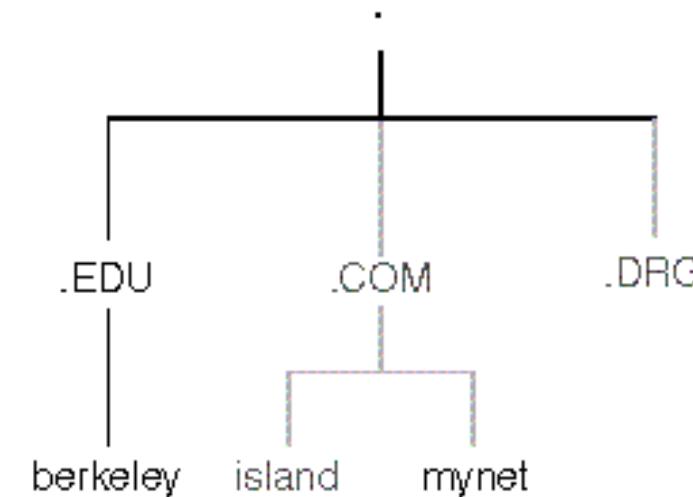
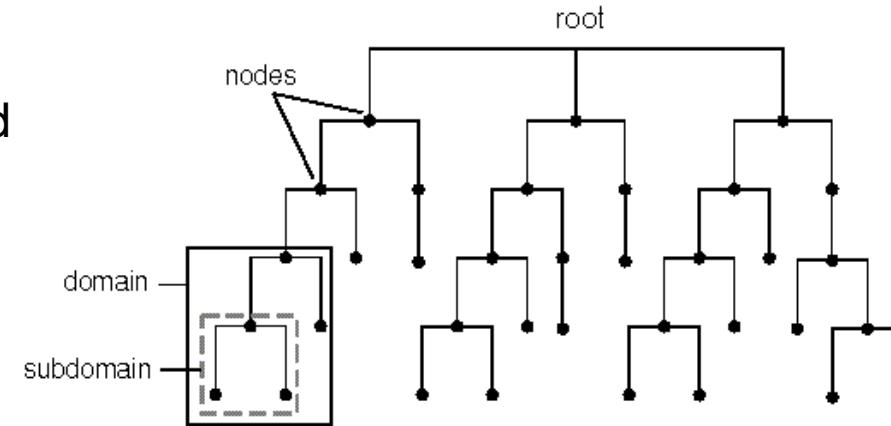
Tenk gjennom hvordan
dette kan
(mis-) brukes!



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

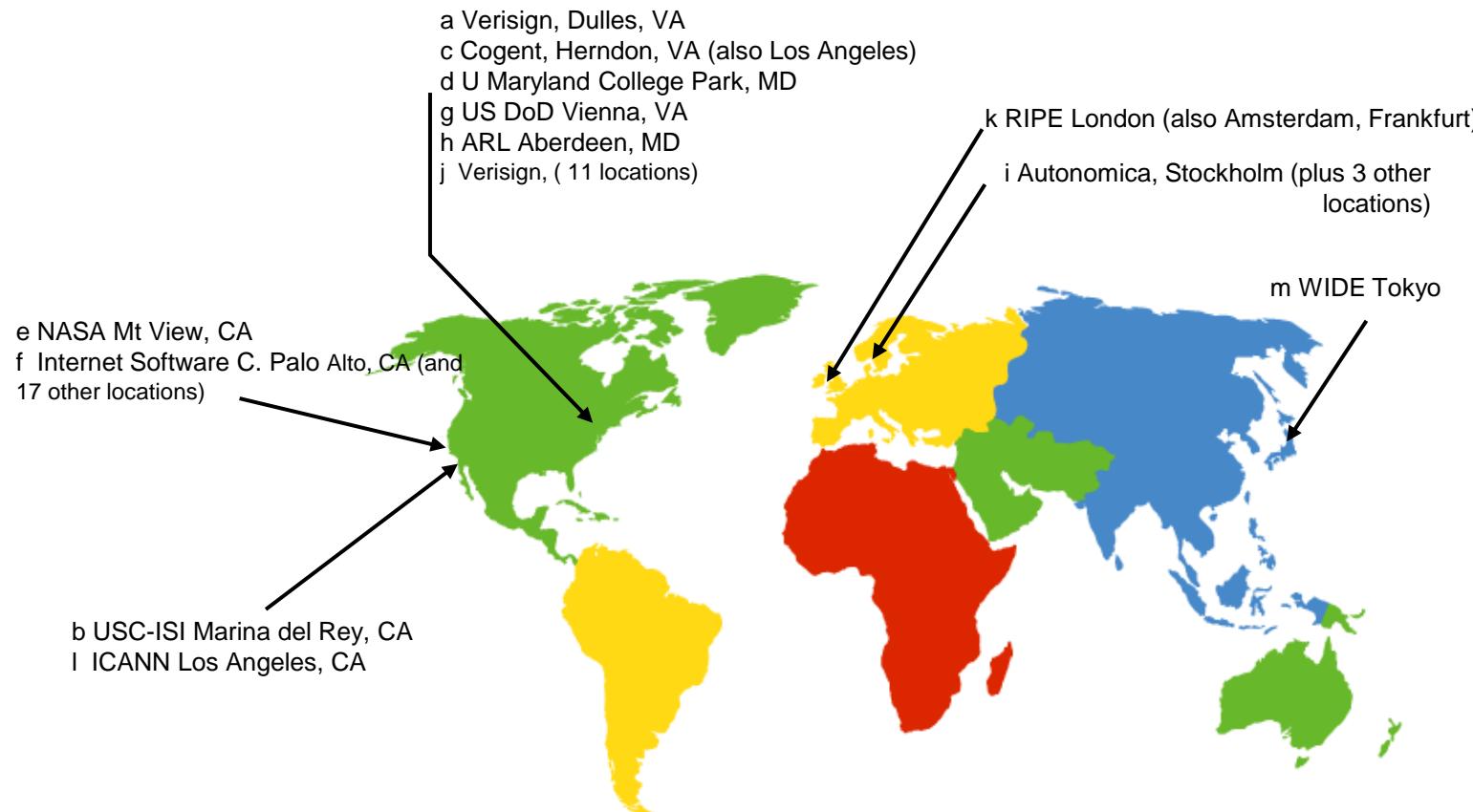
DNS navne-tjenere

- 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 = 2001:503:231c
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

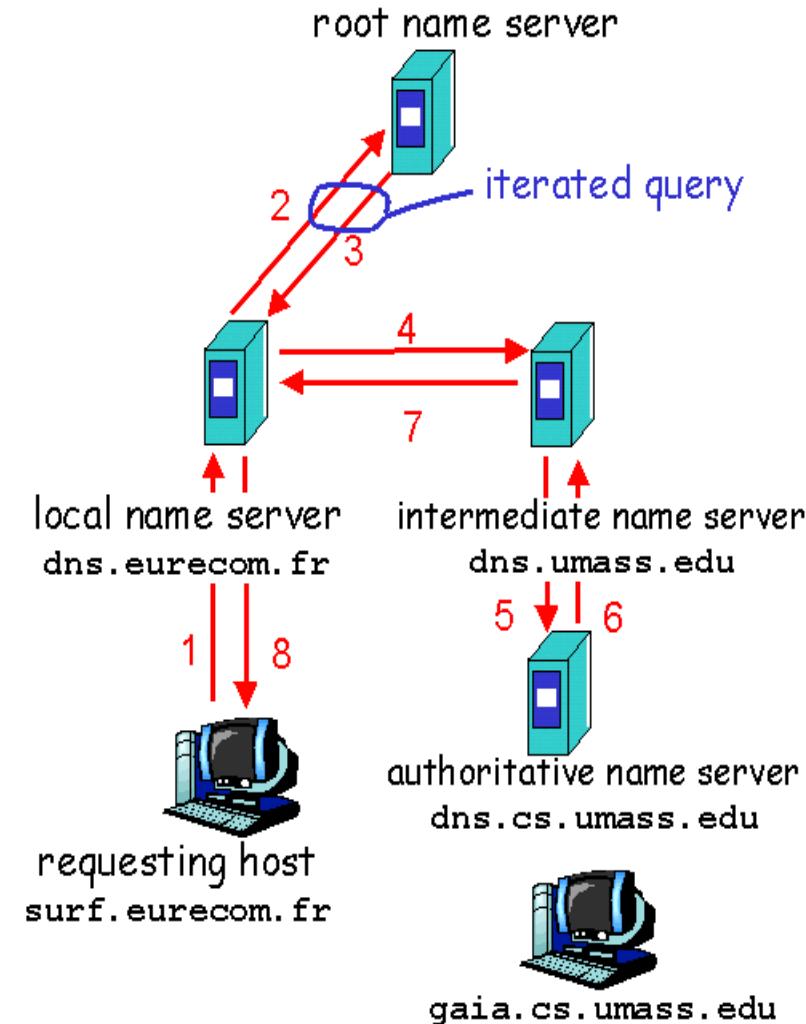
Non-authoritative answer:
no nameserver = z.nic.no
no nameserver = i.nic.no
no nameserver = njet.norid.no
no nameserver = x.nic.no
no nameserver = y.nic.no
no nameserver = not.norid.no

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

Gjentatte spørninger

- 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

Norske bokstaver?

- For å støtte andre bokstav-sett enn ASCII benytter DNS nå *Punycode* (Internationalized Domain Names – IDN), 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..
- Jeg eier for eksempel www.østby.net = www.xn--stby-fra.net



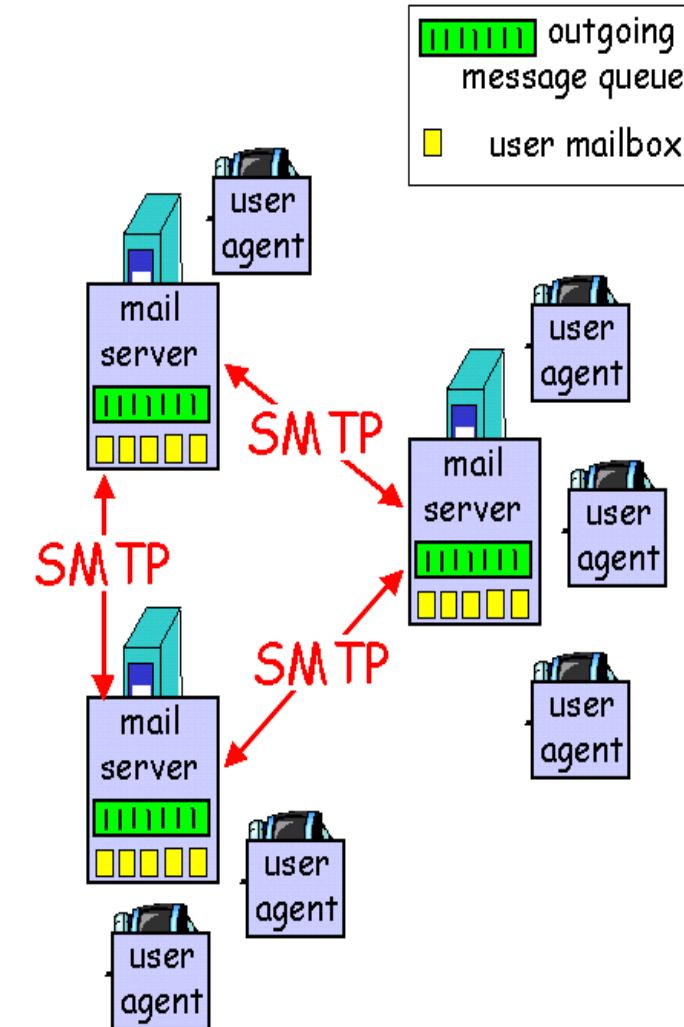
post

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.



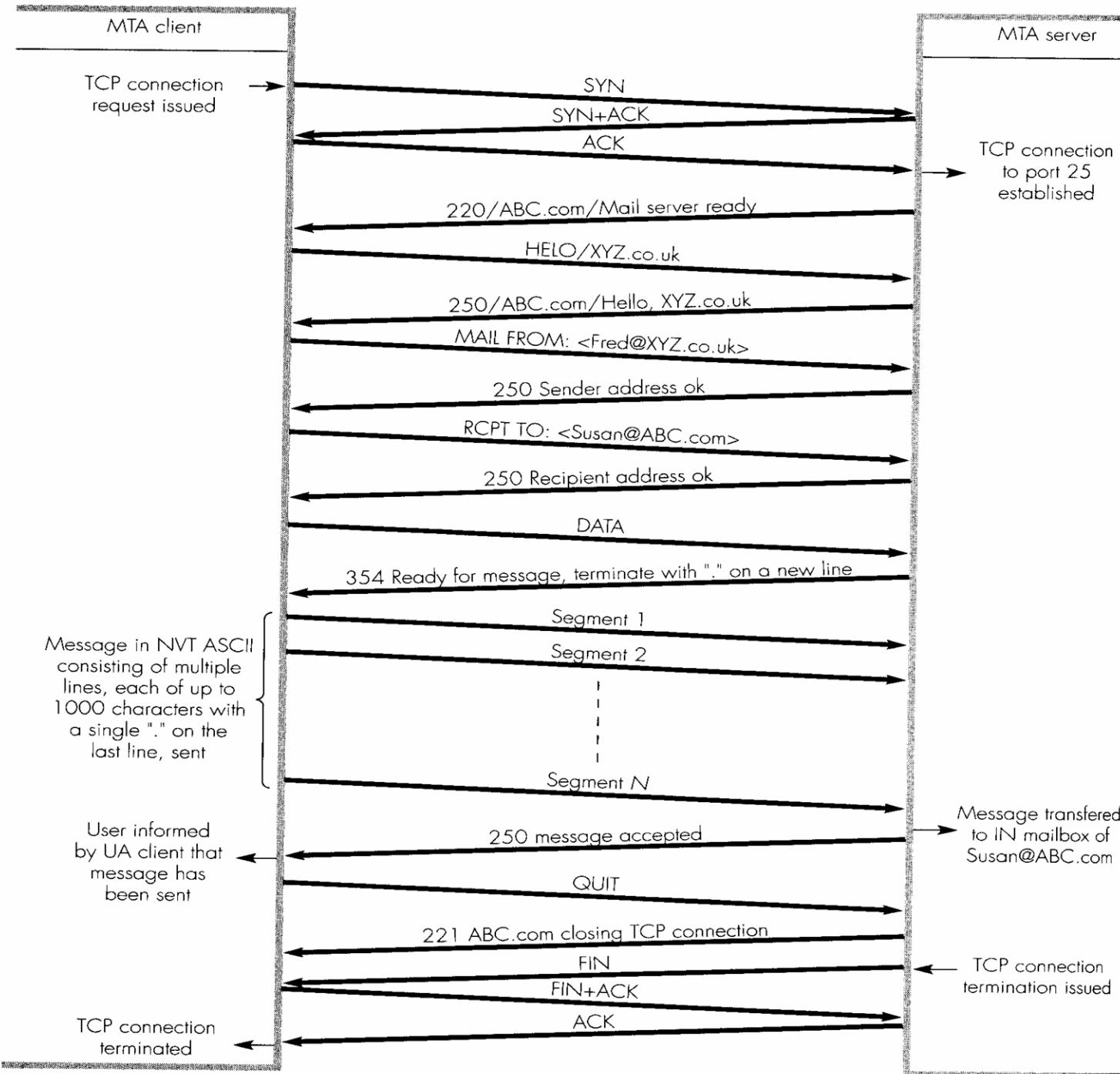
Post tjener

- 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

Sende epost



Eksempel (2010) - PuTTY

login.nith.no - PuTTY

```
220 login.nith.no ESMTP Exim 4.69 Tue, 08 Feb 2011 13:13:24 +0100
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
```

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address) Port

Connection type:

Raw Telnet Rlogin SSH Serial

Priority Inbox (2)

- [Inbox \(12\)](#)
- [Sent Mail](#)
- [Drafts](#)
- [Bin](#)
- [NKI-stipend](#)
- [Notes](#)
- [58 more▼](#)
- [Contacts](#)
- [Tasks](#)

Litt testing [Inbox](#) | [X](#)

heisan@hoppsan.com to someone

Dette er en liten test..

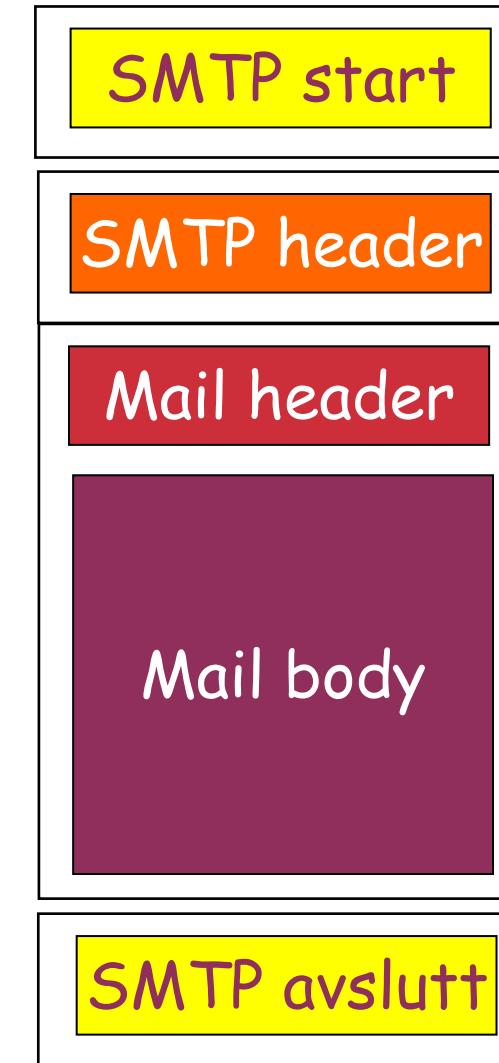
[Reply](#) [Reply to all](#) [Forward](#)

SMTP kontra HTTP

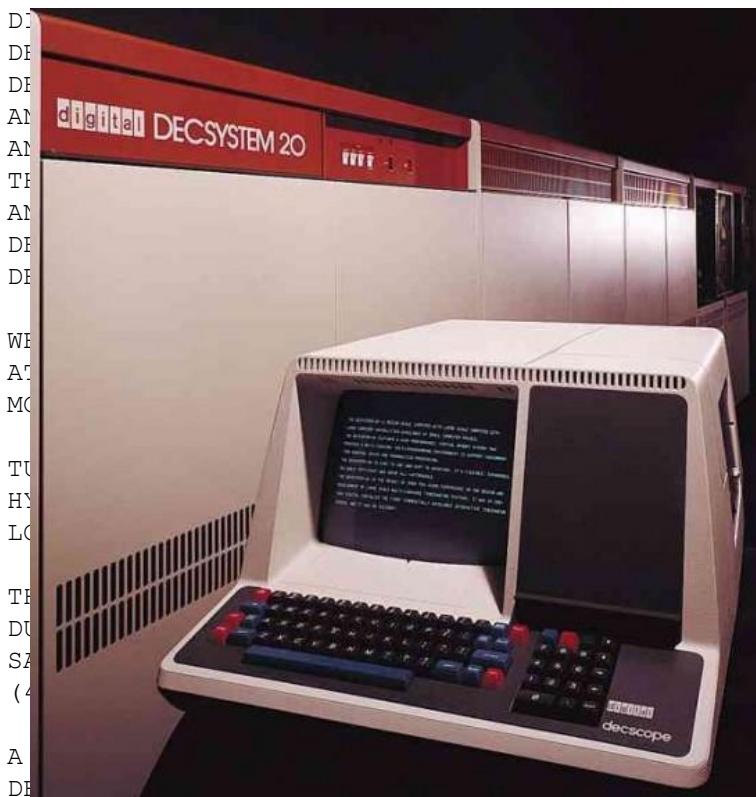
- **SMTP** bruker vedholdende forbindelse
- Noen karakterstrenger er ulovlige i meldinger
- Alt går i ASCII kode
 - Bl.a derfor omkodes meldingen (base64,Uuencode, hex64 ...)
 - CRLF
 - CRLF
avslutter en melding
- **HTTP** henter data (pull), epost skyver data (push)
- **HTTP** overfører (vanligvis) 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



3. Mai 1978:



PLEASE FEEL FREE TO CONTACT THE NEAREST DEC OFFICE
FOR MORE INFORMATION ABOUT THE EXCITING DECSYSTEM-20 FAMILY

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

THE DECSYSTEM-20 FAMILY G IN CALIFORNIA THIS

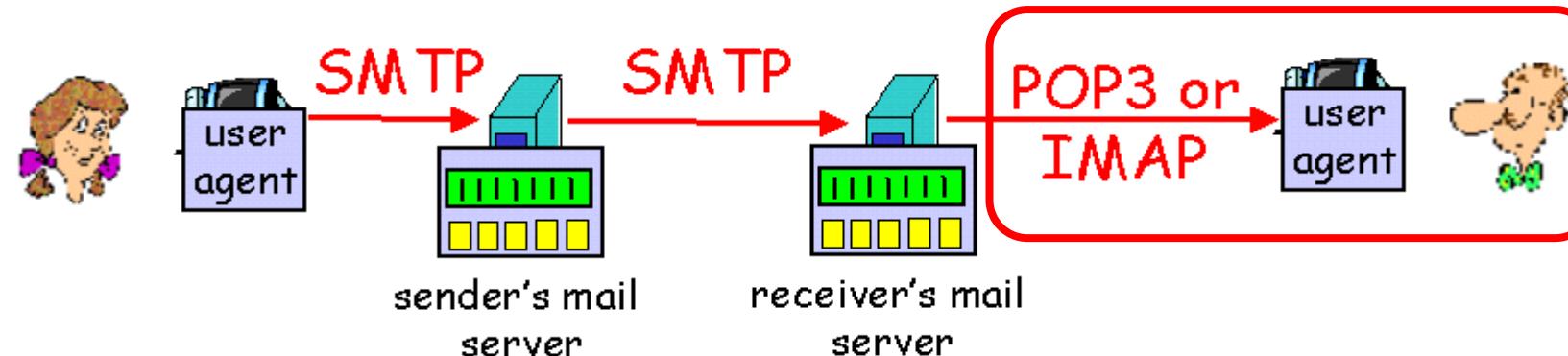
1 AND RT 92)

ALS ON-LINE TO OTHER
ARE UNABLE TO ATTEND.



Gary Thuerk

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
```

Eksempel (POP3)

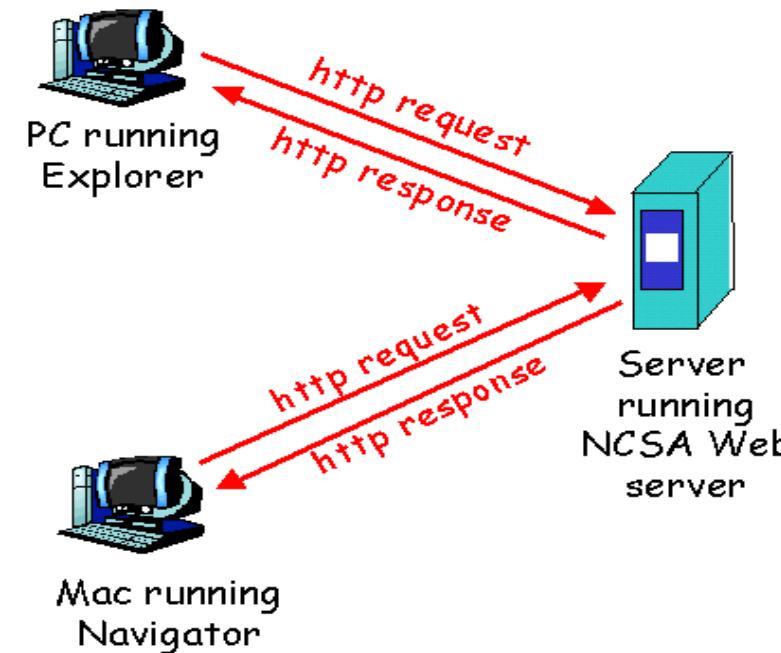
```
S: +OK mail.oslo.dph.no POP3 server (Netscape Messaging Server - Version 3.6)
    ready Wed, 5 Feb 2006 15:42:30 +0100
C: user blistog
S: +OK Password required for blistog
C: pass passord
S: +OK blistog's maildrop has 206 messages (17625186 octets)
C: retr 206
S: +OK 366 octets
S: Status: U
S: Return-Path: <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>

Hyper**t**ext transfer p**ro**to**c**ol

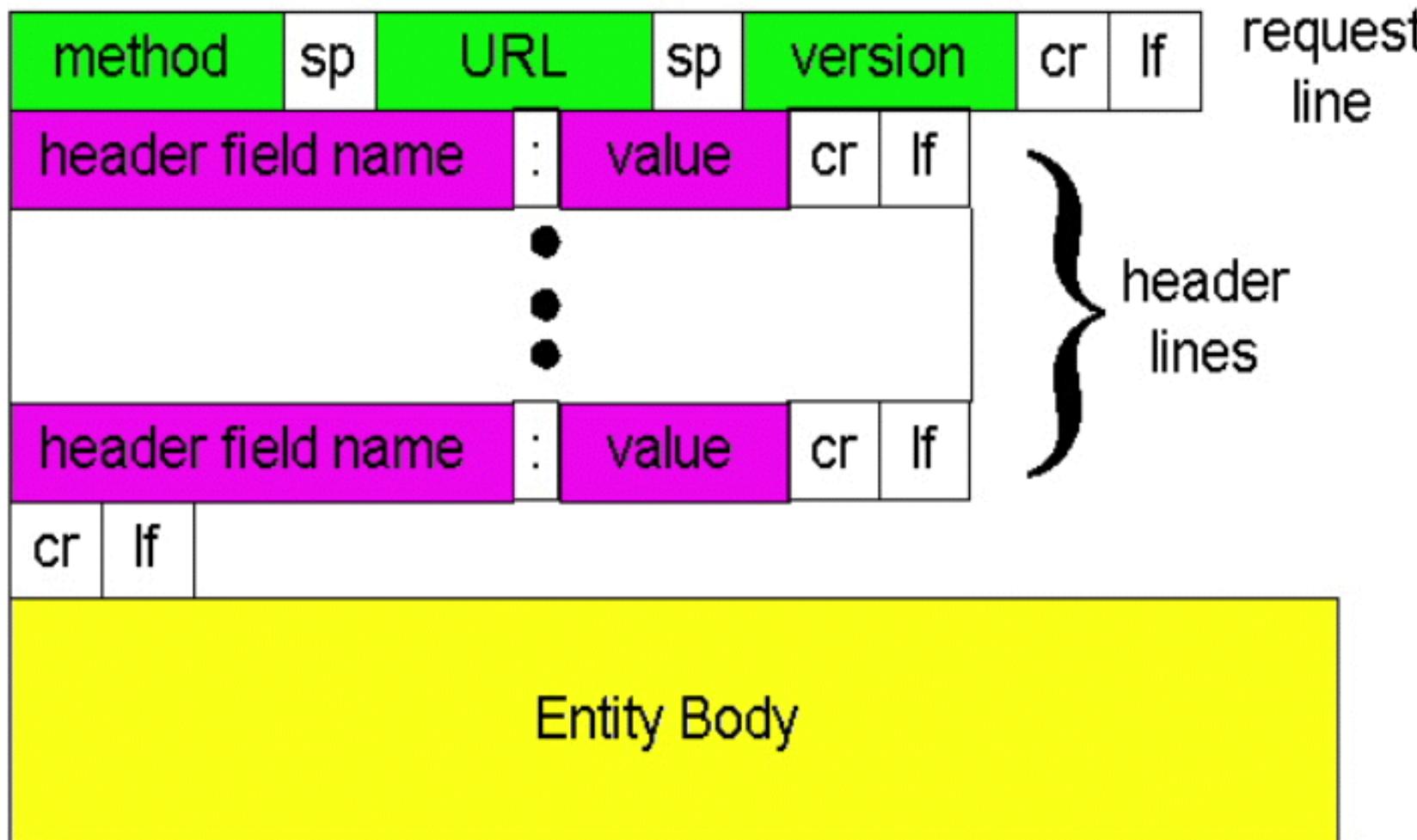
HTTP (HyperText Transfer Protocol)

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

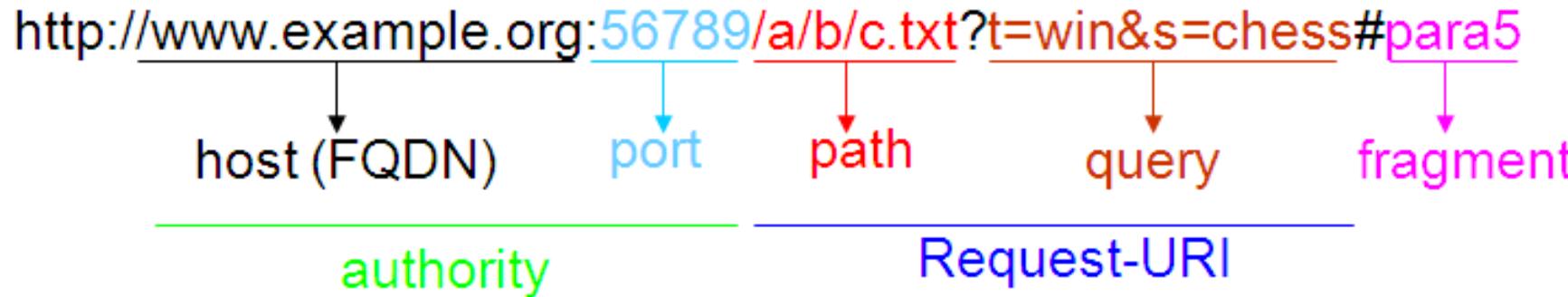


HTTP meldingsformat: spørring

- 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")

Typer metoder

HTTP/1.0

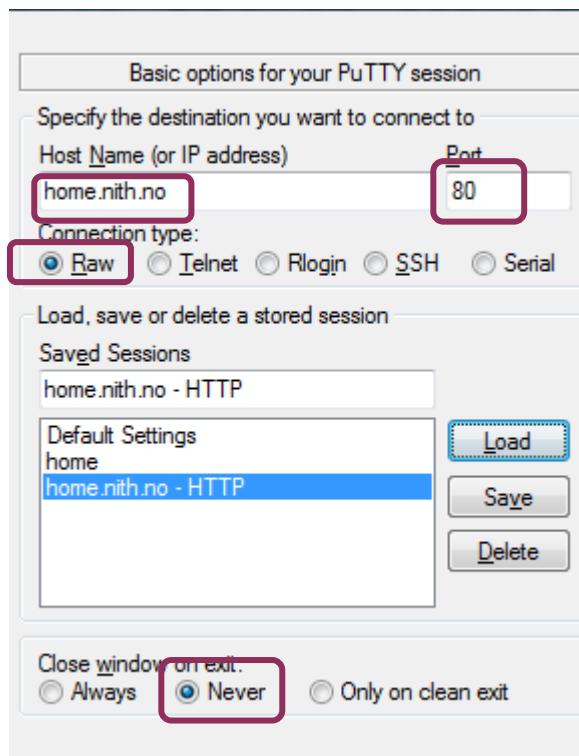
- GET
- POST
- HEAD
 - Spør bare server om metainformasjon = headere

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)

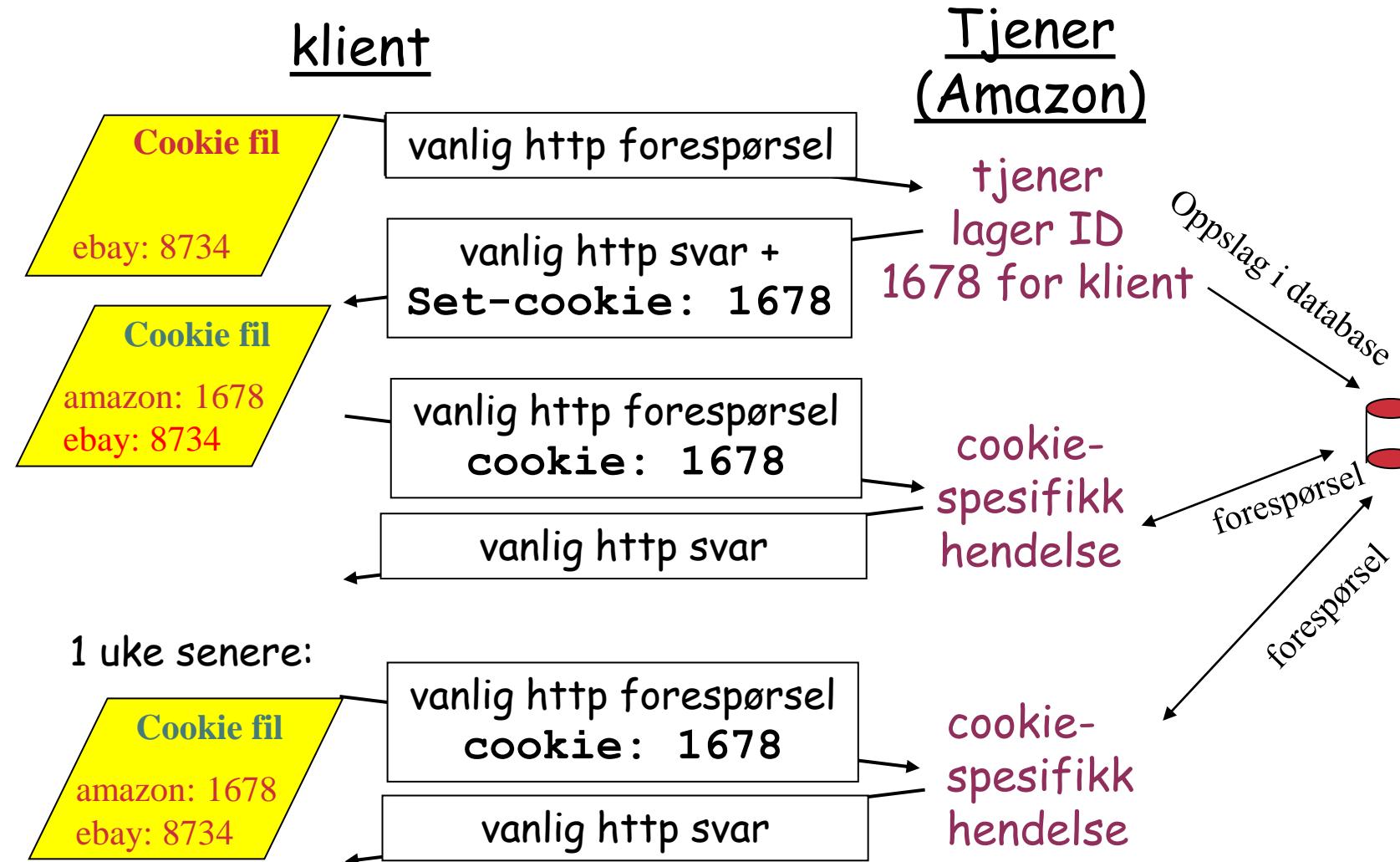


```
GET /~blistog/index.html HTTP/1.1
Host: home.nith.no Spørring
HTTP/1.1 200 OK
Date: Mon, 31 Jan 2011 15:42:28 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Wed, 24 Nov 2010 07:29:12 GMT
ETag: "70043a-923-495c772174200"
Accept-Ranges: bytes
Content-Length: 2339
Cache-Control: no-store, no-cache, must-revalidate,
ax-age=0
Connection: close
Content-Type: text/html; charset=UTF-8
<?xml version="1.0" encoding="UTF-8"?>
SVAR
```

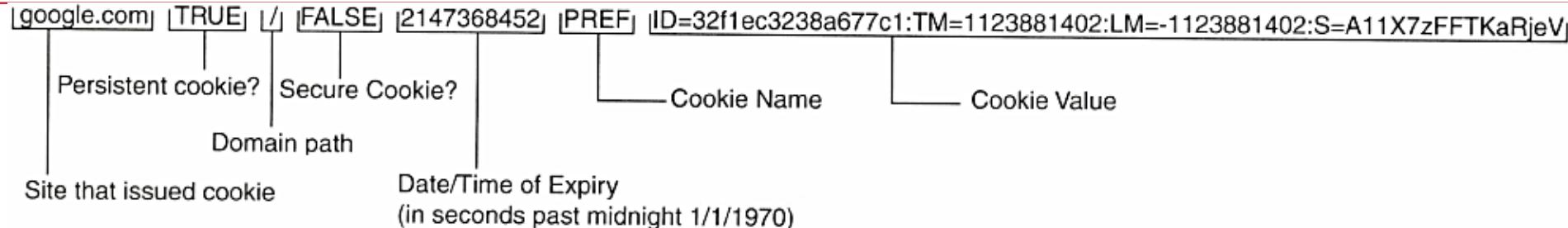
Beholde tilstanden med cookie

- 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)

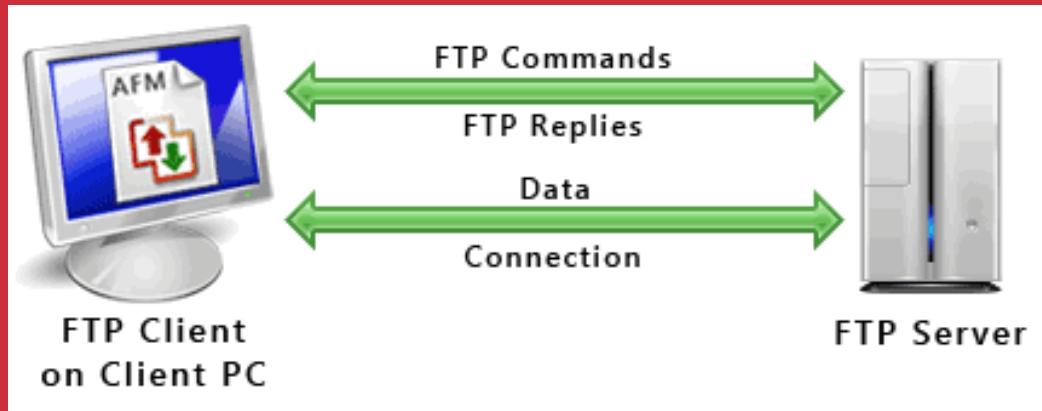


Cookies (2)



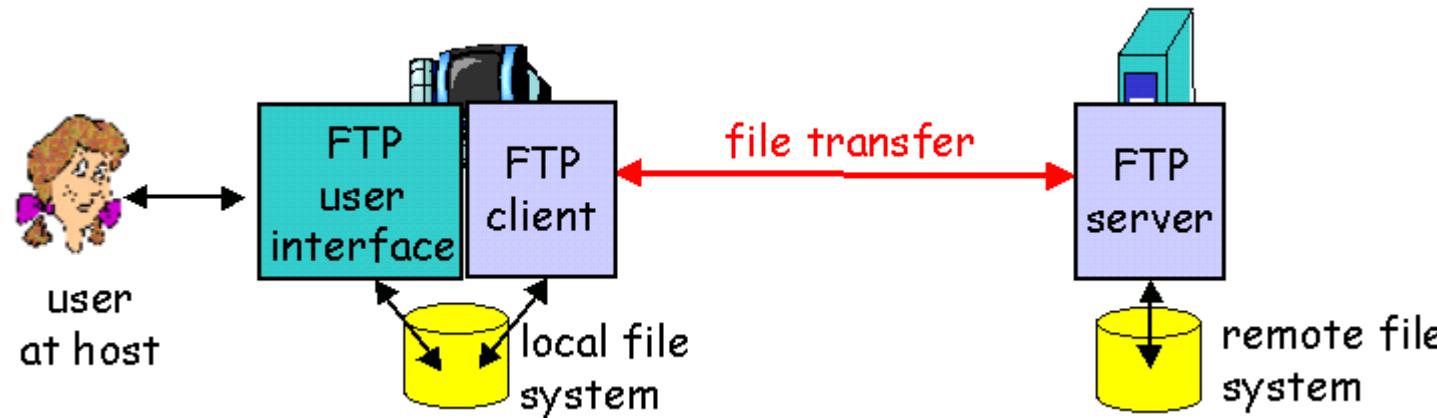
- Cookies kan være **persistente** eller **ikke-persistente**
 - Persistente lagres på klient-maskin frem til utløpsdatoen
 - Ikke-persistente brukes kun i den opprettede sesjonen og slettes når browser avsluttes.
- Cookies kan være **sikre** eller **usikre**
 - Sikre cookies sendes kun over **HTTPS (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)

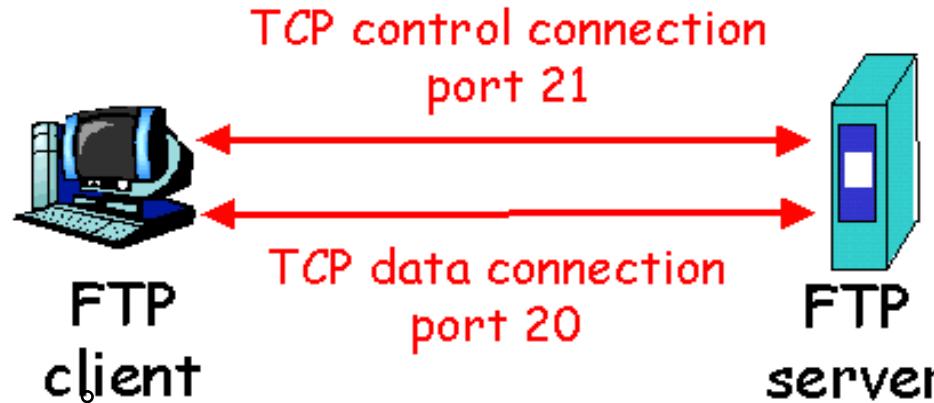


- Overføring av filer
- Bruker klient/tjener modell
- Rask overføring
- *Kan* bruke browser som grensesnitt

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



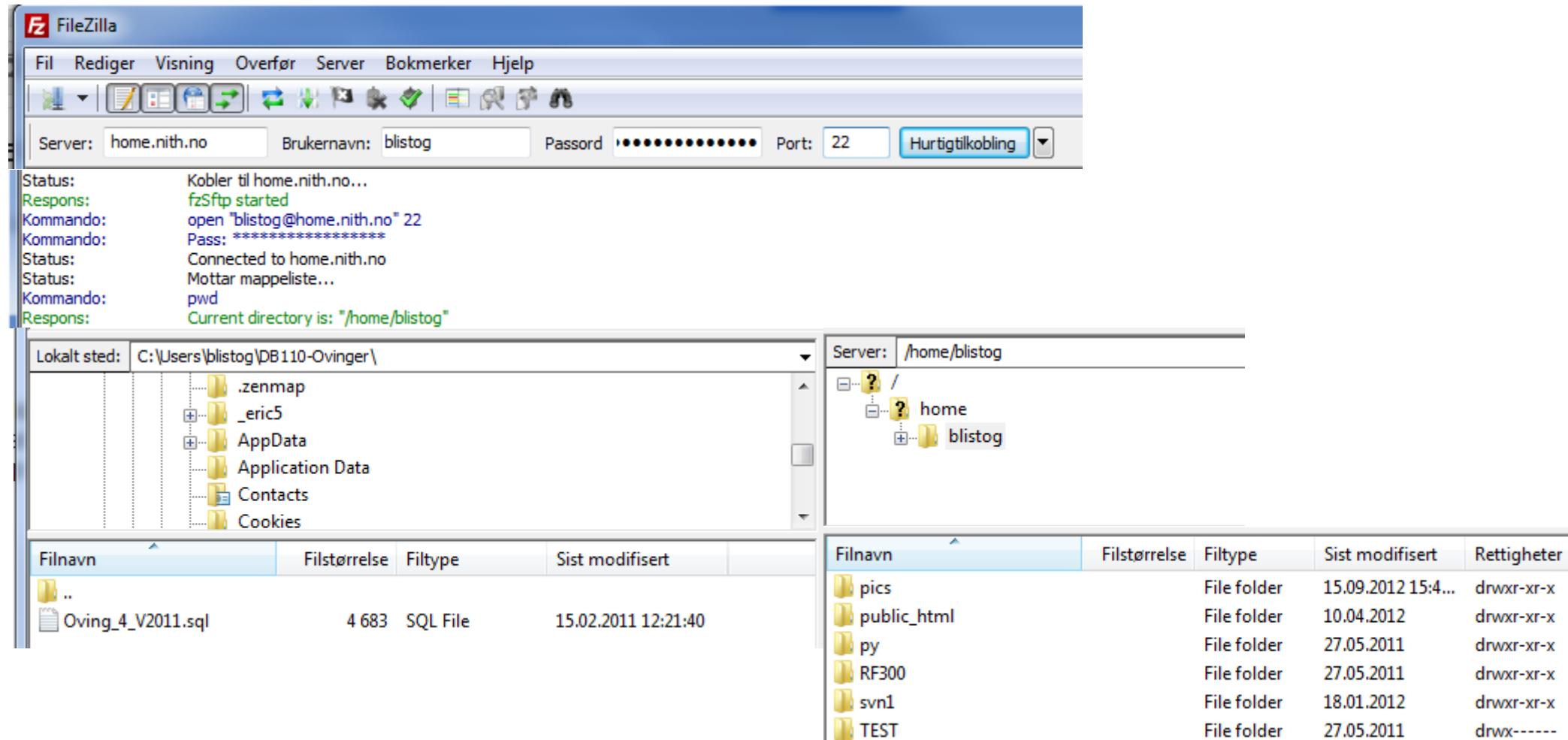
FTP kommandoer og returkoder

- 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

- **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



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 spørninger og svar
 - Tilstandsløshet og konsekvenser av det
- SMTP
 - Meldingsutveksling og syntax
 - Forskjell på SMTP-kommandoer og Epost-header

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)

Øvingsoppgaver

- Oppgavesett multiple-choice
- Teste hosts fil i praksis, og lære om HSTS
- Sende SMTP mail manuelt (Putty)
- Sende HTTP requests manuelt (Putty og Curl)
- Bruke FTP i shell (ftp)
- Inspisere applikasjonslag trafikk i Wireshark

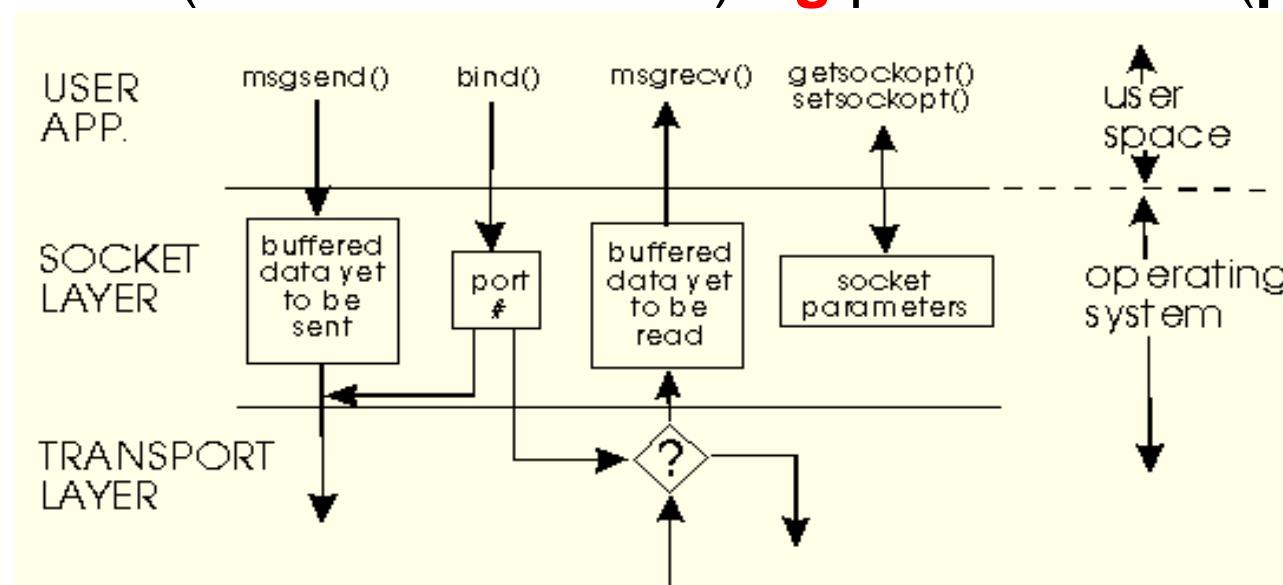
For valgfritt egenstudie

For de som ønsker å lære noen emner mer i dybden for å forstå det bedre er det her samlet noen ekstra temaer relatert til dagens undervisning, det må forventes en del egenarbeid for å forstå disse emnene.

Det vil ikke komme spørsmål på eksamen fra disse, og dette er altså ikke ansett for å være en del av pensum.

Sockets (API)

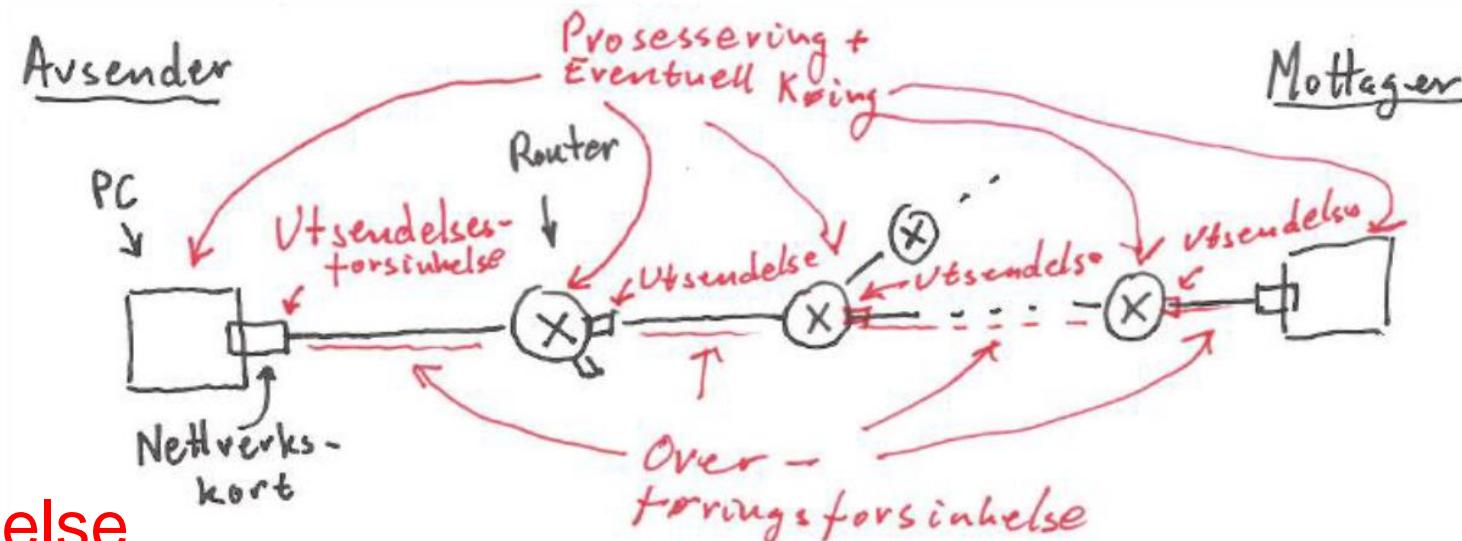
- Definerer forbindelsen ("grensesnittet") mellom applikasjons- og transport-laget
- **Socket** = "Internett API"
 - To prosesser kommuniserer med hverandre over Internett ved å sende data inn i socket og lese data ut fra socket
- Adresse til ønsket kommunikasjons-partner dannes av IP-adresse (**vertsmaskin-«id»**)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



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 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



• 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**

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

Autoritative navnetjenere

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

DNS Records: MX

- MX-oppslug utføres for å finne 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.googlemail.com
westerdals.no  MX preference = 5, mail exchanger = alt1.aspmx.l.google.com
westerdals.no  MX preference = 10, mail exchanger = aspmx2.googlemail.com
westerdals.no  MX preference = 5, mail exchanger = alt2.aspmx.l.google.com
westerdals.no  MX preference = 10, mail exchanger = aspmx3.googlemail.com
westerdals.no  MX preference = 10, mail exchanger = aspmx4.googlemail.com
```

```
> 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-oppslug 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**

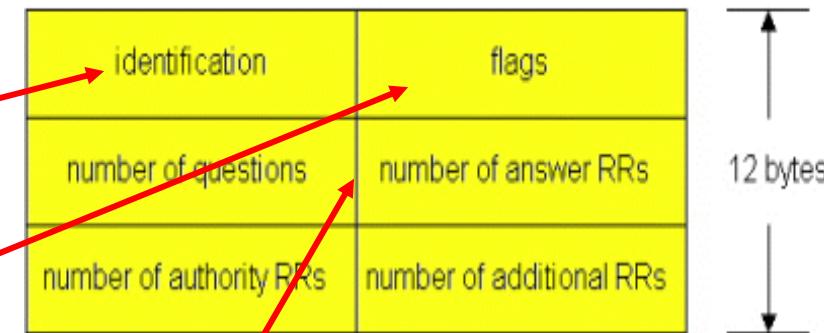
- Lar samme IP-adresse tilsvare flere ulike navn under domenet

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 spørninger og svar



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

MIME version
method used
to encode data
multimedia data
type, subtype,
parameter declaration
encoded data

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.....
.....base64 encoded data
```

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, innleddet med =-tegn...

HØR HÆR!!! [Inbox](#) | X

from **Bjørn Olav Listog** <bjorn@listog.no>
 to **Bjørn Olav Listog** <blistog@nith.no>
 cc "Bjørn Olav Listog (E-post)" <Bjorn.Listog@nith.no>
 date 8 February 2011 15:17
 subject **HØR HÆR!!!**
 mailed-by nith.no

Hvordan overføres dette mon tro?

ÆØÅ
æøå

MIME-Version: 1.0
 Sender: blistog@nith.no
 Received: by 10.229.184.3 with HTTP; Tue, 8 Feb 2011 06:17:01 -0800 (PST)
 Date: Tue, 8 Feb 2011 15:17:01 +0100
 Delivered-To: blistog@nith.no
 X-Google-Sender-Auth: riPgU9P9C-VjugelxvlikBeRs4
 Message-ID: <ANALkTinfSf=Vij2DObu-YK47C41K0ueXaz89ptsETG=@mail.gmail.com>
 Subject: =?UTF-8?B?SMOYUiBIw4ZSISEh?=

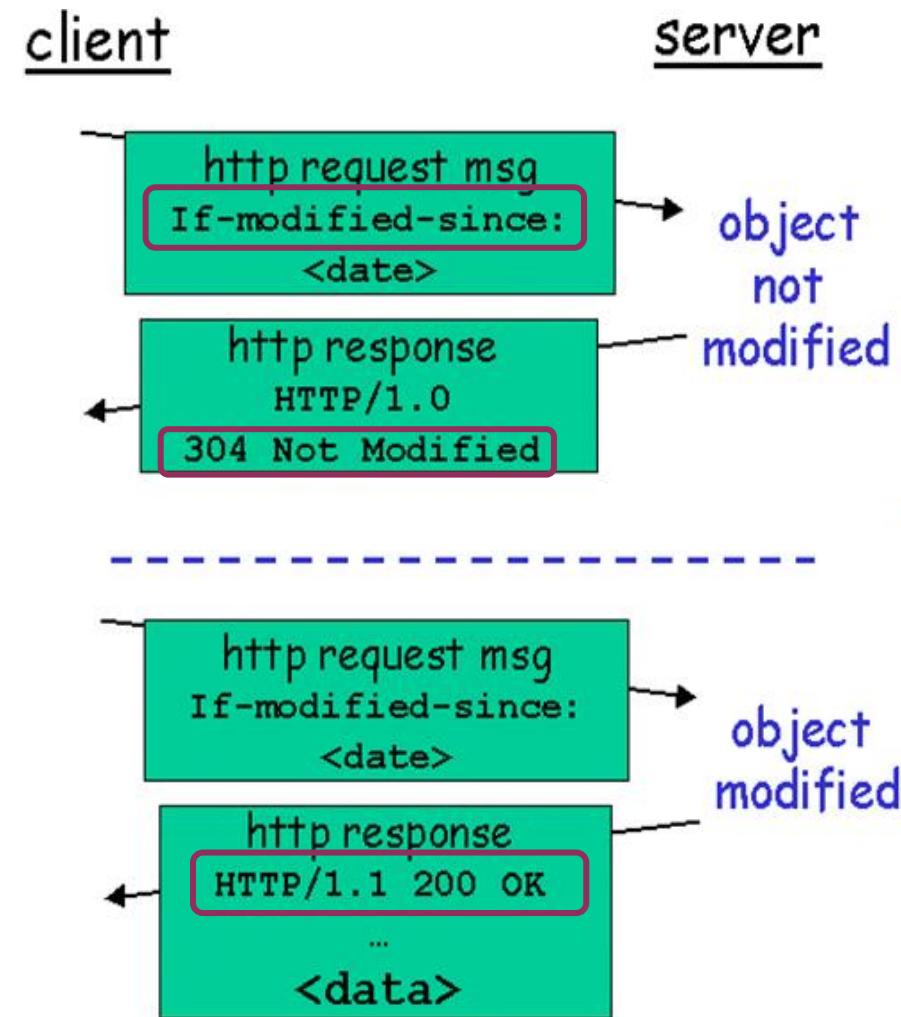
From: =?UTF-8?B?SMOYUiBIw4ZSISEh?= <bjorn@listog.no>
 To: =?UTF-8?Q?Bj=C3=B8rn_Olav_Listog?= <blistog@nith.no>
 Cc: =?UTF-8?B?QmrDuHJuIE9sYXYgTG1zdG9nIChFLXBvc3Qp?= <Bjorn.Listog@nith.no>
 Content-Type: text/plain; charset=UTF-8
 Content-Transfer-Encoding: quoted-printable

Hvordan overf=88res dette mon tro?

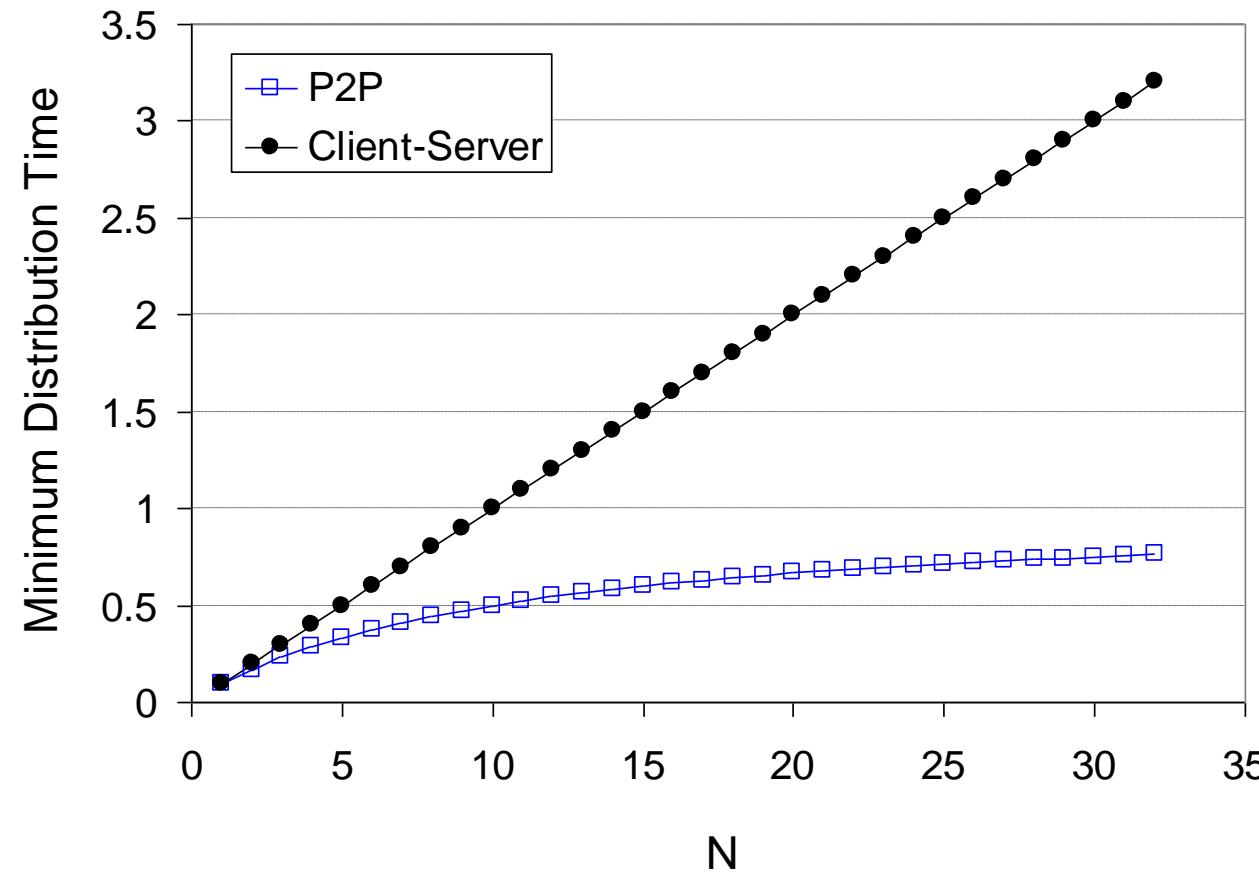
=C3=86=C3=98=C3=85
 =C3=A6=C3=B8=C3=A5

Klient-tjener kommunikasjon (Ex)

- Betinget GET
 - If-modified-since:
 - Ikke send svar hvis klient har oppdatert versjon
 - Sjekker tidsstempel 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



Klient-tjener vs. P2P

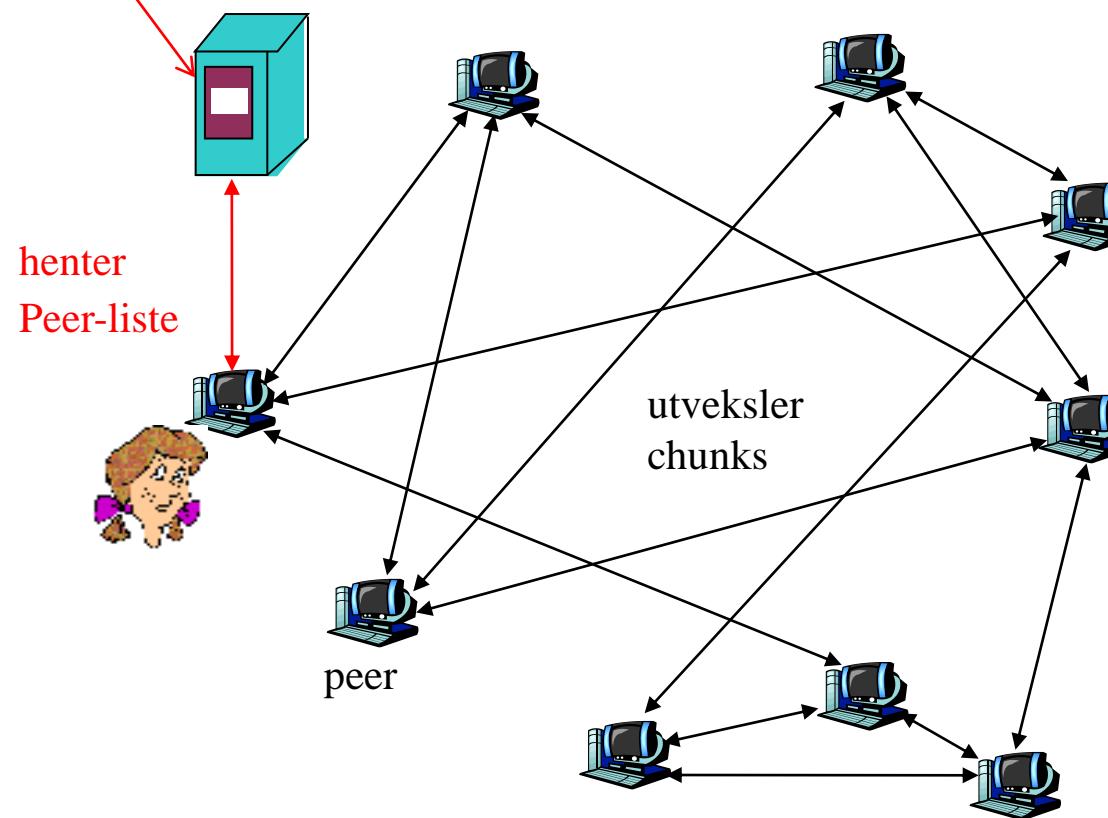


*P2P kan redusere (samlet) nedlastingstiden da det
“fjerner” **flaskehalsen inn til tjeneren***

Fildistribusjon: BitTorrent

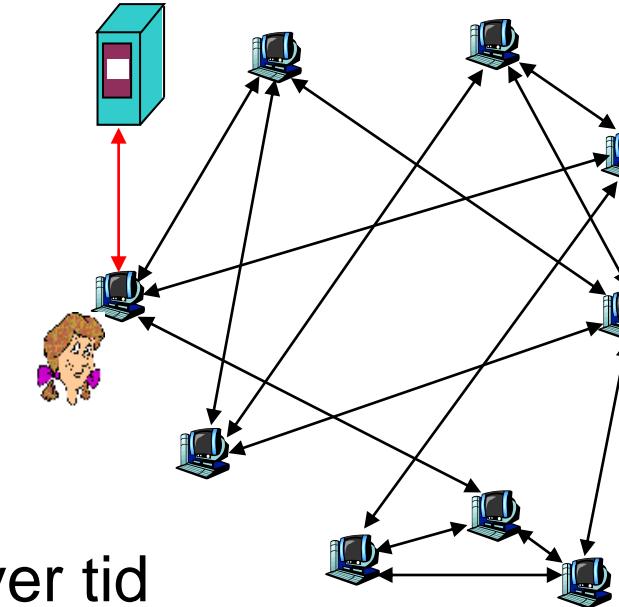
tracker: overvåker (tracks) hvilke “peers” som deltar i en torrent

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 “sjeldnest”
- Peers kommer og forsvinner som de vil.



BitTorrent (2)

Hente Chunks

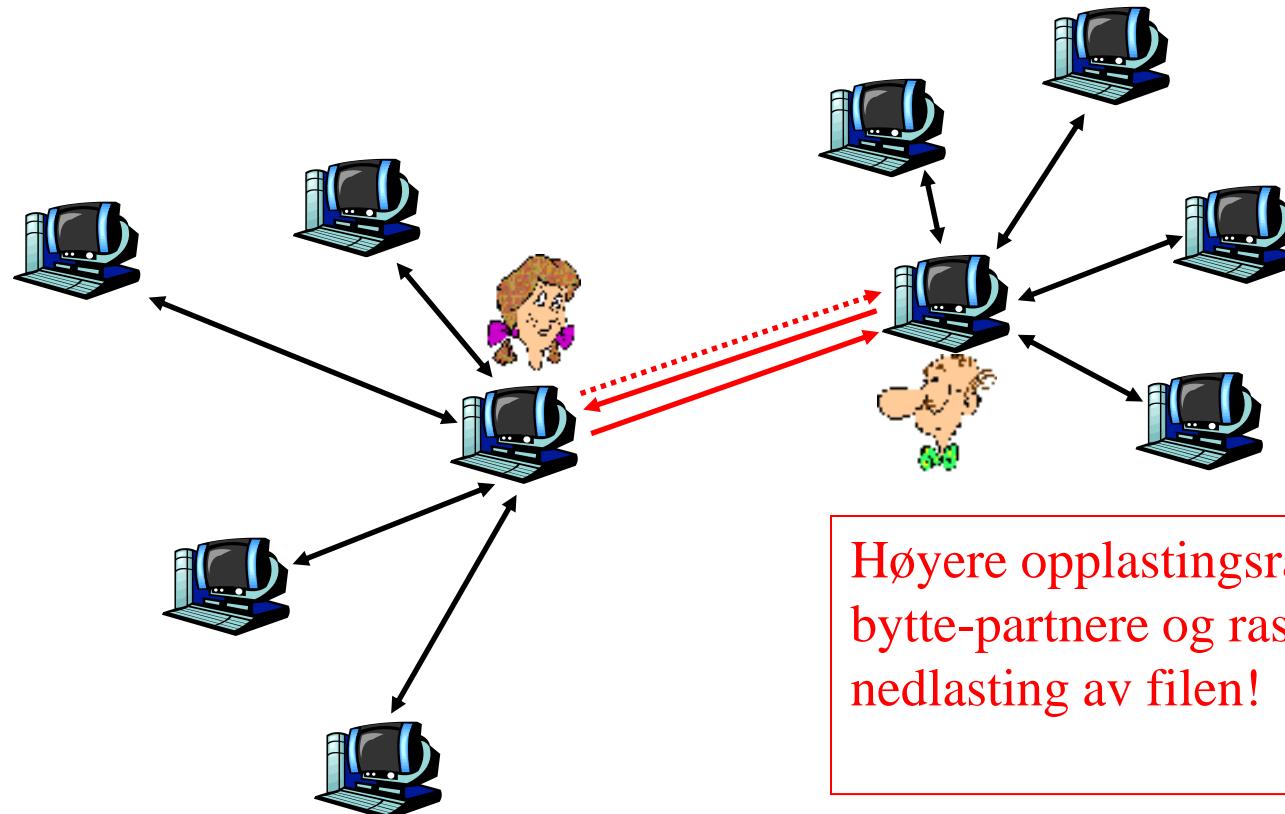
- Ulike peers har ulike deler av filen på et gitt tidspunkt
- periodisk, spør en peer (Anne) naboene om hvilke chunks de har
- Anne etterspør så de manglende delene ("chunks")
 - sjeldneste først)

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



DHT (Distribuert Hash Tabell)

- DHT er en teknikk for å opprettholde, og videreføre/fordele 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.

Denne forelesningsøkten vil bli tatt opp og lagt ut i emnet i etterkant.

Hvis du ikke vil være med på opptaket:

 ^ Start Video	La være å delta med webkameraet ditt.
 ^ Unmute	La være å delta med mikrofonen din.
To: Marianne Sundby (Privately) Type message here...	Still spørsmål i Chat i stedet for som lyd. Hvis du ønsker kan spørsmålet også sendes privat til foreleser.



Høyskolen
Kristiania

Pensum i faget er ALLE forelesninger, ALLE øvingsoppgaver (både teori oppgaver og praktiske oppgaver), og eksterne ressurser publisert på Canvas.

Her er en samlet oversikt over verktøy som kan være relevante for de praktiske oppgavene på eksamen, hentet fra de forskjellige forelesningene (hvis forskjell fra OSX og Windows er de oppgitt slik «OSX/Windows»):

- «En hex editor» for eksempel : **0xED/TinyHexer**
- Operativsystem inkl shell, valgfritt: Linux/OSX/Windows, må kunne løse oppgavene fra forelesninger og øvinger, vær obs på at alle oppgavene i PRAKTISK øvingsoppgave til 0x05 Operativsystem må mestres – med unntak av oppgave 3.3
- Alle shell handlinger vi har vært gjennom innen OS, inkl navigering, pipes, omdirigeringsoperatorer, og vanlige shell kommandoer
- **Activity Monitor / Taskmanager**
- **Arp, traceroute/tracert, ping, ifconfig/ipconfig, netstat, dig+nslookup/nslookup, ftp, route, curl, ncat**
- Applikasjon for å kjøre manuelle «rå» nettverkskommandoer; **telnet** på Linux og OSX / **PuTTY** på Windows
- **Wireshark**

TK1100

Forelesning 0x09:

Nettverkslaget

Er det noe galt her?

Wireless LAN adapter WLANUSB:

```
Connection-specific DNS Suffix . : ad.nith.no
Description . . . . . : D-Link DWA-140 Wireless N USB Adapter(rev.B3)
Physical Address. . . . . : B8-A3-86-90-50-E8
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::50e5:40ff:6794:1d5a%16 (Preferred)
IPv4 Address. . . . . : 10.21.30.228 (Preferred)
Subnet Mask . . . . . : 255.255.252.0
Lease Obtained. . . . . : 6. november 2013 15:25:30
Lease Expires . . . . . : 6. november 2013 23:25:30
Default Gateway . . . . . : 10.21.28.1
DHCP Server . . . . . : 1.1.1.1
DHCPv6 IAID . . . . . : 548971398
DHCPv6 Client DUID. . . . . : 00-01-00-01-14-6A-F2-0B-D8-D3-85-77-A0-3F
DNS Servers . . . . . : 158.36.131.10
Primary WINS Server . . . . . : 158.36.131.10
NetBIOS over Tcpip. . . . . : Enabled
```

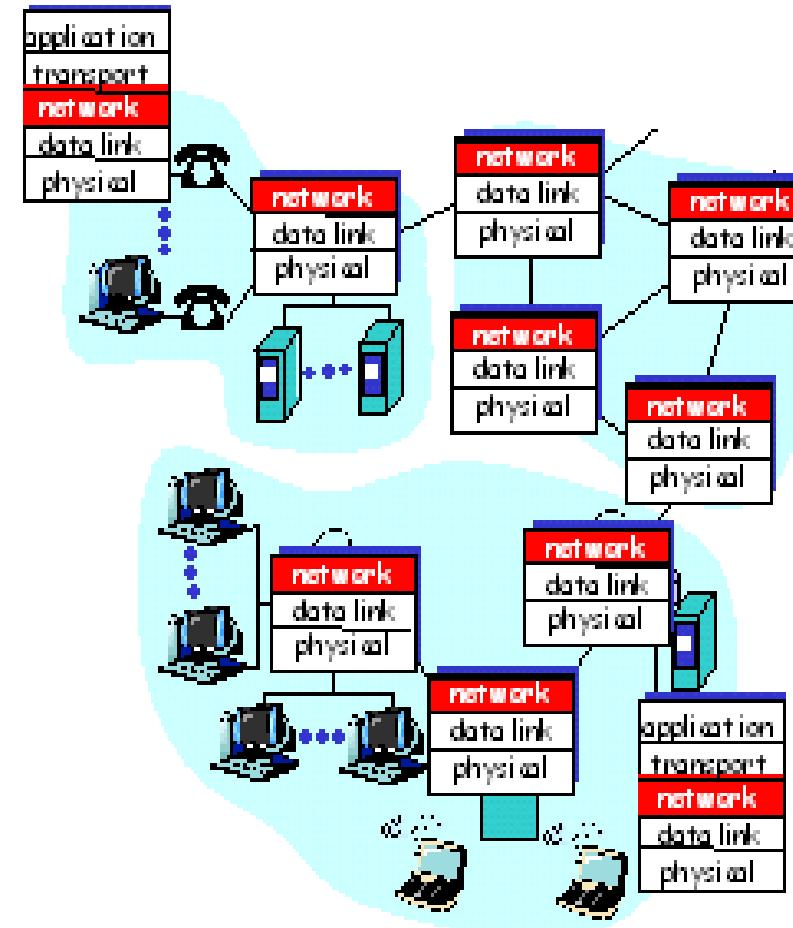
Dagsorden

- Nettlagets oppgaver
 - Prefix-/datagram-svitsjing på nettlagsnivå
- IPv4
 - headeren
 - IP-adresser og prefix-routing
 - IP fragmentering
 - Litt om DHCP
- ICMP
- NAT
- Litt om AS og routing i LAN, WAN og stamnett

Navn på laget	Betegnelse på overføringsenhet	Viktigste oppgaver/funksjoner Exempel på protokoller/standarder
Applikasjonslaget	Melding (Message)	Støtte nettverksapplikasjoner 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 Ex: TCP, UDP,..
Nettverkslaget	Datagram	Routing av datagram fra/til vertsmaskin gjennom nettverkskjernen Ex: IP (v4 og v6), ICMP, RIP, OSPF, BGP,...
Datalinjelaget	Ramme (Frame)	(Pålitelig) Levering av ramme fra nabo-node til nabo-node Ex: Ethernet II, FDDI, IEEE 802.11 ...
Fysisk	Bit	(Kode og) Flytte enkeltbit mellom kommunikasjonspartnere. Ex: 10BaseT, ...

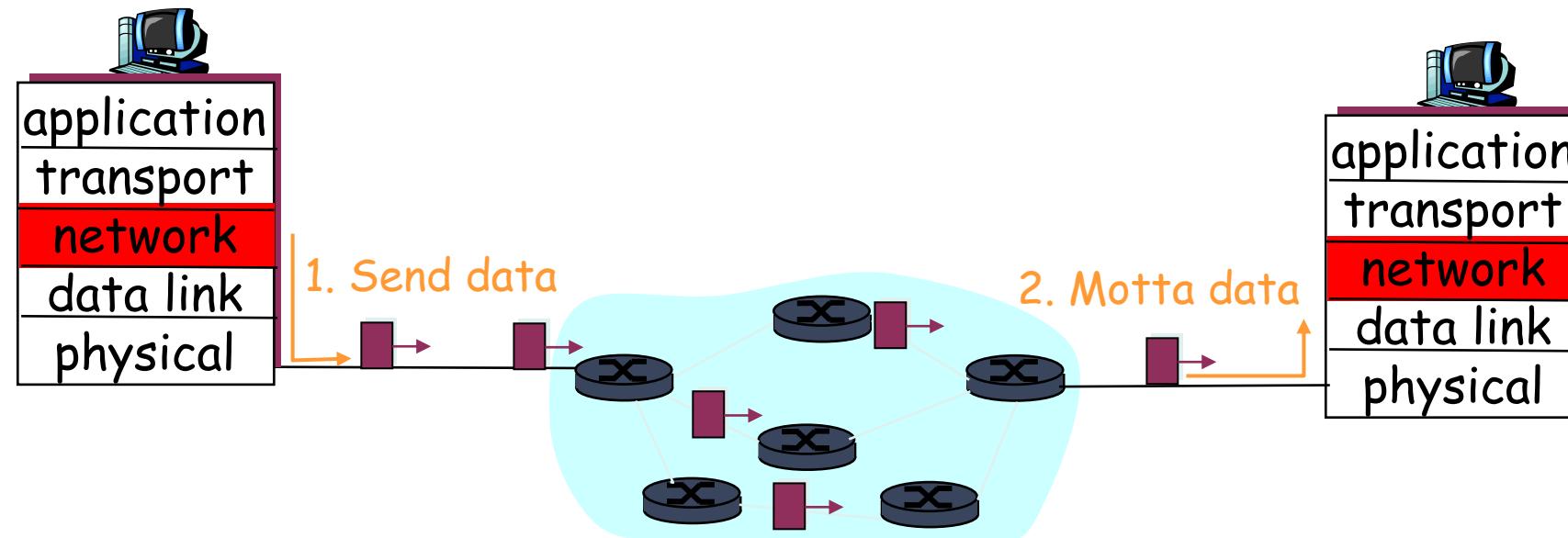
Nettverkslaget

- Flytter pakker fra avsender til mottaker
- Nettverks-protokoll også på hver mellomlanding
- Routing fra avsender til mottaker
- Switching av pakker fra routers input-side til routers output-side
- Hvis nødvendig defineres router kall oppsett for hele ruten før pakke sendes



Datagram nettverk (Internett)

- Ikke noe forhånds-oppsett på nettverklags-nivå
- Routerne bryr seg ikke om *rutens* tilstand
 - Tilstandsløse routere
- Pakkene rutes ut fra mottaker-ID
 - Pakkene mellom samme avsender og mottaker **kan** følge forskjellige ruter



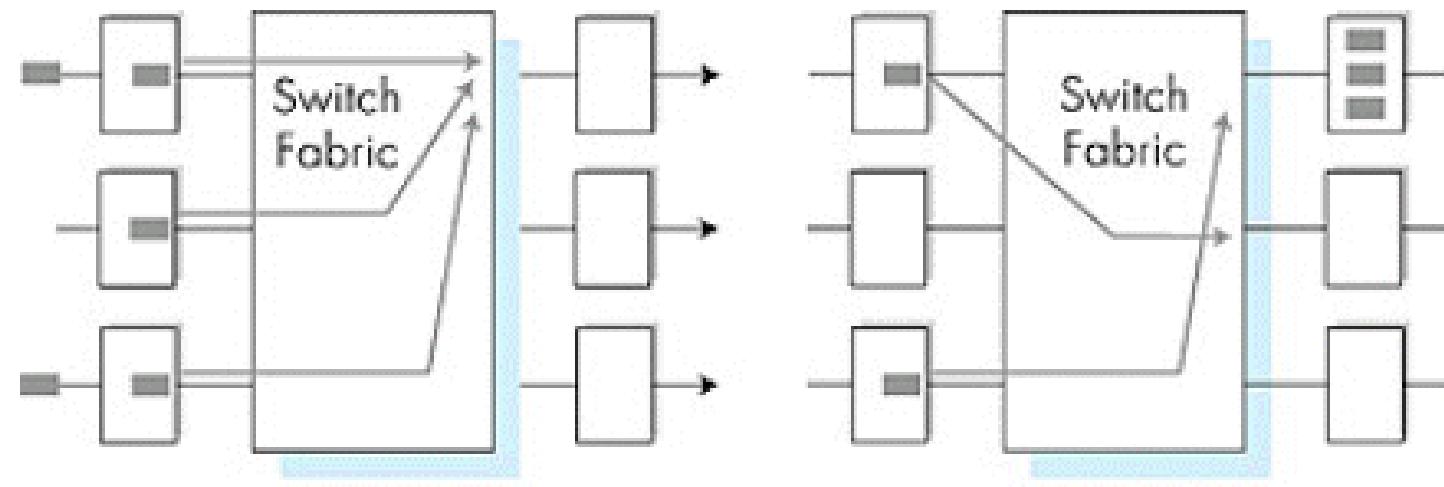
route print

```
cmd C:\WINDOWS\system32\cmd.exe
C:\>route print
=====
Grensesnittliste
0x1 ..... MS TCP Loopback interface
0x2 ...00 13 72 94 ff 78 ..... Broadcom NetXtreme 57xx Gigabit Controller - Min
iport for pakkeplanlegger
=====
=====
Aktive ruter:
Nettverksmål Nettverksmaske Gateway Grensesnitt Metrikk
      0.0.0.0      0.0.0.0 10.21.4.1 10.21.5.94 20
      10.21.4.0  255.255.252.0 10.21.5.94 10.21.5.94 20
      10.21.5.94 255.255.255.255 127.0.0.1 127.0.0.1 20
  10.255.255.255 255.255.255.255 10.21.5.94 10.21.5.94 20
      127.0.0.0      255.0.0.0 127.0.0.1 127.0.0.1 1
      169.254.0.0  255.255.0.0 10.21.5.94 10.21.5.94 30
      224.0.0.0      240.0.0.0 10.21.5.94 10.21.5.94 20
  255.255.255.255 255.255.255.255 10.21.5.94 10.21.5.94 1
Std. gateway:          10.21.4.1
=====
Faste ruter:
  Ingen
C:\>
```

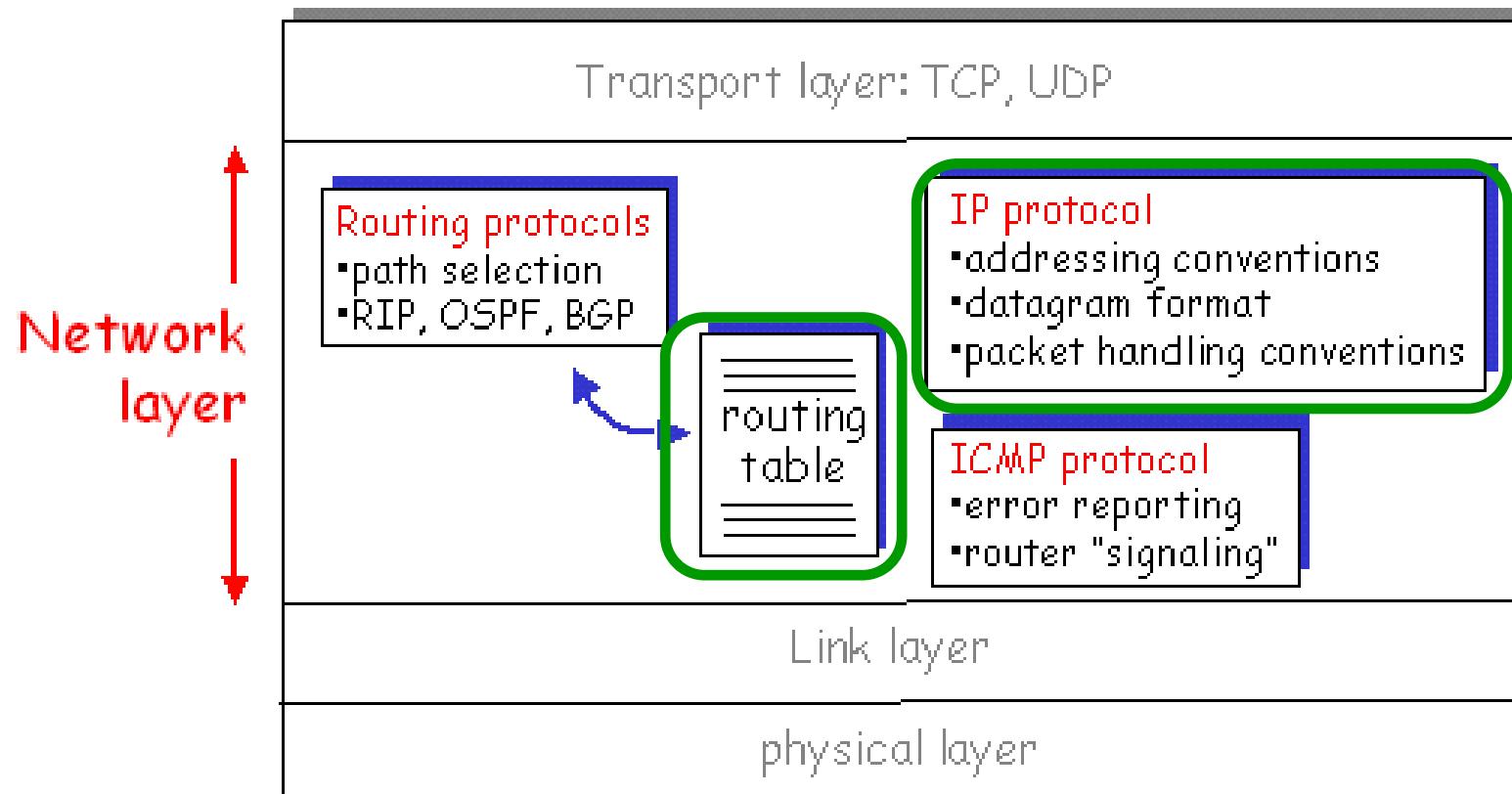
Kan også bruke netstat -r

Køing på utgående port

- Data kan gå tapt i køen hvis bufferet ikke er stort nok

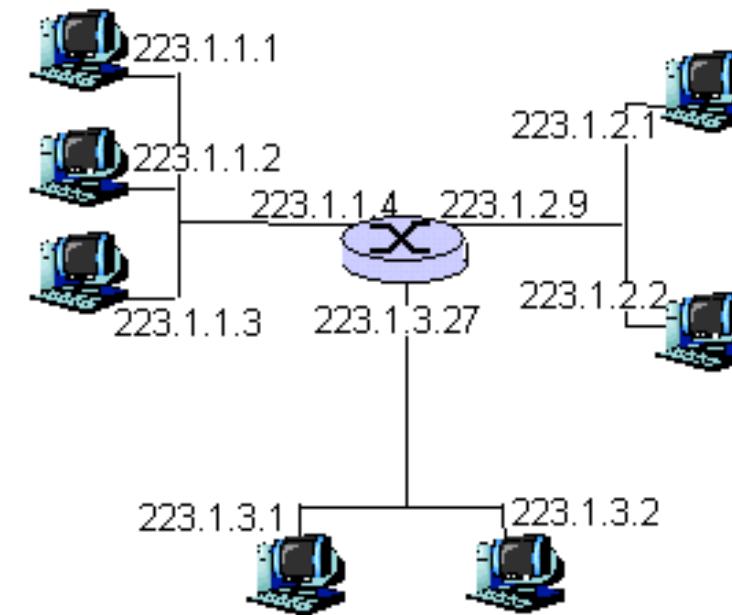


I nternet P rotocol v4



IPv4 adressering

- IPv4 adresse: **32-bit**
«**id**» for hver
vertsmaskin og router
interface (adapter)
- En vertsmaskin kan ha
flere interface
- En router har vanligvis
flere forbindelser, med
hver sin interface
- IP-adresse hører til
hvert interface



223.1.1.1 = 11011111 00000001 00000001 00000001
223 1 1 1

ipconfig (ifconfig)

- ipconfig viser nettverksparametrene for interfacene/adapterene

```
C:\Users\blistog>ipconfig
Windows IP Configuration

Ethernet adapter e0:
  Connection-specific DNS Suffix  . . . . . : 
  IPv6 Address . . . . . : 2001:700:2e00::51
  Link-local IPv6 Address . . . . . : fe80::b46c:b98f:85ec:dba0%12
  IPv4 Address . . . . . : 158.36.131.51
  Subnet Mask . . . . . : 255.255.255.128
  Default Gateway . . . . . : 2001:700:2e00::1
                                         158.36.131.1

Ethernet adapter Bluetooth Network Connection:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix' . . . . . : 

Tunnel adapter isatap.{84470FB0-16A7-4A31-B6B9-8AECF834115C}:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix' . . . . . : 

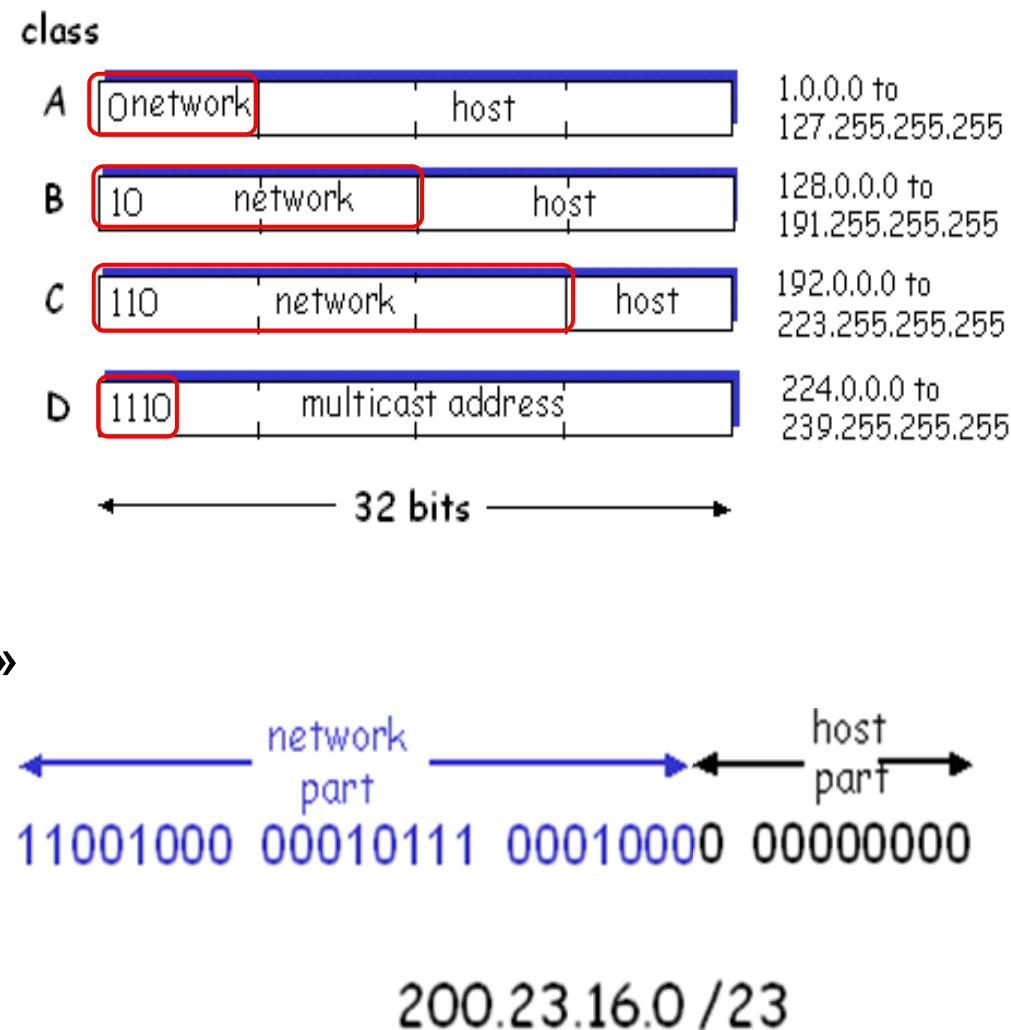
Tunnel adapter Teredo Tunneling Pseudo-Interface:
  Connection-specific DNS Suffix  . . . . . : 
  IPv6 Address . . . . . : 2001:0:5ef5:73b8:3826:3138:61db:7ccc
  Link-local IPv6 Address . . . . . : fe80::3826:3138:61db:7ccc%15
  Default Gateway . . . . . : 

Tunnel adapter isatap.{C682A4AE-3FEC-4053-8456-5A377FD0FF55}:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix' . . . . . : 
```

IPv4-adressen
&Nettmaske
= Nettverksprefix
som det routes ut fra;
Std Gateway = veien ut
i Internett

Teknikker for
å sende IPv6
gjennom IPv4
nettverk

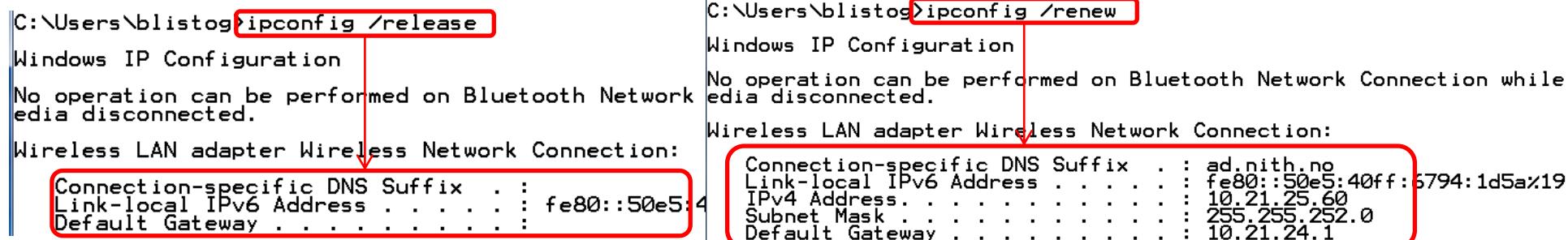
- Opprinnelig delt opp i 6 forskjellige klasser med hver sin forhåndsdefinerte **prefix**-lengde
 - Klasseinndeling av adresser ble for "stivt"
 - En klasse kan risikere å inneholde (mange) ubrukte adresser
 - Klasse A, B, C er vanlige addresser, D er multicast, E er reservert for research, og 127.* er en reservert «klasse» for loopback
 - Classless Inter-Domain Routing (CIDR)
 - Nettverks-delen har vilkårlig lengde, x
 - Format a.b.c.d/x



- For vertsmaskiner i LAN
 - Kan settes manuelt/statisk
 - **Dynamic Host Configuration Protocol(DHCP)**
- For nettverk
 - Får tildelt sin del av ISP sitt tildelte adresserom
- For Internet Service Provider (ISP)
 - Internasjonalt organ (**ICANN**) tildeler adresser, styrer DNS, tildeler domenenavn og løser tvister
 - “Kontinent-registraren”: **RIPE** deler ut IP-adresser og AS-nummer til Europa m.fl.

Dynamic Host Configuration Protocol

- Hver DHCP-tjener har et sett med mulige adresser (**pool**)
- Setter adressen dynamisk med "plug-and-play"
- **Vertsmaskin sender**: DHCP discover
- **DHCP tjener svarer**: DHCP offer
- **Vertsmaskin sender**: DHCP request
- **DHCP tjener sender**: IP-adresse og andre nettverksparameter (f.eks. DNS-tjener) + DHCP ack
- **Vertsmaskin settes opp med disse verdiene**



```
C:\Users\blistost>ipconfig /release
Windows IP Configuration
No operation can be performed on Bluetooth Network Connection while media disconnected.

Wireless LAN adapter Wireless Network Connection:
  Connection-specific DNS Suffix . . . . . : ad.nith.no
  Link-local IPv6 Address . . . . . : fe80::50e5:40ff:fe80::50e5:40ff%19
  Default Gateway . . . . . : 10.21.25.60

C:\Users\blistost>ipconfig /renew
Windows IP Configuration
No operation can be performed on Bluetooth Network Connection while media disconnected.

Wireless LAN adapter Wireless Network Connection:
  Connection-specific DNS Suffix . . . . . : ad.nith.no
  Link-local IPv6 Address . . . . . : fe80::50e5:40ff:fe80::50e5:40ff%19
  IPv4 Address . . . . . : 10.21.25.60
  Subnet Mask . . . . . : 255.255.252.0
  Default Gateway . . . . . : 10.21.24.1
```

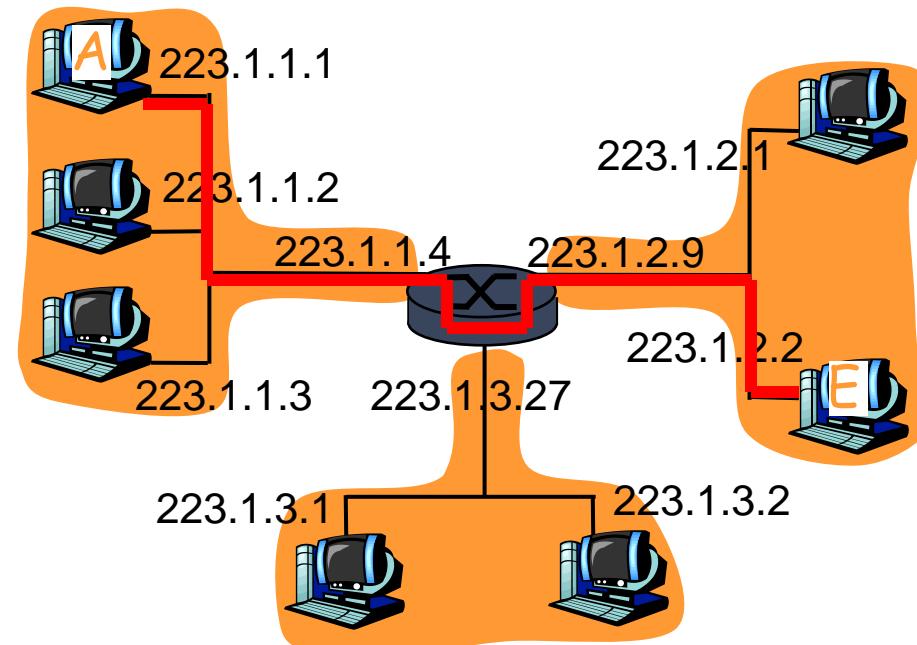
- Hvordan vet DHCP-serveren hvor den skal sende dine nettverksparametere (IP, nettmaske, std gw, DNS m.m.)?
 - Din maskin kringkaster (MAC-adresse: FF-FF-FF-FF-FF-FF) den første forespørselen i LANet
 - Dersom det finnes en DHCP-server der, så svarer den med et tilbud om IP m.m.
 - Resten kan da foregå på Nettverkslaget
 - Setter en periode du «leaser» parameterene for
 - Må fornyes når leasen går ut.

Datagram fra avsender til mottaker

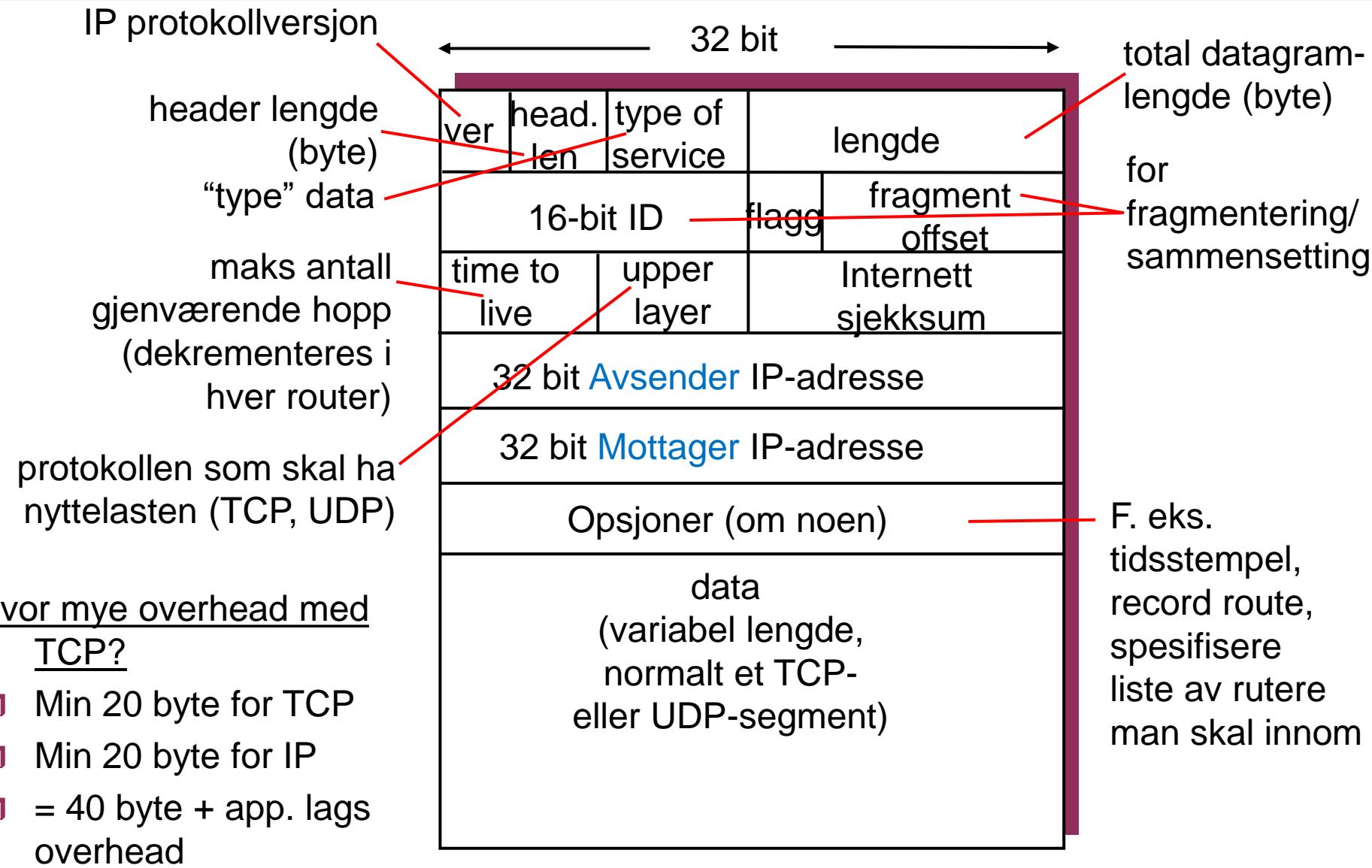
Avsender A, mottaker E

- Finn nettverksadresse til E
- E på annet nettverk
 - A, E ikke direkte forbundet
- Ruting tabell: neste hopp router til E er 223.1.1.4
- Link laget sender datagram til router 223.1.1.4 i link-lagets ramme
- Datagram ankommer 223.1.1.4
- E på samme nettverk som 223.1.2.9
- Datagram sendes til 223.1.2.2

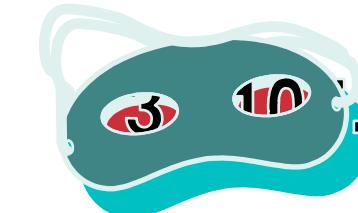
ulike felt	kilde IP addr	mottag IP addr	data
---------------	------------------	-------------------	------



IPv4 datagram-format

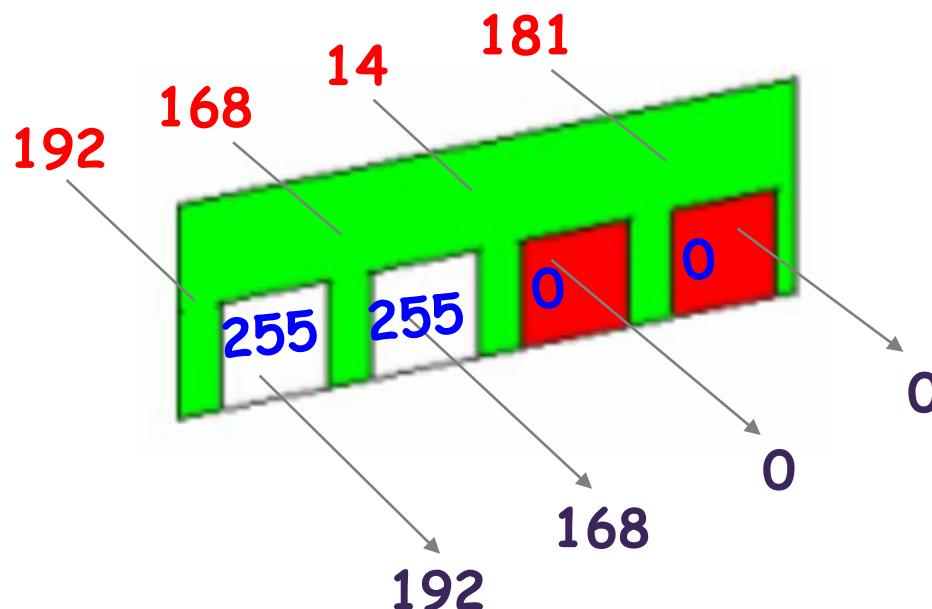


Nettmaske



3.27

- Nettmasken angir hvilke bit som er PREFIX og hvilke som er HOST
- En nettmaske er en bitmaske anvendt på en IP-adresse
 - Adresse 192.168.14.181, maske 255.255.0.0

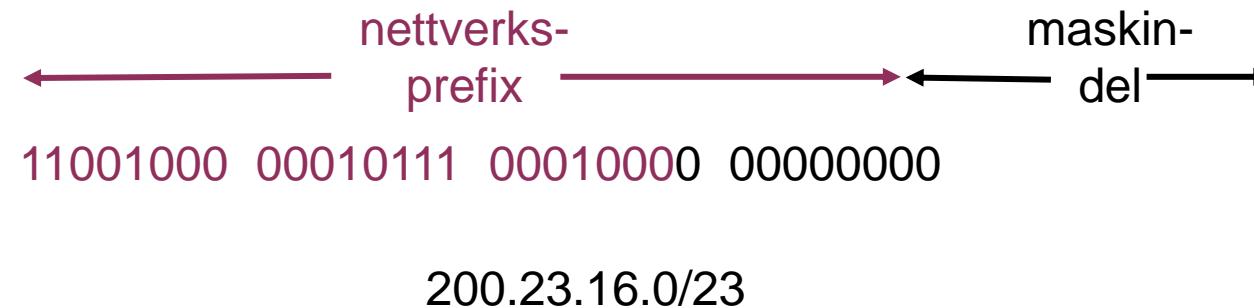


$$\begin{array}{r} 192.168.14.181 \\ \text{AND} \quad 255.255.0.0 \\ \hline = \quad 192.168.0.0 \end{array}$$

IP-adresse & Nettmaske = IP-Nettverk

- Maskiner/adapttere må tilhøre samme IP-nettverk for å kunne sende direkte til hverandre
 - 10.21.3.5 / 255.255.254.0 kan sende direkte til 10.21.2.255 / 255.255.254.0
 - 10.21.3.5 / 255.255.255.0 må sende via **gateway** (router) for å nå 10.21.2.255 / 255.255.255.0
- **Prefixen** bestemmes av IP-adressen og nettmasken, og det er denne som bestemmer om man tilhører samme IP-nett eller ikke.

- “Classfull” adressering (A, B, C, D, ...):
 - ineffektiv bruk av adresserom, går fort tom for ledige adresser
 - f. eks: et klasse B nett har nok adresser til 65 000 maskiner, selv om det kun er f. eks. 2000 maskiner i nettet
- **CIDR: Classless InterDomain Routing**
 - Nettverksdel (prefix) av adressen er av vilkårlig lengde
 - adresseformat: **a.b.c.d/x**, hvor x er antall bit i nettverks-delen av adressen



Ex: Hvilket nettverk?

- 10.21.26.184 med nettmaske
255.255.252.0 tilhører hvilket nettverk?

10 . 21.0001 10 10.1011 1000

& 255.255.1111 11 **00.0000 0000**

10 . 21.0001 10 00.0000 0000

22 bit til **prefix**, 10 bit til **host**

Nettverket er **10.21.24.0/22**

Laveste adresse er **10.21.24.1**

Broadcast er **10.21.27.255**

alle host-bit satt til 1!!!

Spesielle IP-adresser

- Noen IP-adresser er reservert for spesiell bruk
 - Private adresser
 - Dokumentasjon
 - Selv-konfigurering
 - Kringkasting
 - Multicast
 - Nettverksadresse (hele lokale IP-nett)
 - Midlertidig adressering
 - Loopback (meg selv)
- Se RFC 1166

Spesielle IP-adresser (2)

- *Private adresser* brukes bare innenfor et WAN
 - kan **ikke routes** utenfor LAN/WAN
 - droppes automatisk av Internett-routere
- Gir fleksibilitet for organisasjoner internt
- Samme adresse *kan* også ha ekstern IP (NAT)

IPv4 adresser	Nettverk	<u>RFC 1918</u>
10.0.0.0 – 10.255.255.255	1 klasse A nettverk	
172.16.0.0 – 172.31.255.255	16 klasse B nettverk	
192.168.0.0 – 192.168.255.255	65536 klasse C nettverk	

Spesielle IP-adresser (3)

- I *dokumentasjon* skal man bruke adresser som ikke benyttes noe annet sted
 - 192.0.2.0/24
 - 198.51.100.0/24
 - 203.0.113.0/24
- Ved *selv-konfigurering* av IP-adresse kan det hende at DHCP-serveren er utilgjengelig.
 - bruker da en «automatisk», spesiell adresse:
 - **169.254.1.0 – 169.254.254.254** (/16)
 - Disse er heller ikke route-bare
 - Oftest kan disse tolkes som at det er problemer med å få kontakt med DHCP-server, eller at du ikke har tilgang til LAN

Spesielle IP-adresser (5)

- *Multicast* er det samme som kringkasting begrenset til en gruppe noder i en liste (som ligger på router)
 - 224.0.0.0 – 239.255.255.255

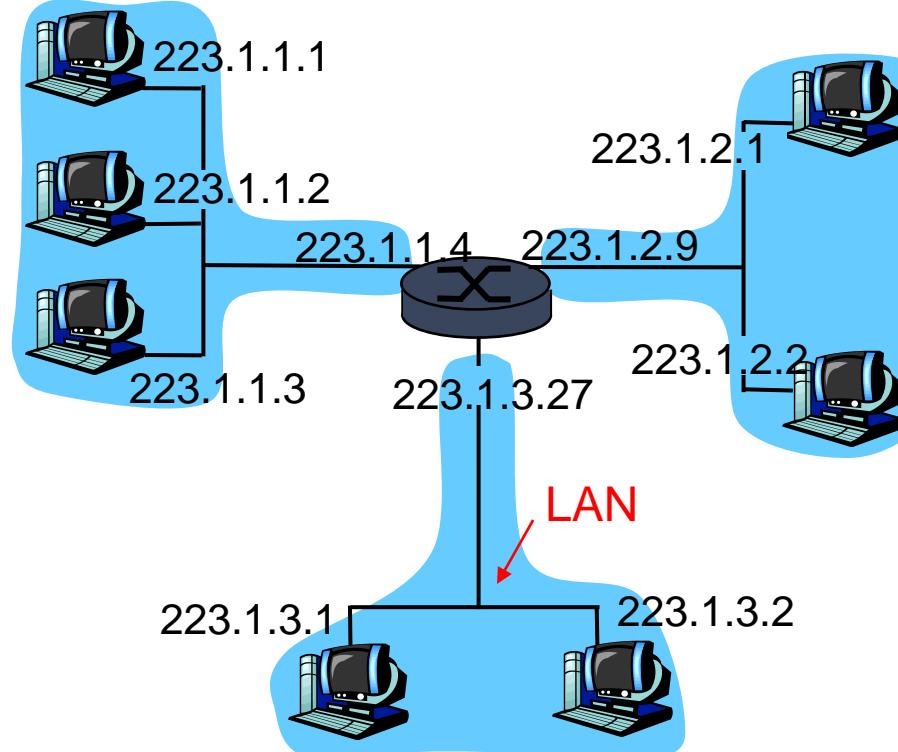
Spesielle IP-adresser

- Ved booting *kan* en maskin identifisere seg med en *midlertidig adresse*
 - 0.0.0.0 (default route)
- *Loopback* betyr å adressere seg selv
 - 127.0.0.1
 - På mange systemer brukes 127.0.0.0/8
- Laveste (nettverket) og høyeste (broadcast) adresse brukes i vanlige adressering av vertsmaskiner (hosts) eller routere



Subnett

- IP-adresser – to deler:
 - subnettdel (mest signifikante bits, bits i venstre ende)
 - maskindel (minst signifikante bits, bits i høyre ende)
- *Hva er et subnett?*
 - grensesnitt med lik subnettdel av IP-adressen
 - kan nå hverandre fysisk uten å gå via ruter

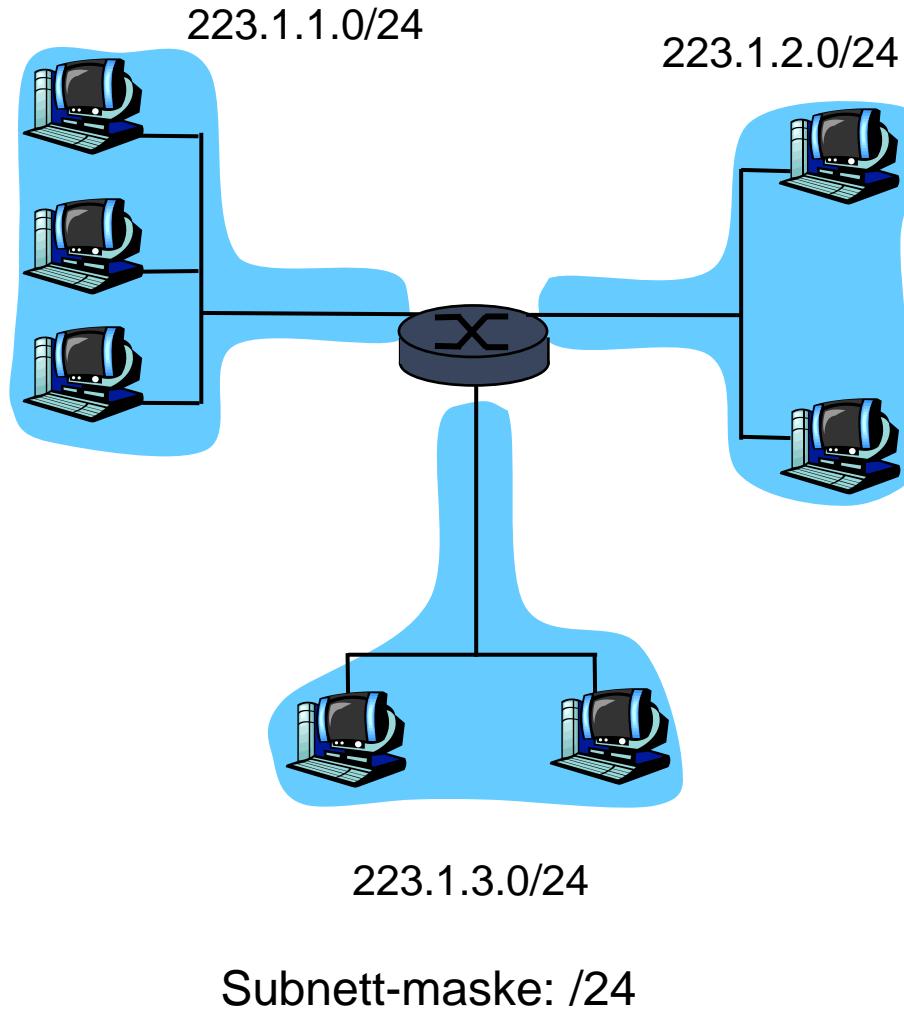


nettverk bestående av 3 IP-nett

Subnett

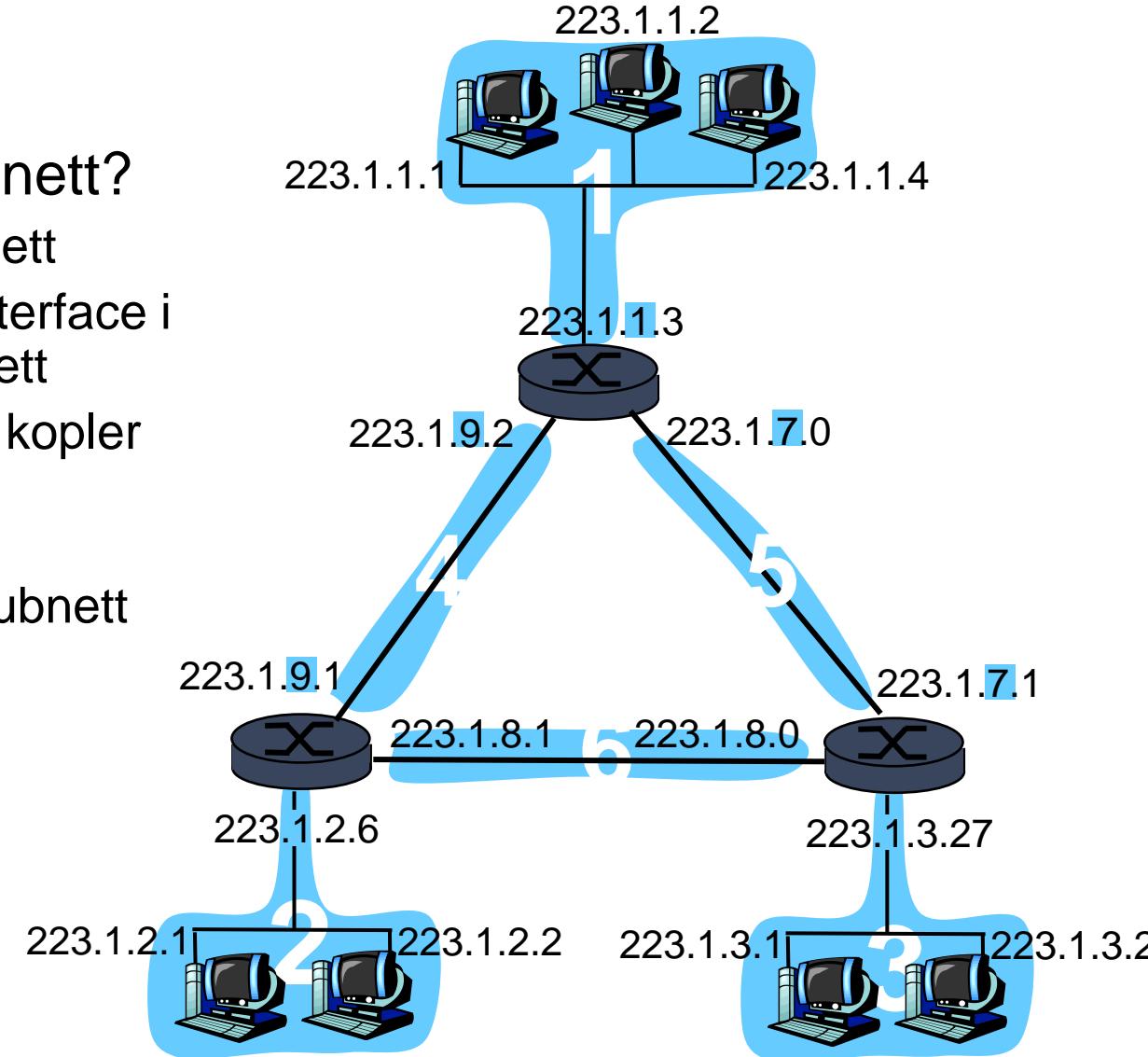
Oppskrift

- For å finne subnettene, koble hvert interface fra sin maskin eller ruter slik at vi får øyer av isolerte nett. Hvert isolerte nett kalles da et **subnett**.
- Maskiner på ulike subnett må da ha en router i mellom for å få kontakt med hverandre



Subnett

- Hvor mange subnett?
 - 3 subnett/lokalnett
 - Routerne har interface i forskjellige IP-nett
 - 3 “linknett” som kopler sammen de tre subnettene
 - 6 ulike Ip-nett/subnett



I nternet C ontrol M essage P rotocol

ICMP - Internet Control Message Protocol

- Brukes av host, router og gateway
 - Feil-rapportering
 - Ekko forespørsel/svar (ping)
- Nettverkslag ”over” IP
 - ICMP multiplekses med datagrammet
 - ICMP-melding
 - Type, kode og første 8 byte i datagrammet med feilen
 - ping og traceroute utnytter ofte ICMP

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

tracert / traceroute

- Applikasjon som setter **TTL** feltet i IP-headeren først til 1, så 2, så 3 osv
- Utløser da ICMP tilbakemelding type 11 fra hver router langs veien
- Nyttig til å sjekke hvor på ruten forsinkelser/problem kan ha oppstått

```
~~>traceroute google.com
traceroute to google.com (173.194.32.51), 30 hops max, 60 byte packets
 1  stolav-gw4.uninett.no (158.36.84.169)  1.990 ms  1.947 ms  2.133 ms
 2  stolav-gw2.uninett.no (128.39.230.137)  2.118 ms  2.102 ms  2.187 ms
 3  dk-uni.nordu.net (109.105.102.25)  10.464 ms  10.453 ms  10.442 ms
 4  se-tug.nordu.net (109.105.97.9)  18.141 ms  18.129 ms  18.114 ms
 5  se-tug2.nordu.net (109.105.97.18)  18.100 ms  18.089 ms  18.074 ms
 6  google-gw.nordu.net (109.105.98.6)  18.059 ms  17.214 ms  17.207 ms
 7  216.239.43.122 (216.239.43.122)  17.770 ms  17.751 ms  17.747 ms
 8  216.239.43.255 (216.239.43.255)  18.858 ms  18.854 ms  18.850 ms
 9  arn06s02-in-f19.1e100.net (173.194.32.51)  17.992 ms  18.352 ms  18.335 ms
```

```
C:\>tracert google.com
Tracing route to google.com [173.194.32.51]
over a maximum of 30 hops:
 1  <1 ms    <1 ms    <1 ms  stolav-gw4.uninett.no [158.36.84.169]
 2  <1 ms    1 ms    <1 ms  stolav-gw2.uninett.no [128.39.230.137]
 3  9 ms    9 ms    9 ms  dk-uni.nordu.net [109.105.102.25]
 4  16 ms   16 ms   16 ms  se-tug.nordu.net [109.105.97.9]
 5  16 ms   16 ms   17 ms  se-tug2.nordu.net [109.105.97.18]
 6  17 ms   16 ms   17 ms  google-gw.nordu.net [109.105.98.6]
 7  17 ms   17 ms   17 ms  216.239.43.122
 8  18 ms   17 ms   17 ms  216.239.43.255
 9  17 ms   17 ms   17 ms  arn06s02-in-f19.1e100.net [173.194.32.51]

Trace complete.
```

ping

- Sender en ICMP-ekkopakke til adressen man spesifiserer
- Nyttig til å sjekke om IP-adressen finnes og er mulig å nå.

```
~->ping vg.no
PING vg.no (195.88.55.16) 56(84) bytes of data.
64 bytes from www.vg.no (195.88.55.16): icmp_req=1 ttl=251 time=0.801 ms
64 bytes from www.vg.no (195.88.55.16): icmp_req=2 ttl=251 time=0.817 ms
64 bytes from www.vg.no (195.88.55.16): icmp_req=3 ttl=251 time=0.824 ms
^C
--- vg.no ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.801/0.814/0.824/0.009 ms
```

```
C:\>ping vg.no

Pinging vg.no [2001:67c:21e0::16] with 32 bytes of data:
Reply from 2001:67c:21e0::16: time<1ms
Reply from 2001:67c:21e0::16: time<1ms
Reply from 2001:67c:21e0::16: time<1ms
Reply from 2001:67c:21e0::16: time<1ms

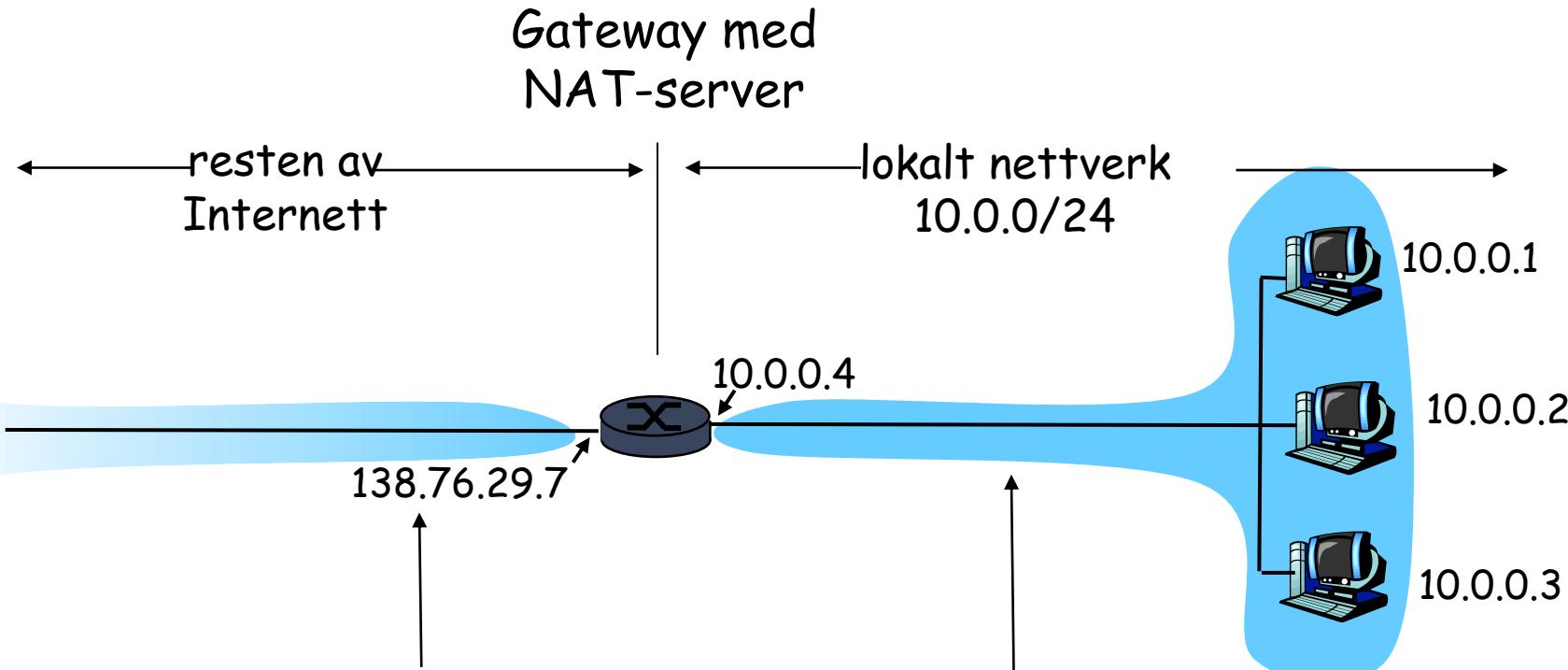
Ping statistics for 2001:67c:21e0::16:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Network Address Translation

NAT: Network Address Translation

- **Hvorfor?:** LANet har kun en/noen få IP-adresse fra Internettets perspektiv:
- ISP slipper å tildele et adresseområde:
 - kun en/noen få IP-addresse(r) for en hel organisasjons nett
- Kan endre adresser innenfor LAN uten å måtte informere omverdenen om det
- Kan skifte ISP uten å måtte endre adresser i LANet
- Utstyr i LANet er direkte adresserbare eller synlige for utenforstående (bedre sikkerhet)

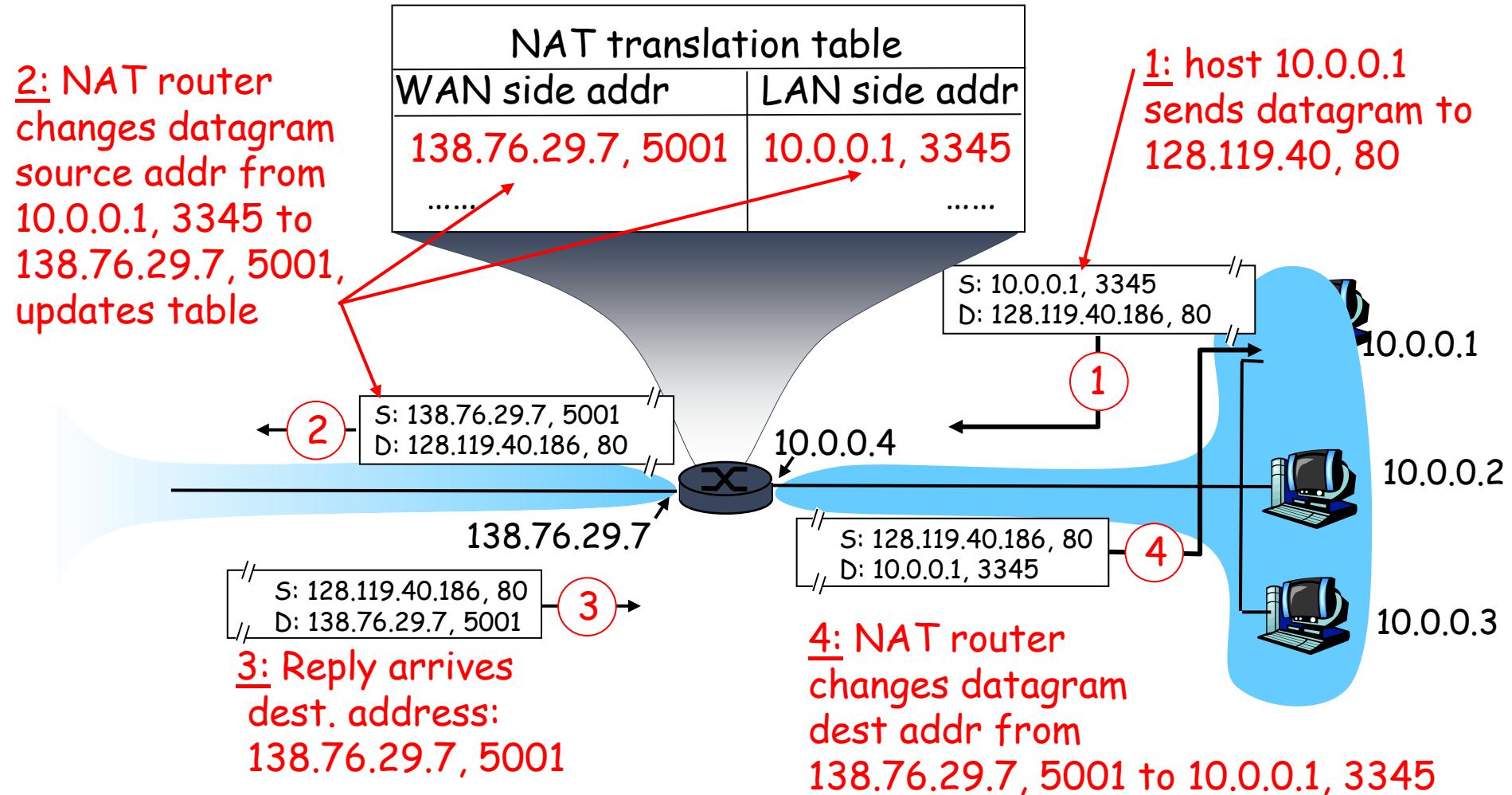
NAT: Network Address Translation



Alle datagram som **forlater** LAN
har **samme** avsender IP
adresse: f.eks. 138.76.29.7,
Ulike avsender-portnummer

Datagram avsender eller
mottager innenfor dette nettverket
har 10.0.0/24 adresse for
kilde, mål (som vanlig)
Bruker (typisk) PRIVATE ADRESSER
(10.x.x.x, 192.168.x.x,...)

NAT: Network Address Translation



NAT: Network Address Translation

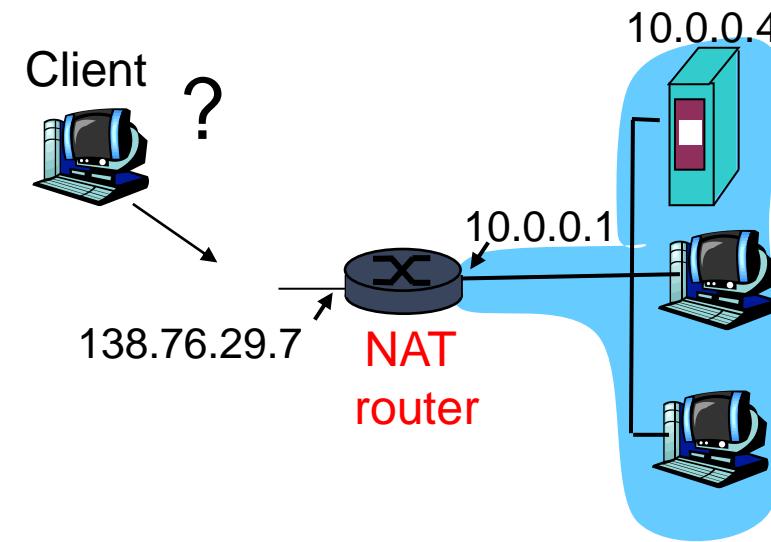
Implementasjon:

NAT-router:

- *utgående datagrammer: erstatte* avsender IP-adresse og portnummer med NAT IP-adresse og nytt portnummer
 - ... maskiner som svarer vil da bruke NAT IP-adresse og det nye portnummer som mottageradresse.
- *huske (i NAT translasjonstabell)* hvert (avsender IP-adresse, portnummer) til (NAT IP-adresse, nytt portnummer) oversettelsepar
- *innkommende datagrammer: erstatte* NAT IP-adresse og det nye portnummeret i mottagerfelter med de korresponderende avsender IP-adresse og portnummer lagret i NAT-tabell

NAT traversering problemet

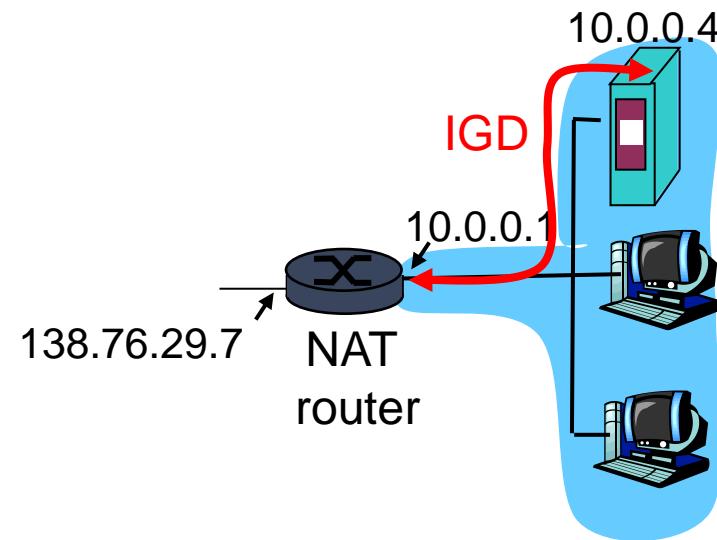
- Ekstern klient vil til server med adresse 10.0.0.4
 - server addressen 10.0.0.4 er lokal på LANet (klienten kan ikke bruke den som mottageradresse)
 - Bare en eksternt synlig NATet adresse: 138.76.29.7
- Løsning 1: statisk konfigurere NAT til å vidersende innkommende forbindelseforespørsler til en bestemt port på serveren
 - F.eks., (123.76.29.7, port 2500) alltid til 10.0.0.4 port 25000



NAT traversal problem

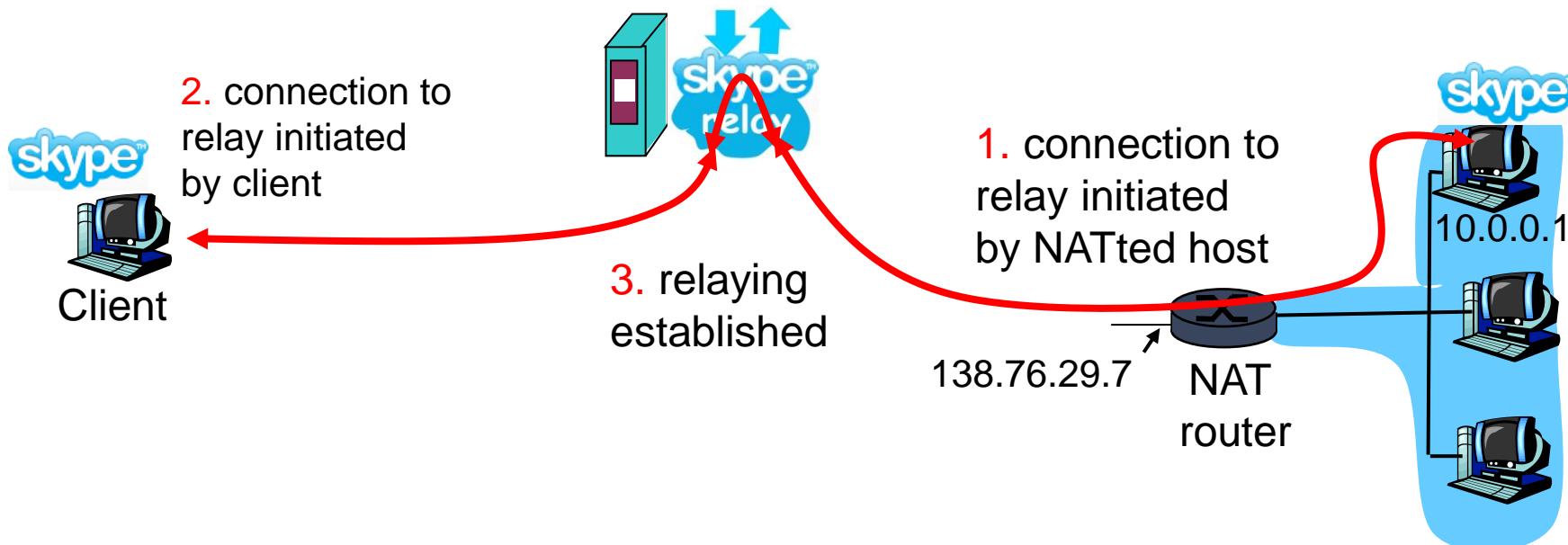
- Løsning 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) protokoll. Tillater NATet maskin å:
 - lære offentlig IP adresse (138.76.29.7)
 - Legge til/fjerne portkartlegginger (med lease tider) på router

mao, automatiser statisk NAT “port map configuration”

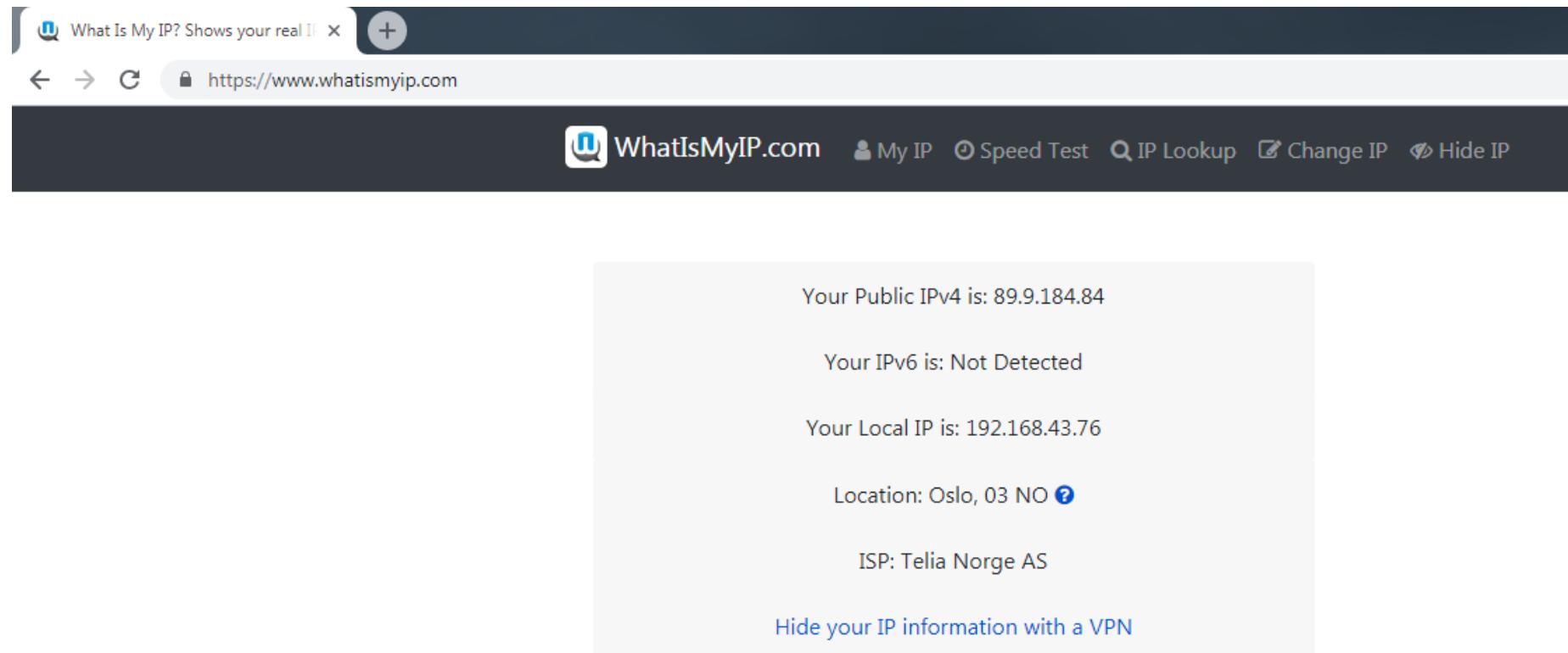


NAT traversal problem

- Løsning 3: relaying (f.eks. Skype)
 - NATtet klient etablert forbindelse til relay
 - Extern klient oppretter forbindelse til relay
 - relay videresender pakker mellom to forbindelser



Finne din eksterne IP



What Is My IP? Shows your real IP +/-

https://www.whatismyip.com

WhatIsMyIP.com My IP Speed Test IP Lookup Change IP Hide IP

Your Public IPv4 is: 89.9.184.84

Your IPv6 is: Not Detected

Your Local IP is: 192.168.43.76

Location: Oslo, 03 NO ?

ISP: Telia Norge AS

Hide your IP information with a VPN

Oppsummering

Skal nå kunne?

- Forklare hva en **IPv4 adresse** og **nettmaske** er
- Finne ut hvilket **IP-nett** en gitt IP-adresse tilhører, og beregne **broadcast**-adressen
- Kjenne igjen **localhost**, **private** og **selvkonfigurerete** adresser (**IPv4**)
- Forklare rollen til **Standard Gateway**

Skal nå kunne?

- Beskrive virkemåten til **DHCP**
- Beskrive virkemåten til **NAT** (NAT/PAT)
- Forklare rollen til **routere** og **routing-tabeller**
- **bruke** `ipconfig` (ifconfig), `ping`, `tracert` (traceroute) **og**
`netstat -r` (**route print**)

- Tekstoppgave sett
- Praktiske øvelser
 - ipconfig /release, ipconfig /renew (Linux/Mac: ifconfig)
 - ping
 - tracert (Linux/Mac: traceroute)
 - netstat -r
 - Wireshark – nå se på nettverkslaget
 - Wireshark – gå gjennom de siste oppgaver med HTTP, FTP osv; men se på nettverkslaget

For valgfritt egenstudie

For de som ønsker å lære noen emner mer i dybden for å forstå det bedre er det her samlet noen ekstra temaer relatert til dagens undervisning, det må forventes en del egenarbeid for å forstå disse emnene.

Det vil ikke komme spørsmål på eksamen fra disse, og dette er altså ikke ansett for å være en del av pensum.

Virtuell forbindelse (VC)

Vanlig på nettverkslaget er switchet forbindelse, men det er mulig å sette opp en fast forbindelse:

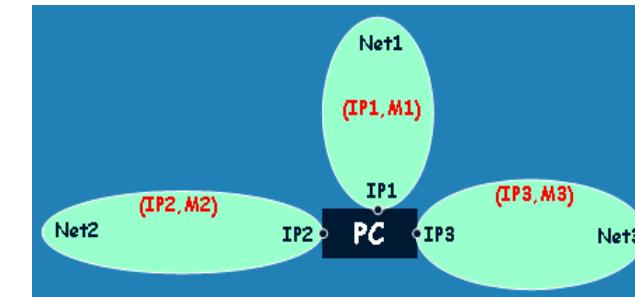
- Ligner mye på en telefon-forbindelse
- Må etablere en forbindelse (rute) først
- Hver pakke inneholder forbindelsens **rute-ID**
- Hver router (mellomlanding) opprettholder tilstanden for hver rute
 - Transport-laget ser bare hver ende av en rute
- Data overføres når ruten er etablert
- Ruten blir ”revet ned” når overføringen er ferdig

Forwardingtabell: eksempel

<u>Adresseområde for mottager</u>	<u>Link interface</u>	
11001000 00010111 00010 000 0000000	til	0
11001000 00010111 00010 111 11111		
11001000 00010111 00011 000 00000	til	1
11001000 00010111 00011 000 11111		
11001000 00010111 00011 001 0000000	til	2
11001000 00010111 00011 111 11111		
ellers		3

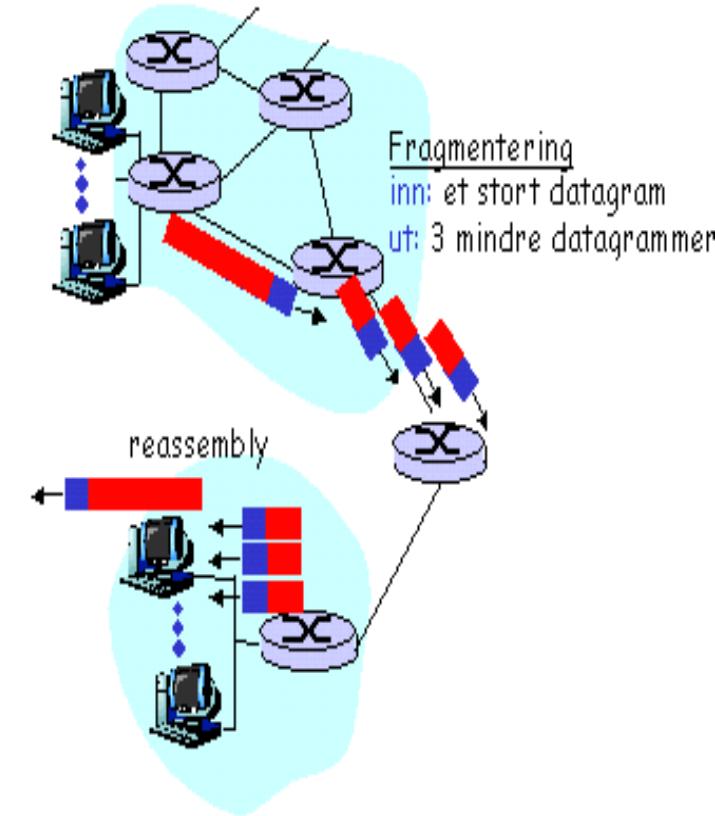
Broadcast

- Ved kringkasting (generell spørring etter en tjeneste) kan man adressere **enten** det lokale adresserommet (subnettet) **eller** hele IP-nettverket
- Begrenset kringkasting
 - 255.255.255.255
- Nettverks-kringkasting
 - Bruker nett-delen av adressen
 - 192.0.2.235/24 vil bruke 192.0.2.255
 - 192.0.2.5/27 vil bruke 192.0.2.31
 - 110000000000000000000000001000011111



IP fragmentering

- Nettverk har begrensning på pakke-størrelsen, Maximum Transfer Unit (**MTU**)
 - Alle routere må minimum takle 576 byte MTU, så det har blitt defacto pakkestørrelse
 - De fleste routere har MTU 1500 bytes (inkl ethernet header, IP header, TCP header, osv)
- Store datagrammer deles opp i mindre, selvstendige men sammenhengende datagrammer
 - Fragmenterings-flagg
 - Offset
- Settes sammen hos mottaker
 - Feiler en, feiler alle
 - TCP resender hele datagrammet



IP fragmentering

	length	ID	fragflag	offset	3980
	=4000	=x	=0	=0	

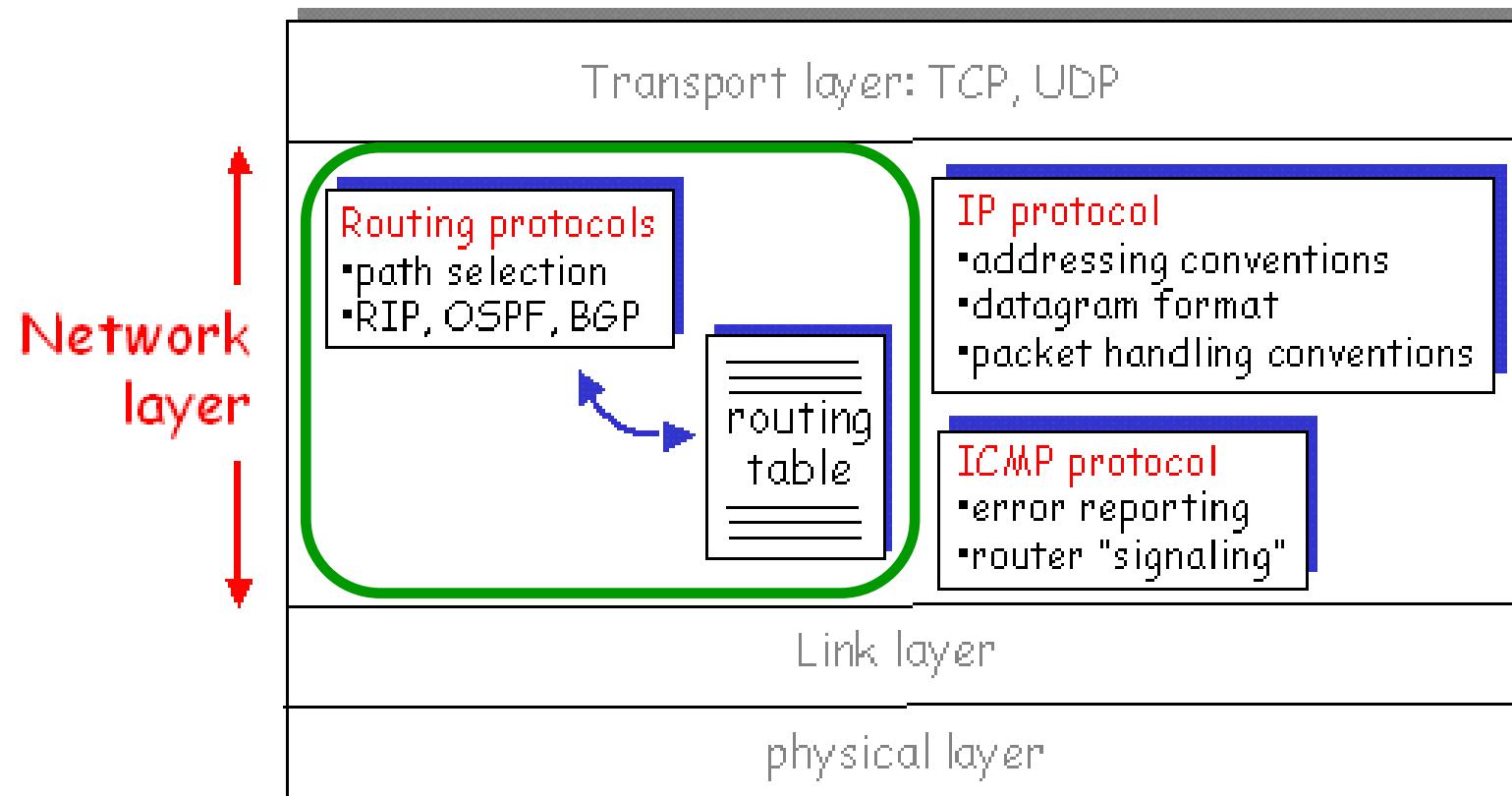
Et stort datagram blir til
mange små datagrammer

	length	ID	fragflag	offset	1480
	=1500	=x	=1	=0	

	length	ID	fragflag	offset	1480
	=1500	=x	=1	=1480	

	length	ID	fragflag	offset	1020
	=1040	=x	=0	=2960	

Routing i stamnett

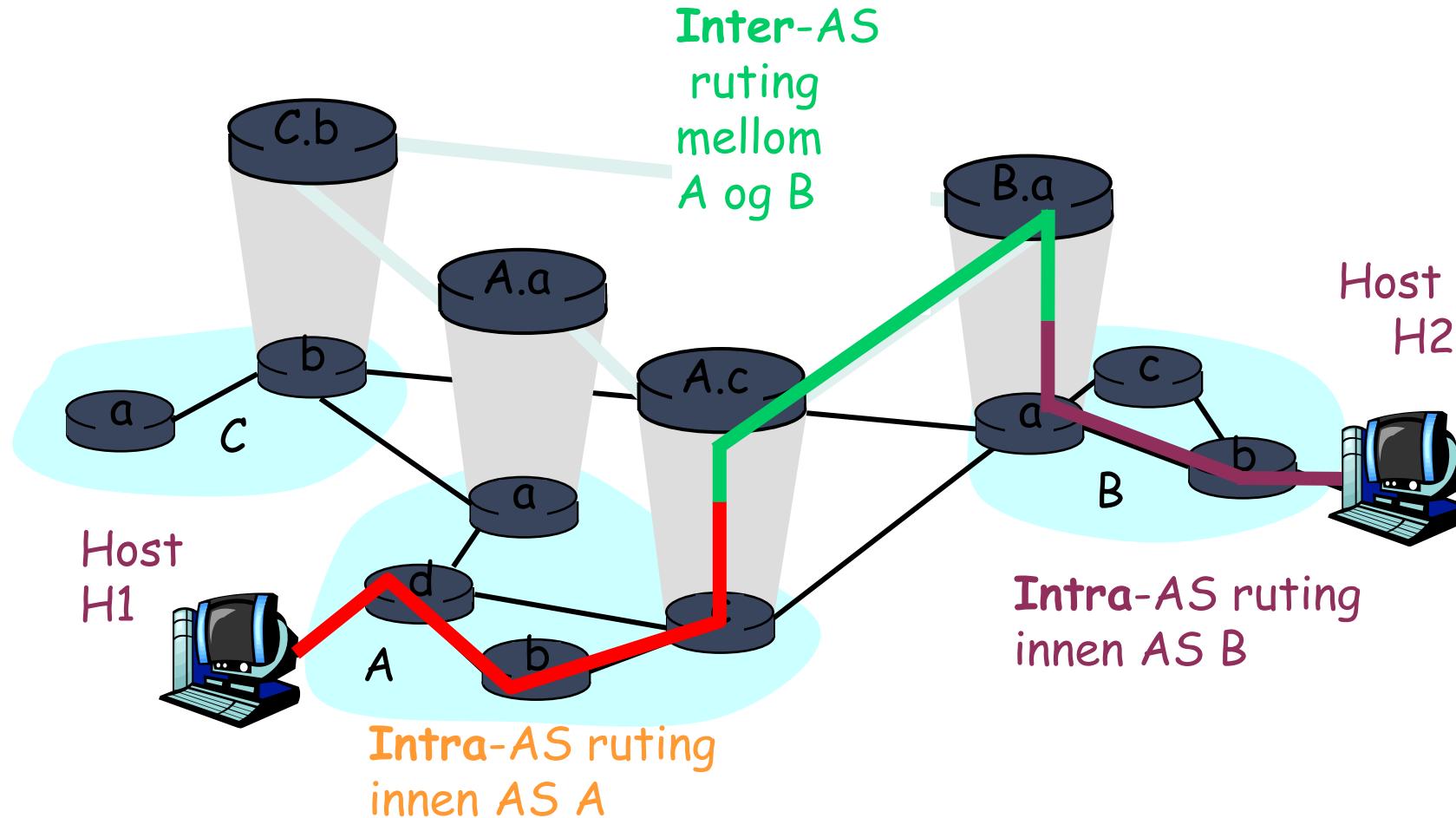


- Routing handler om å sette opp en **routingtabell** for hver enkelt router
- Tabellen angir hvilket **interface** et datagram skal **videresendes** ut av
- Alle host-maskiner har routingtabeller, men de er små
- Routere i stamnettet har tabeller med ~200000 linjer!
 - Hver linje er et IP-nettverk som routeren har funnet en **billigste** vei til

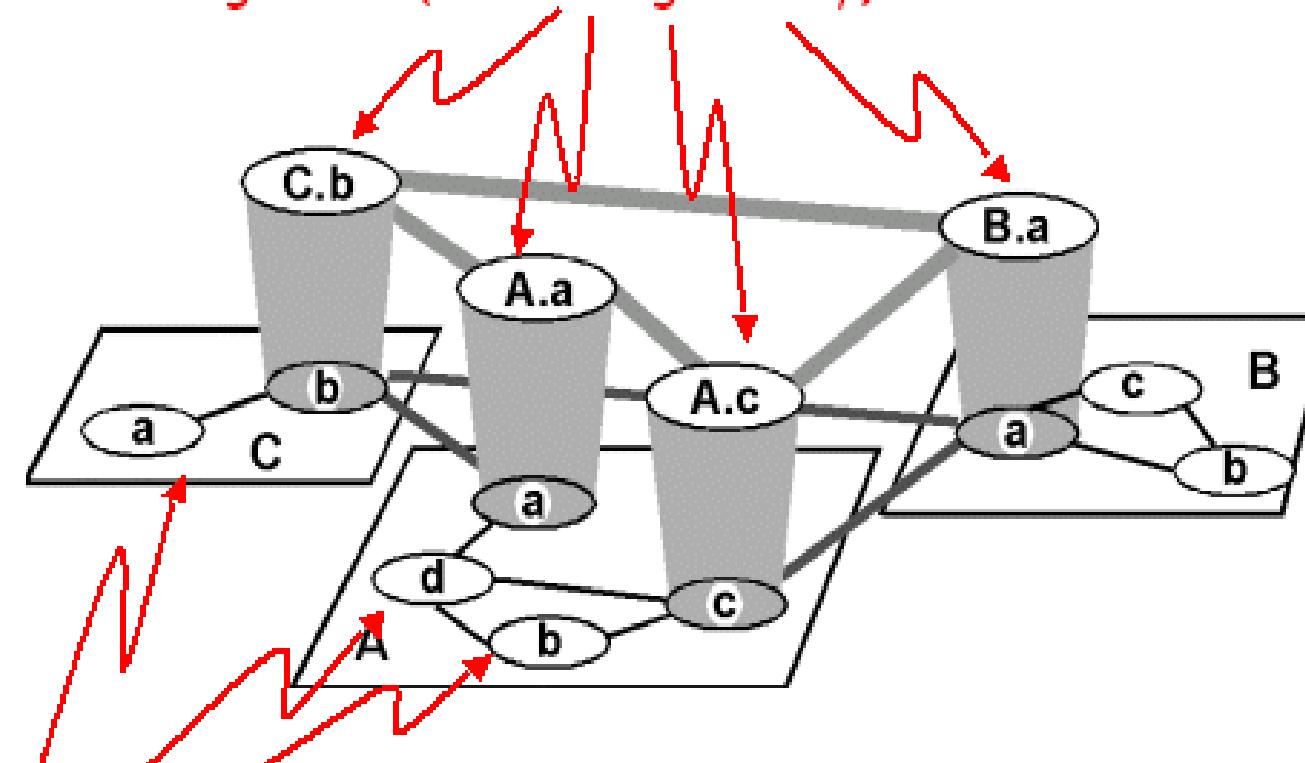
Hierarkisk ruting

- Internett er stort og komplekst
- Internett = nettverk av nettverk
- Deler routerene i **regioner**
 - **Autonome systemer** (AS)
- Routere i samme **AS** kjører samme routing-protokoll (**intra**-AS protokoll)
- Gateway utfører **inter**-AS routing mellom ulike AS
 - Bruker **BGP** som routingprotokoll

Intra-AS og Inter-AS ruting



Intra-AS grense (ekstern gateway) routere

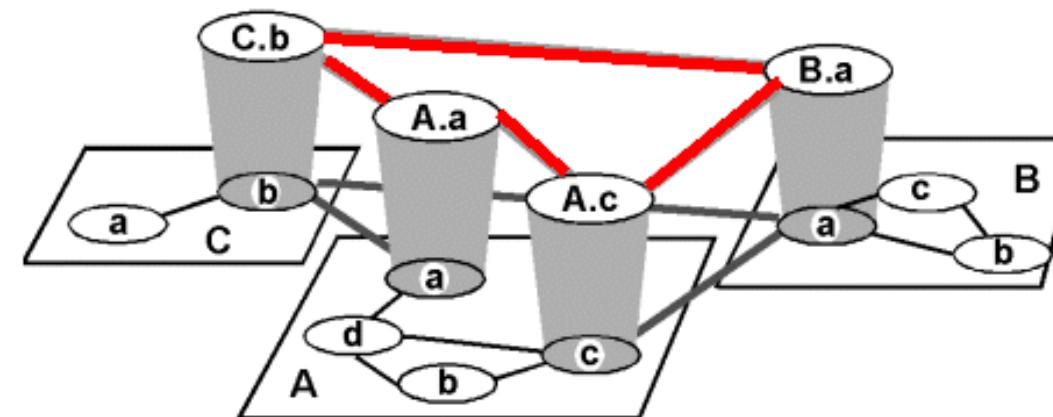


Inter-AS interne (gateway) routere

Intra-AS ruting

- Også kalt Interior Gateway Protocols (IGP)
- Vanligst forekommende er
 - Routing Information Protocol (RIP)
 - Open Shortest Path First (OSPF)
 - Interior Gateway Routing Protocol (IGRP)
 - Cisco proprietær
 - EIGRP videreutvikling av IGRP
- Implementert i programmer på routerene
 - Utveksler routing-informasjon med andre routere innenfor AS/WAN

- Border Gateway Protocol (**BGPv4**) er standarden på Internett
- Ruter **også** ut fra **AS-Nummer**
- Bruker Path Vector protokoll
 - Finner billigste vei ut fra «nabosladder»
 - Lagrer «AS-ruten» (path) til mottaker



BGP - Border Gateway Protokol

- Når gateway X sender sitt rute-forslag til node Z over til gateway Y, kan følgende skje:
 - Ruten godtas ikke (kostnad, politikk, pålitelighet...)
 - Ruten godtas og brukes deretter fra Y til Z
 - Y sender sin oppdaterte rute-kostnad til sine «andre naboer»
- Bruker TCP mellom routerne.
- 4 typer meldinger
 - OPEN: Åpner TCP forbindelse og bekrefter avsender
 - UPDATE: Åpner ny rute eller lukker en gammel
 - KEEPALIVE: Holder liv i forbindelsen uten UPDATE
 - NOTIFICATION: Sender feilmelding. Lukker forbindelse

- **Policy**

- Inter-AS: Ønsker kontroll over routingen mellom områdene; må ta hensyn til peering-avtaler og priser
- Intra-AS: Enhetlig kontroll av rutingen innen området; primært ute etter effektivitet og lastbalansering

- **Skala**

- Hierarkisk ruting sparer tabell-plass og reduserer oppdaterings-mengden

- **Ytelse**

- Inter-AS: Policy kan være viktigere enn ytelse
- Intra-AS: Fokuserer (oftest) på ytelse

Looking glass routere

- Noen «snille» firma lar deg se stamnett-routere fra «innsiden» f.eks. <http://lg.he.net>

core1.fra1.he.net> show ip bgp summary

Local AS Number		6939					
Number of Neighbors Configured		475, 460 up					
Number of Routes Installed		2512117 (216042062 bytes)					
Number of Routes Advertised							
Number of Attribute Entries		529657 (47669130 bytes)					
Neighbor Address	ASN	State	Time	Rt:Accepted	Rt:Filtered	Rt:Sent	Rt:ToSend
64.62.142.154	6939	ESTAB>	3d11h 1m	0	0	377561	1
64.212.32.45	3549	ESTAB	168d 7h 4m	368525	56468	672	0
80.81.192.3	21473	ESTAB	100d18h12m	41	0	33042	0
80.81.192.4	8804	ESTAB	43d 5h 6m	20	0	33042	0
80.81.192.6	8365	ESTAB	26d19h39m	20	0	33042	0
80.81.192.7	8767	ESTAB	68d 7h50m	172	21	33042	0
80.81.192.8	2857	ESTAB	100d22h16m	11	0	33042	0
80.81.192.9	8928	ESTAB	151d15h 0m	5821	185	33042	0
80.81.192.11	5413	ESTAB	100d16h47m	115	0	33042	0
80.81.192.12	6774	ESTAB	100d19h 4m	413	926	33042	0
80.81.192.13	8218	ESTAB	45d 7h32m	1371	0	21613	0
80.81.192.14	4589	ESTAB	142d11h24m	230	0	33042	0
80.81.192.15	12897	ESTAB	45d 6h20m	30	0	33042	0
80.81.192.16	3292	ESTAB	45d 4h 6m	1201	0	33042	0

Looking Glass
Welcome to Hurricane Electric's Network Looking Glass. The information provided by these are some of our routers at core locations within our network. We also operate server.he.net.

Routers

<input type="checkbox"/> NIKHEF Amsterdam	<input type="checkbox"/> Intexion Frankfurt 1	<input type="checkbox"/> Telehouse Docklands	<input type="checkbox"/> Intexion Paris 2	<input type="checkbox"/> Telehouse Voltaire Paris	<input type="checkbox"/> Teletel Stockholm	<input type="checkbox"/> Equinix Zürich
NL 	DE 	UK 	FR 	FR 	SE 	CH 
<input type="checkbox"/> Equinix Ashburn	<input type="checkbox"/> Telx Atlanta					
US 	US 					

Commands

Ping
 Traceroute
 BGP Route
 BGP Summary (IPv4)
 BGP Summary (IPv6)

Arguments

IP:
 Raw output (no tables)

Buttons

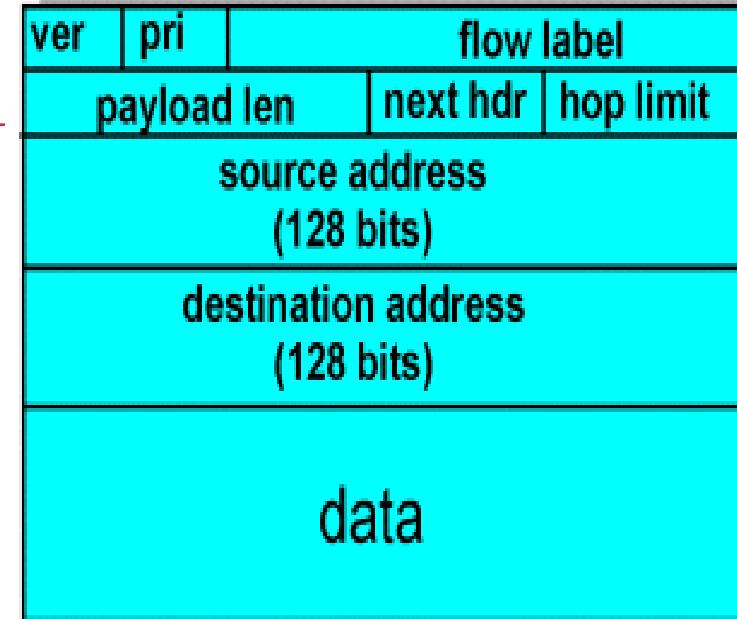
Probe

I nternet P rotocol v6

- Den kolossale veksten av Internett krever stadig flere adresser
- 3. Feb. 2011 ble de siste IPv4 adresse-blokkene tildelt Verdensdel-registratene!
- IPv6 øker adressefeltet **fra 32 til 128 bit**
- **Forenkler** header-format
 - Fjerner mulighet for fragmentering
 - Fjerner sjekksummen
- Forenkler adressering og data-flyt
- Fast lengde på header = 40 byte

IPv6 header

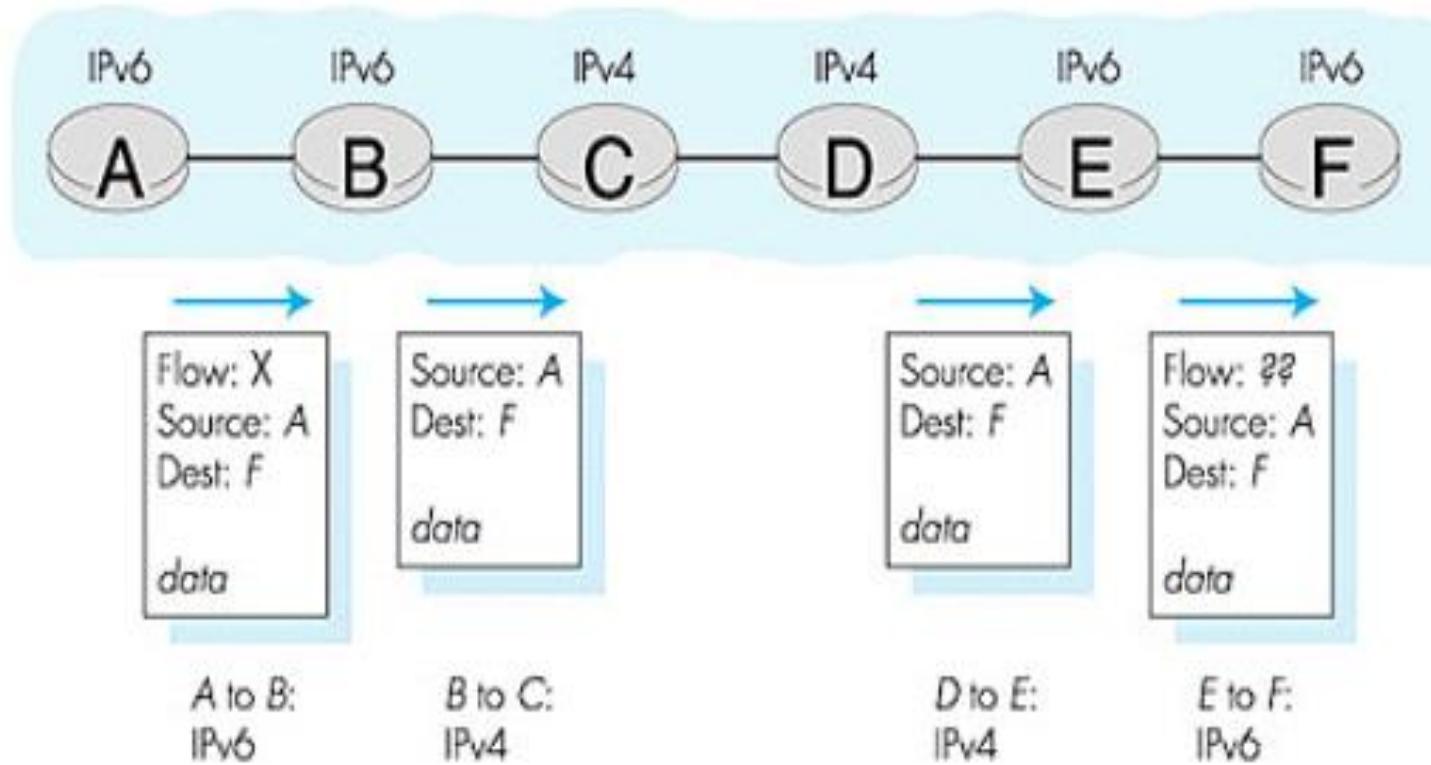
- Version: 0110
- Traffic Class
 - Prioritering innad i en datastrøm
- Flow Label
 - QoS
 - Sikre diffrensiering i tjenestekvalitet
 - Noe uklart definert – variabel routerstøtte
- Payload length
 - Antall byte med nyttelast
- Next Header
 - Protokoll på nivået over i stacken (UDP, TCP, ...?)
 - Kan/vil også være Header-utvidelser slik som **IPSec**
- Hop Limit
 - Tilsvarer TTL slik det ble praktisert IPv4
- DATA



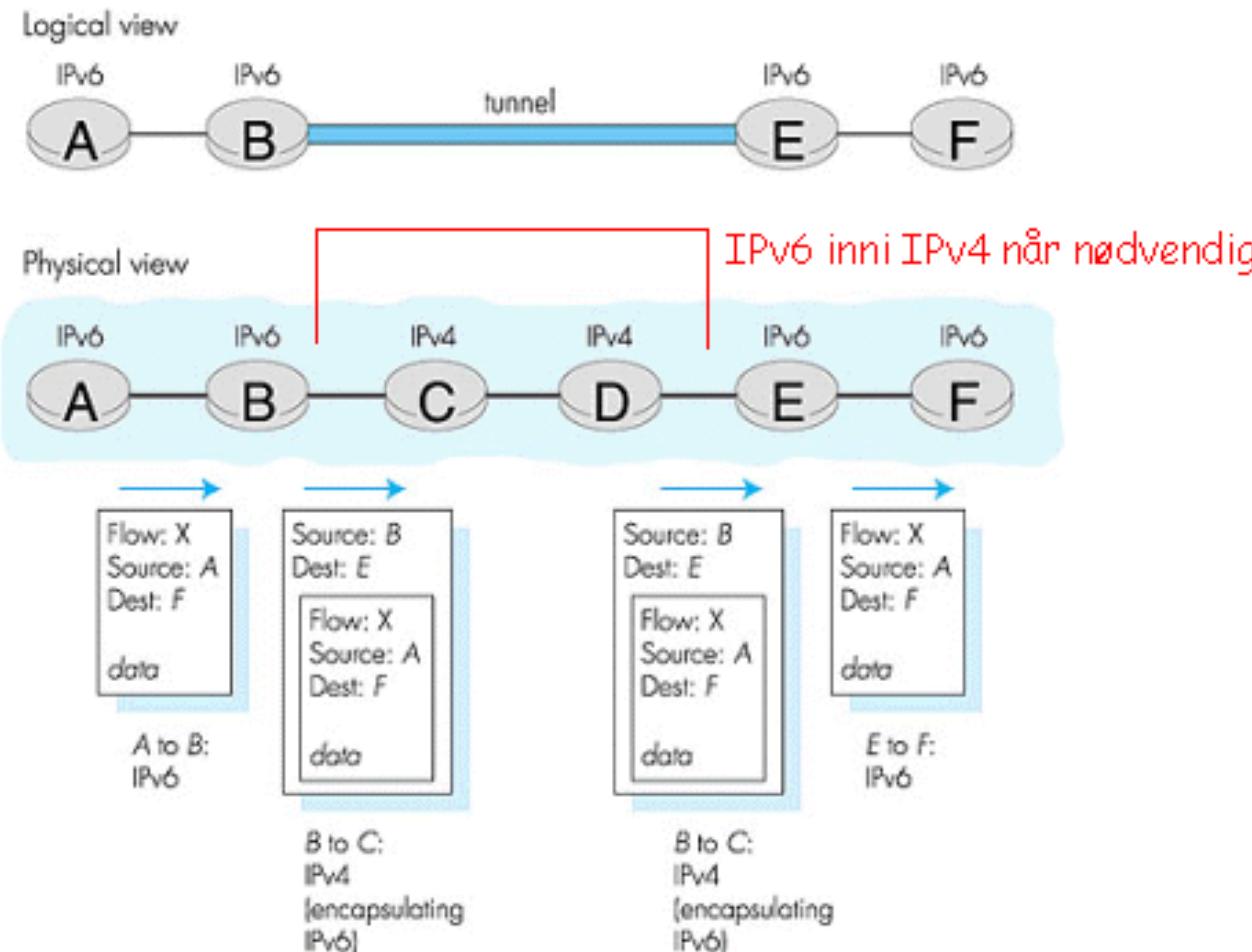
Overgang fra IPv4 til IPv6

- Umulig å oppgradere alle routere samtidig
- Overgangsperiode hvor begge typer routere må kunne operere sammen
- To (tre) løsninger er i bruk
 - Dual stack
 - Routeren forstår begge protokoller og oversetter mellom disse
 - NAT64/DNS64, (SLAAC)
 - Tunneling
 - En IPv4 router behandler IPv6 datagram som data og slipper datagrammet urørt igjennom (innpakket i IPv4)
 - 6in4, 6rd, Teredo, ISATAP

Dual stack



Tunneling



- 128 bit (16 Byte) blir skrevet hexadesimalt i 8 grupper på 2 byte
 - IPCONFIG /all gir f.eks.:

Ethernet-kort eth0:

on

```
Tilkoblingsspesifikt DNS-suffiks : oslo.nith.no
Beskrivelse . . . . . : Intel(R) PRO/1000 PL Network Connecti
Fysisk adresse . . . . . : 00-0E-7B-98-F8-A1
DHCP aktivert . . . . . : Ja
Automatisk konfigurasjon aktivert : Ja
IP-adresse . . . . . : 10.21.11.173
Nettverksmaske . . . . . : 255.255.255.0
IP-adresse . . . . . : fe80::20e:7bff:fe98:f8a1%5
Standard gateway . . . . . : 10.21.11.1
DHCP-server . . . . . : 10.21.11.20
DNS-servere . . . . . : 10.21.4.131
                                         10.21.21.101
```

- %5 er Win-adaptternr. (ikke egentlig del av standarden)
 - fe80:: = fe80:0000:0000:0000 = nettprefix
 - :: er minimum fire nuller, her 12 ut fra resten av adressen
 - 02**0e:7b**ff:fe**98:f8a1** er basert på MAC-adressen

IPv6 Notasjon

- Bruker CIDR for å angi nettverksprefix
- Noen spesielle adresser
 - ::/128 tilsvarer 0.0.0.0 og brukes ikke annet enn internt på node
 - ::1 tilsvarer 127.0.0.1 (localhost)
- 2000::/3 er Global Unicast (~»Vanlig Internett»)
 - 2001::/32 tildeles ISPer som typisk deler ut /48 og /64 nett til kunder
 - 2002::/16 vil bli "gamle internett" (IANA)
 - 6to4 -- Allokert til RIPE o.l.
- 2001:**db8**::/32 benyttes i dokumentasjon
 - Merk: innledende null skrives ikke
- **fe80**::/64 er en local link adresse. Tilsvarer på mange måter 169.254.X.X adresser (tas dersom IPv6-kapabel router ikke er tilgjengelig)
- ::ffff:/96 benyttes på IPv4-overgangsadresser, som får formatet: ::ffff:192.0.2.114

- Broadcast videreføres ikke fra v4
 - Bruker **multicast** i stedet.
- Tre typer adresser
 - **Unicast**
 - Enkeltadresse
 - **Anycast**
 - Første som tar mot
 - Bestemmes av router
 - **Multicast**
 - Til en forhåndsdefinert gruppe (site)
 - Format: FF00::/8 (F.eks. FF02::1 alle noder på samme link)
 - Bruker MLD (Multicast Listener Discovery) og ND (Neighbour Discovery) protokollene.

- ARP skal bort
- DHCP trengs ikke lenger
 - men **DHCPv6** kan brukes til å dele ut DNS-server og i administrerte nettverk (LAN/WAN)
 - Erstattes av IPv6 meldingsutveksling for å finne gateway som tildeler scope:globale adresse basert på MAC-adresse
- Ny ICMPv6
 - Kan «erstatte» DHCP-oppsett av router o.l.
 - Mange nye typer meldinger.
- Nye versjoner av RIP, OSPF og andre routing-protokoller.

- **StateLess Address AutoConfiguration**
 - Skal automatisk sette opp IPv6 nettet for deg.
- Typisk for bruk i lokal- og hjemme-nettverk med IPv6-kapabel **router** og **ISP som tilbyr IPv6**
- Bruker ICMPv6 til å finne router
 - får tildelt IPv6 adresse og andre parametere av routeren

Ipconfig /all (ifconfig -l) «forteller»

Wireless LAN adapter WLANUSB:

```
Connection-specific DNS Suffix . : ad.nith.no
Description . . . . . : D-Link DWA-140 Wireless N USB Adapter(rev.B3)
Physical Address. . . . . : B8-A3-86-90-50-E8
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::50e5:40ff:6794:1d5a%16(Preferred)
IPv4 Address. . . . . : 10.21.26.56(Preferred)
Subnet Mask . . . . . : 255.255.252.0
Lease Obtained. . . . . : 19. november 2013 14:29:14
Lease Expires . . . . . : 19. november 2013 22:29:14
Default Gateway . . . . . : 10.21.24.1
DHCP Server . . . . . : 1.1.1.1
DHCPv6 IAID . . . . . : 548971398
DHCPv6 Client DUID. . . . . : 00-01-00-01-14-6A-F2-0B-D8-D3-85-77-A0-3F
DNS Servers . . . . . : 158.36.131.10
Primary WINS Server . . . . . : 158.36.131.10
NetBIOS over Tcpip. . . . . : Enabled
```

Tunnel adapter Teredo Tunneling Pseudo-Interface:

```
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Description . . . . . : Teredo Tunneling Pseudo-Interface
Physical Address. . . . . : 00-00-00-00-00-00-E0
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . . : Yes
```

IPv6 i virkeligheten

- I Windows, OSX og Linux (kernel 2.6.x) er IPv6 en del av "standardpakken"
- I Vista, Win7&8 er IPv6 aktivert pr default, OSX (Free BSD) også aktivert som "default" siden 10.4 (?)
- I realiteten må man basere seg på tunneling gjennom IPv4 og en god del håndarbeid.
 - Det kommer (kanskje) ikke til å vare (veldig..) lenge...
 - uPnP er gira mot IPv6 f.eks.

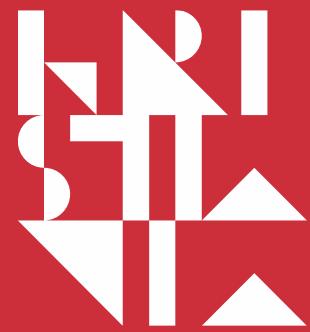
IPv6 problemer pr d.d.

- Tunneling ved [6in4](#) viser seg å stoppes av mange brannmurer fordi den legger en ny versjonskode (41) inn i IPv4-headeren
- Bare et fåtall [nettverksadministratorer](#) kan særlig mye om IPv6.
- Mange [ISPer](#) har ikke gjort noen forberedelser i det hele tatt...
- Kun få år siden [DNS AAAA-records](#) ble tilgjengelige i root-servere
- Mange [applikasjoner](#) klarer ikke å håndtere annet enn IPv4 sockets
- +++

Denne forelesningsøkten vil bli tatt opp og lagt ut i emnet i etterkant.

Hvis du ikke vil være med på opptaket:

 ^ Start Video	La være å delta med webkameraet ditt.
 ^ Unmute	La være å delta med mikrofonen din.
To: Marianne Sundby (Privately) Type message here...	Still spørsmål i Chat i stedet for som lyd. Hvis du ønsker kan spørsmålet også sendes privat til foreleser.



Høyskolen
Kristiania

TK1100

0xA forelesning:

Linklaget

Linklaget (“Datalinjelaget”)

Mål:

- **forstå prinsippene** bak linklagstjenester:
 - feildeteksjon og feilretting
 - deling av en kringkastingskanal: multippel aksess
 - linklagsadressering
 - pålitelig dataoverføring: *gjort!* – se TCP
 - flytkontroll: *gjort!* – se TCP
- ulike linklagsteknologier
 - Konsentrerer oss om **Ethernet** fordi dette er det vi treffer på i hverdagen

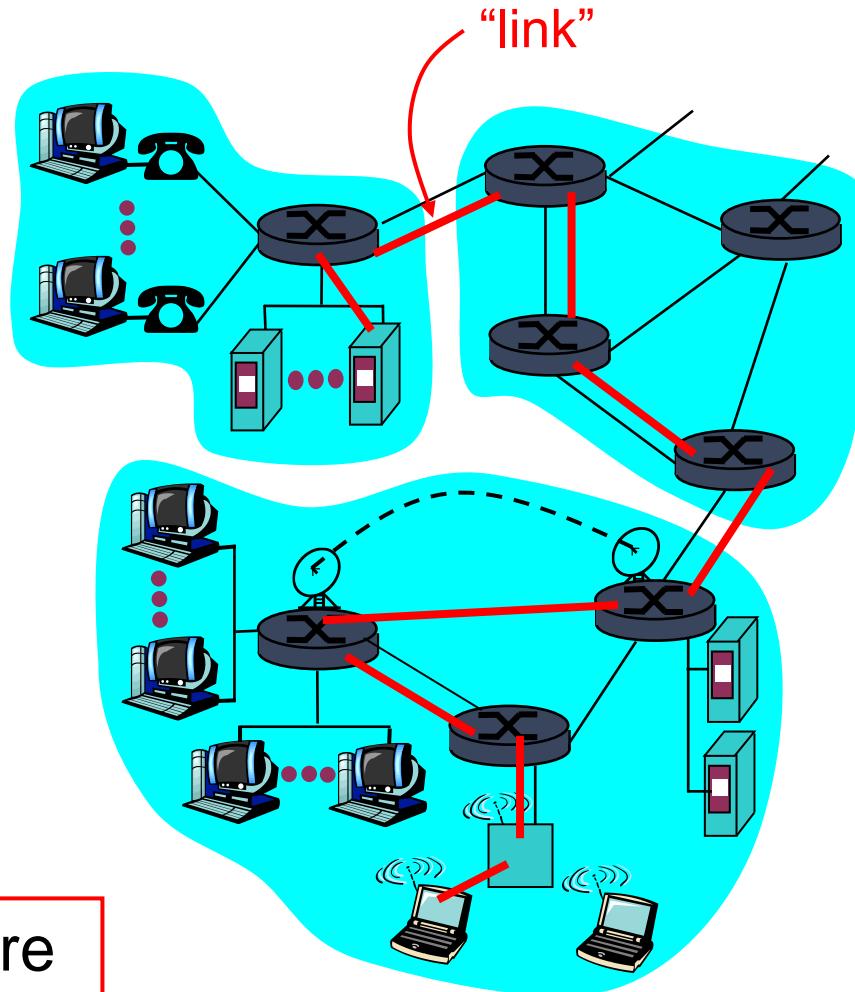
Navn på laget	Betegnelse på overføringsenhet	Viktigste oppgaver/funksjoner Exempel på protokoller/standarder
Applikasjonslaget	Melding (Message)	Støtte nettverksapplikasjoner Ex: HTTP, DNS, FTP, SMTP, POP3....
Transportlaget	Segment	Transport av applikasjonslagsmeldinger mellom klient- og tjener-sidene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

Linklaget / Datalinklaget / Datalinjelaget

Linklaget: Introduksjon

Litt terminologi:

- maskiner, rutere og svitsjer er **noder**
- kommunikasjonskanaler som forbinder nabonoder langs kommunikasjonsveien er **linker**
 - trådbundne (kablede) linker
 - trådløse (radio) linker
 - LAN
- En PDU kalles en **ramme (frame)**, innkapsler datagrammer



linklaget har ansvar for å overføre datagrammer fra en node til en **nabonode** over en link

Linklaget: sammenheng

- Datagram overføres av **ulike linkprotokoller** over ulike linker:
 - f eks
 - Ethernet på første link
 - Frame Relay på neste link
 - FDDI (fiber) på neste link
 - ...
 - 802.11 på siste link
 - Hver linklagsprotokoll tilbyr ulike tjenester
 - f eks: én protokoll kan være pålitelig, en annen upålitelig
- transportanalogi
- tur fra Halden til Trondheim
 - tog: Halden til Gardermoen
 - fly: Gardermoen til Værnes
 - buss: Værnes til Trondheim
 - reisende = **datagram**
 - **transportetappe** = **kommunikasjonslink**
 - **transporttype** = **linklagsprotokoll**
 - **reisebyrå** = **routingalgoritme**

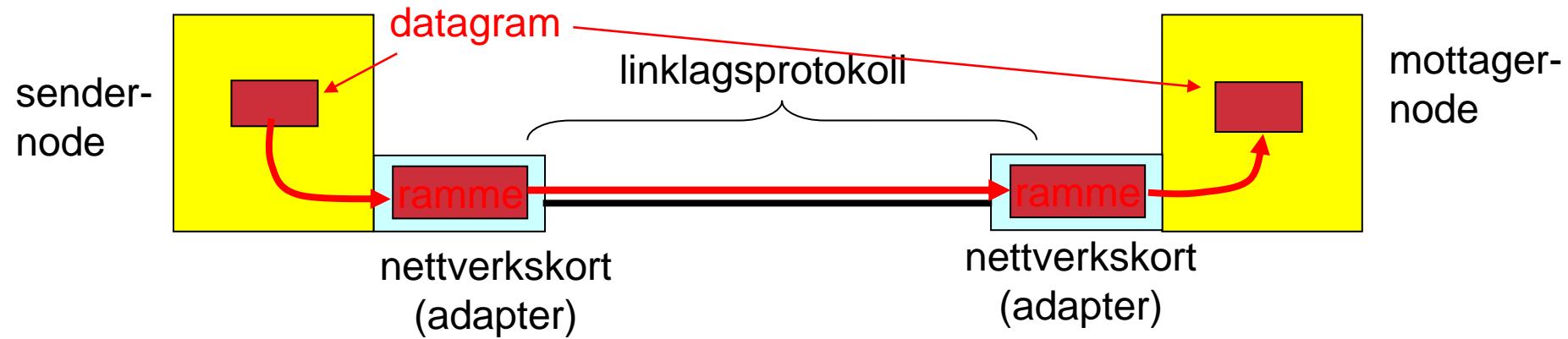
Linklagstjenester (1)

- Omramming (framing) og link-aksess:
 - innkapsling av datagram i rammer, legger til header og trailer
 - kanaltilgang hvis delt medium (MAC = medium access control)
 - **MAC-adresser** benyttes i rammeheader for å identifisere avsender og mottager
 - forskjellig fra IP-adresser!
- **Pålitelig leveranse mellom nabonoder**
 - vi har alt sett på hvordan dette kan gjøres (Forelesning 08)!
 - lite nødvendig på link med lav bitfeilrate (fiber og noen typer kobberkabel)
 - trådløse linker: høy bitfeilrate

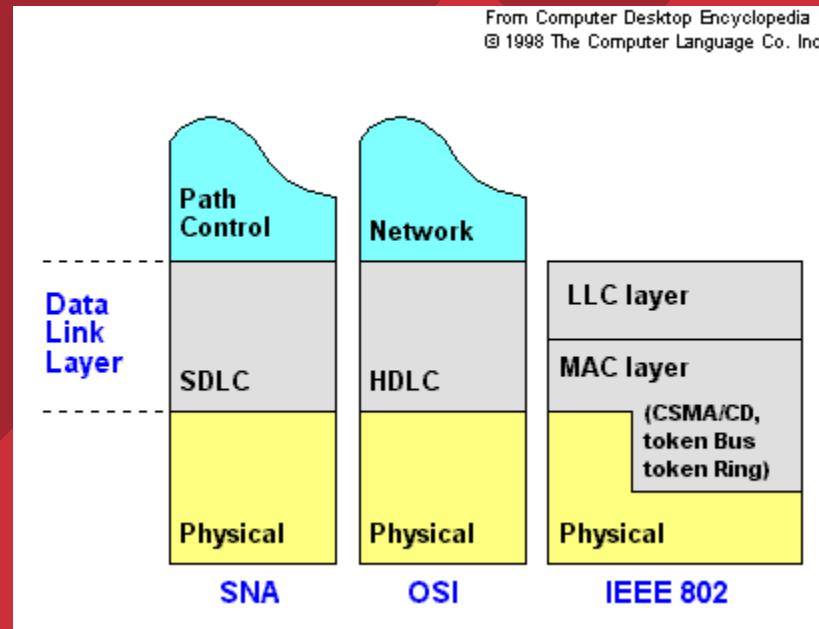
Linklagstjenester (2)

- **Flytkontroll:**
 - sender må ikke sende fortore enn mottager kan ta imot
- **Feildeteksjon:**
 - feil forårsakes av dempning og elektrisk støy
 - mottager oppdager bitfeil:
 - gir enten beskjed til sender om å retransmittere
 - eller den bare kaster rammen
- **Feilretting:**
 - mottager identifiserer *og retter* bitfeil uten at sender må retransmittere
 - (Finnes i 802.11n)
- **Halv-duplex og full-duplex**
 - med halv-duplex kan noder i begge ender av linken sende, men ikke begge samtidig
 - med full-duplex kan begge sende og motta samtidig

Nettverkskort kommuniserer



- linklaget implementert i nettverkskort (NIC)
 - Ethernet-kort, 802.11-kort e.l.
- senderside:
 - innkapsling av datagram i en ramme
 - adderer bit for deteksjon av bitfeil, (sekvensnummer, flytkontroll etc.)
- mottagerside
 - ser etter bitfeil, re-transmisjon, flytkontroll etc.
 - ekstraherer datagram, leverer dette til mottagernode
- NIC er delvis autonomt
- Moderne nettverkskort støtter ofte også transport- og nettverkslagsfunksjonalitet



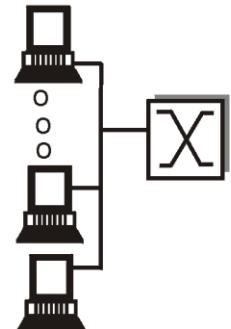
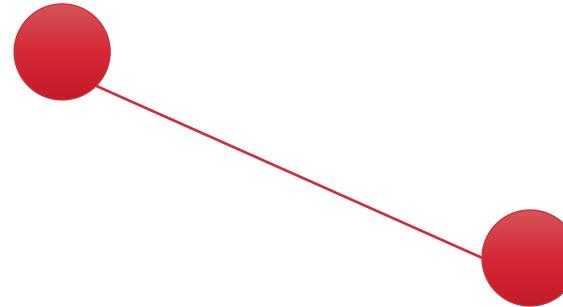
Medium Access Control



Linker og protokoller for multipel aksess

To hovedtyper “linker”:

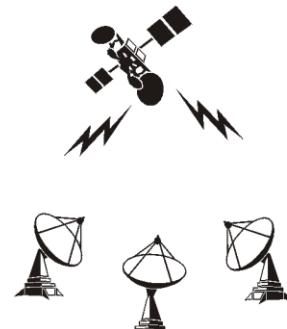
- **punkt-til-punkt**
 - PPP for opprinnigt aksess
 - punkt-til-punkt-link mellom to routere
- **kringkasting (delt medium)**
 - tradisjonelt ethernet
 - 802.11 trådløst lokalnett



shared wire
(e.g. Ethernet)



shared wireless
(e.g. Wavelan)



satellite



cocktail party

Multippel aksess-protokoller (MA)

- én delt kanal som alle kan lytte på (kringkasting)
- to eller flere samtidige transmisjoner fra noder resulterer i interferens
 - bare én må sende om gangen

multippel-aksessprotokoll

- distribuert algoritme som bestemmer hvordan noder deler en kanal, dvs at den bestemmer når en node kan sende
- kommunikasjon om kanaldeling må selv bruke kanalen

MAC-protokoller: taksonomi

Tre ulike tilnærminger til kanaldeling:

- **Kanalpartisjonering**
 - deler kanalen i mindre biter
 - Tidsluker
 - Frekvenser
 - koder
 - tildeler del av kanalen eksklusivt til en node
- **Random access**
 - kanal deles ikke opp, kollisjoner kan forekomme
 - har metoder for å håndtere kollisjoner
 - Ethernet (IEEE 802.3)
- **“Etter tur”**
 - unngår kollisjoner ved at kun en om gangen får sende
 - “Token Ring” (IEEE 802.5)
 - WiFi (IEEE 802.11 med Access Point)

Random access protokoller

- Når en node har pakker å sende
 - Sender med full kanalrate, R
 - ingen *a priori* koordinering mellom nodene
- to eller flere sender samtidig → “kollisjon”,
- **random access MAC protokoll** spesifiserer:
 - hvordan detekteres kollisjoner
 - hvordan håndtere kollisjoner (f eks ved hjelp av forsinkede retransmisjoner)
- Eksempler på random access MAC protokoller:
 - slotted ALOHA
 - ALOHA
 - CSMA, CSMA/CD, CSMA/CA

CSMA (Carrier Sense Multiple Access)

CSMA: lytter før sending:

- Hvis kanalen er ledig: send rammen
- Hvis kanalen er opptatt: utsett transmisjonen
- Menneskelig analogi: før man begynner å snakke, hører man etter om andre snakker – ikke avbryt!

CSMA kollisjoner

kollisjoner kan forekomme:

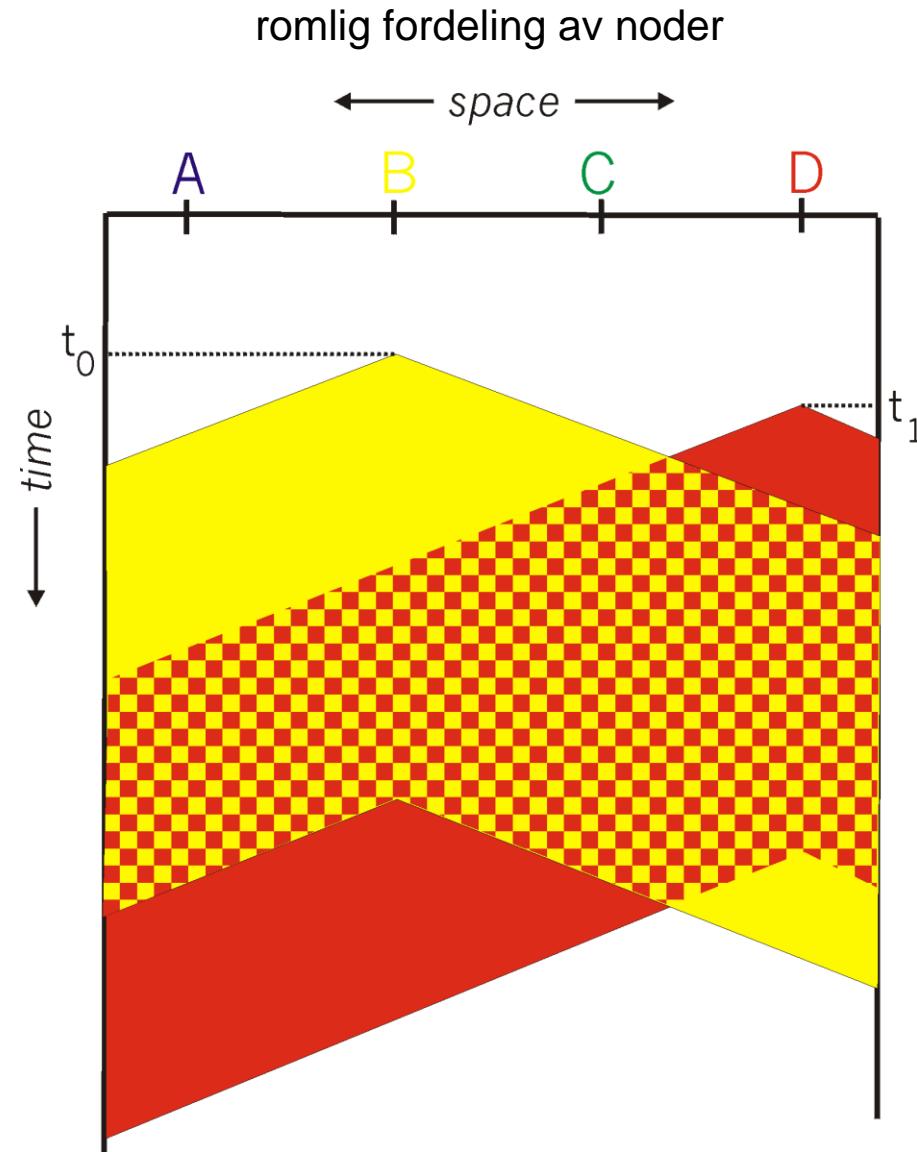
avstanden kan gjøre at en node ikke hører at en annen har startet sending

kollisjon:

hele sendetiden bortkastet

merk:

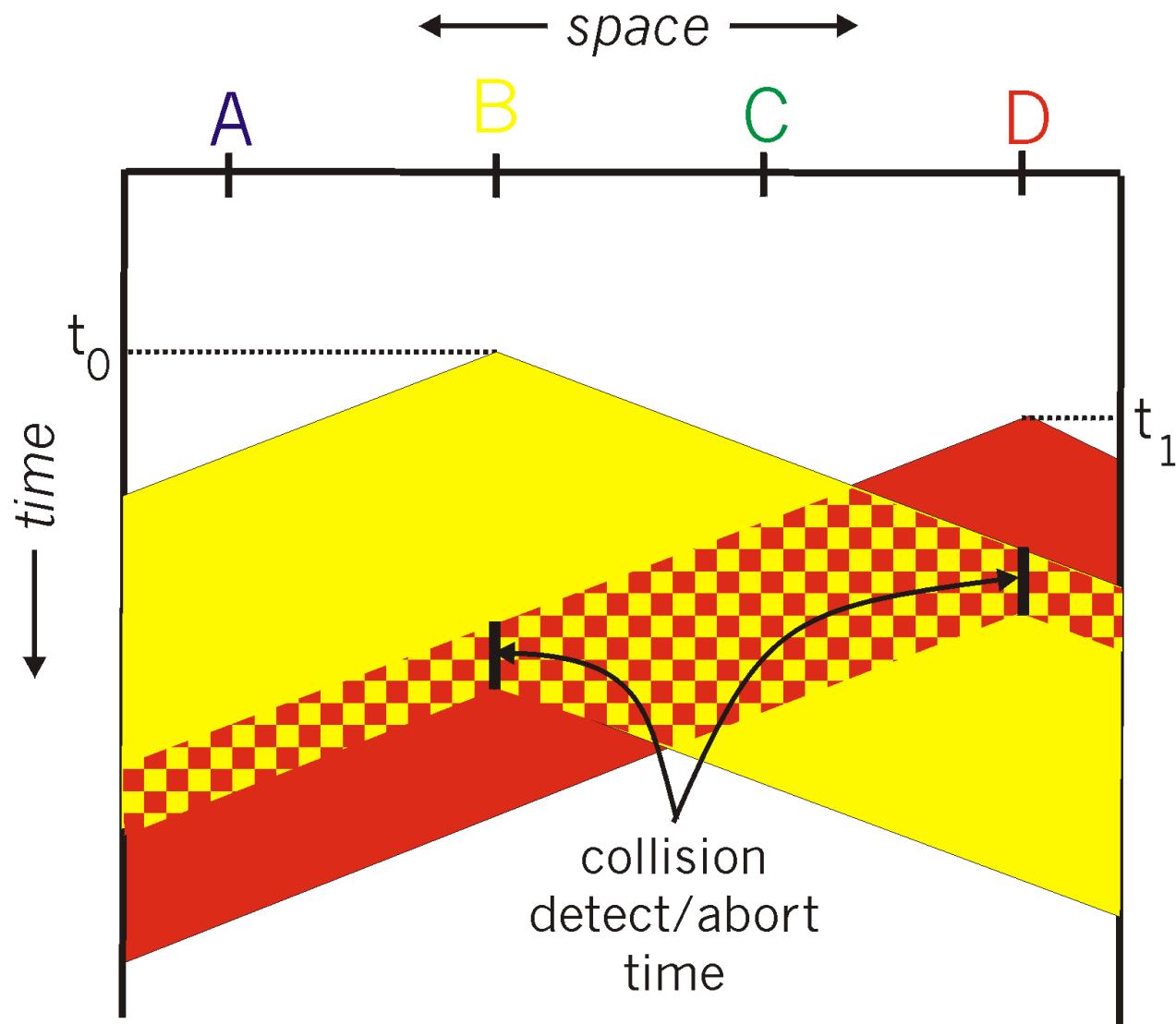
avstander og gangtid har betydning for sannsynligheten for kollisjoner



CSMA/CD: lytter på mediet før sending, venter hvis mediet er opptatt (som i CSMA)

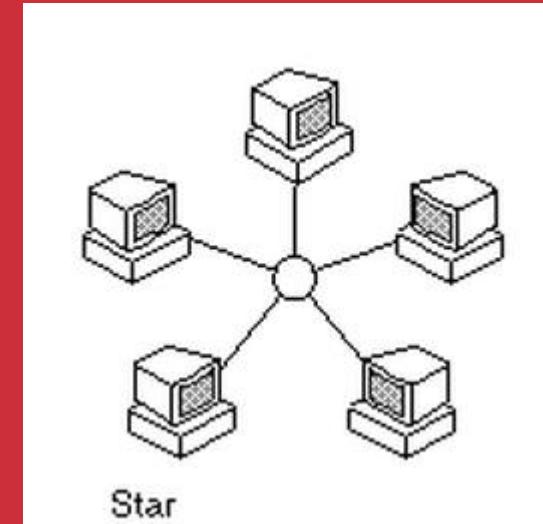
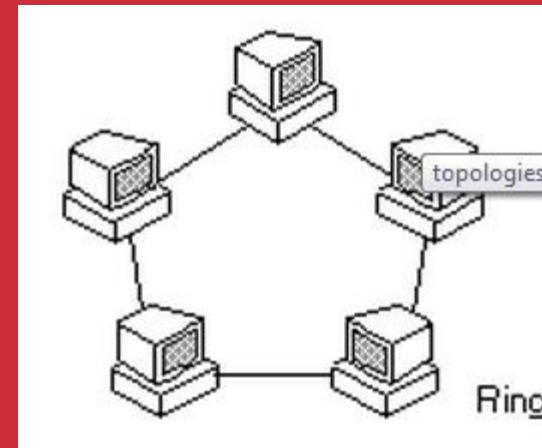
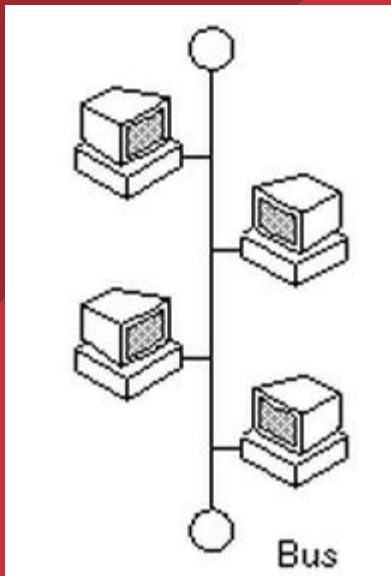
- fortsetter å lytte mens man sender: kollisjoner *detektert* i løpet av kort tid
- ved kollisjon avbrytes sendingen umiddelbart → reduserer sløsing med tid
- collision detection:
 - enkelt i kablede lokalnett: måler signalstyrken, sammenligner sendt og mottatt signal
 - vanskelig i trådløse lokalnett: mottager er vanligvis slått av mens man sender
- menneskelig analogi: den høflige samtalepartner

CSMA/CD kollisjonsdeteksjon



Ex: Konsekvenser av CSMA

- Delt medium medfører deling av tilgjengelig kapasitet
- F.ex: De trådløse Access Pointene på skolen har en max kapasitet på 300Mbps
 - medfører max ca 150 Mps for nedlasting
 - Max antall brukere er 30 => Max 5 Mps pr bruker, men lavere pga pakketap og fordi administrativt overhead for AP øker ved flere brukere.



Local Area Network

Linklaget til nå:

- tjenester, feildeteksjon/-korreksjon, multippel aksess

Neste: LAN-teknologier

- adressering
- Ethernet
- hub, switch
- PPP

LAN-adresser (= MAC-adresser)

32-bit IP-adresse:

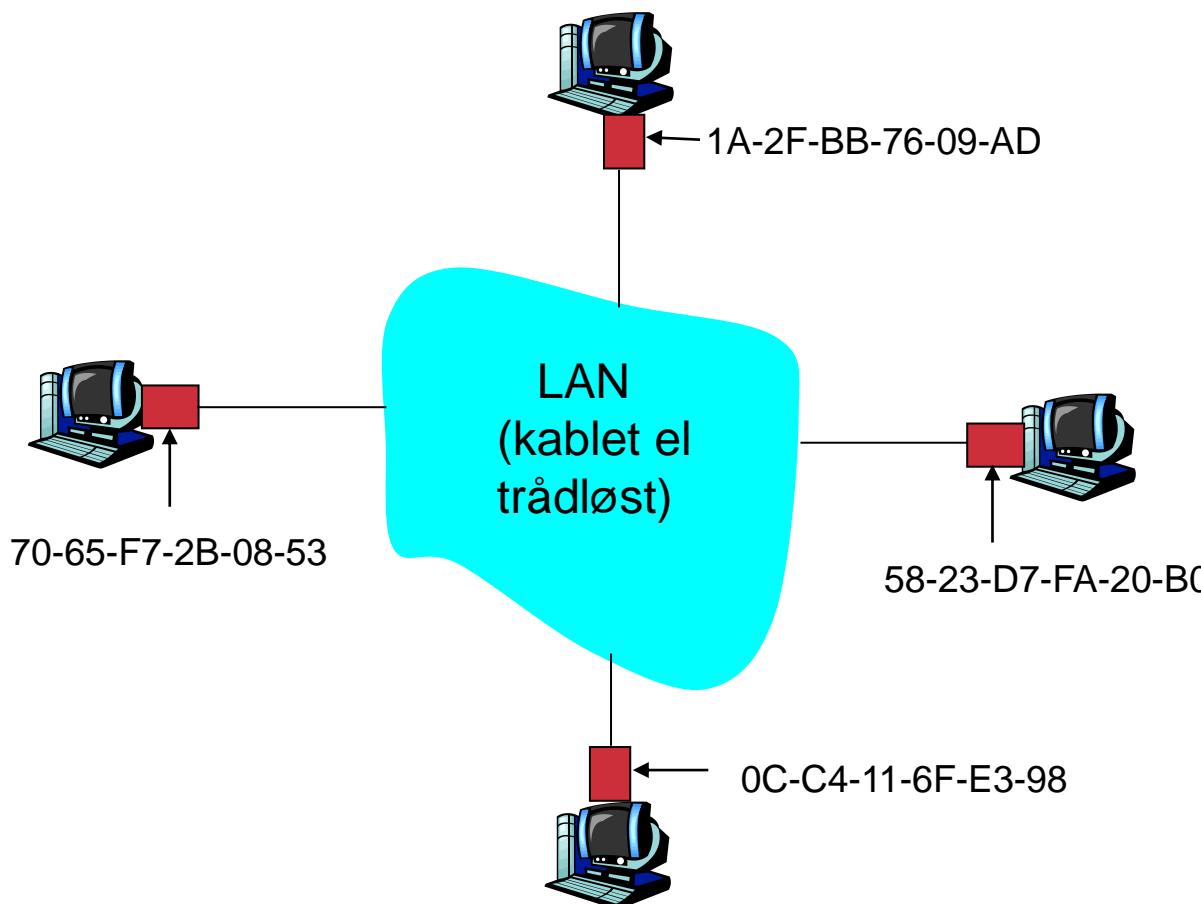
- *nettlags*-adresse
- benyttes for å få datagrammet fra ditt IP-nett fram til mottagerens IP-nett

MAC- (eller “LAN-” eller “fysisk-” eller “ethernet-”) adresse:

- benyttes for å få levert en ramme fra et interface til et annet interface **på samme nettet**
- **48 bit** MAC-adresse (for de fleste lokalnett) brent i nettverkskortets ROM

MAC-adresser

Hvert nettverkskort har en “unik” MAC-adresse



Broadcast-adressen er
FF-FF-FF-FF-FF-FF

Multicast-adresse har en
avsluttende 1'er i første
byte av MAC-adressen, og
behandles vanligvis som
broadcast, f.ex.
A3-05-77-31-27-BA

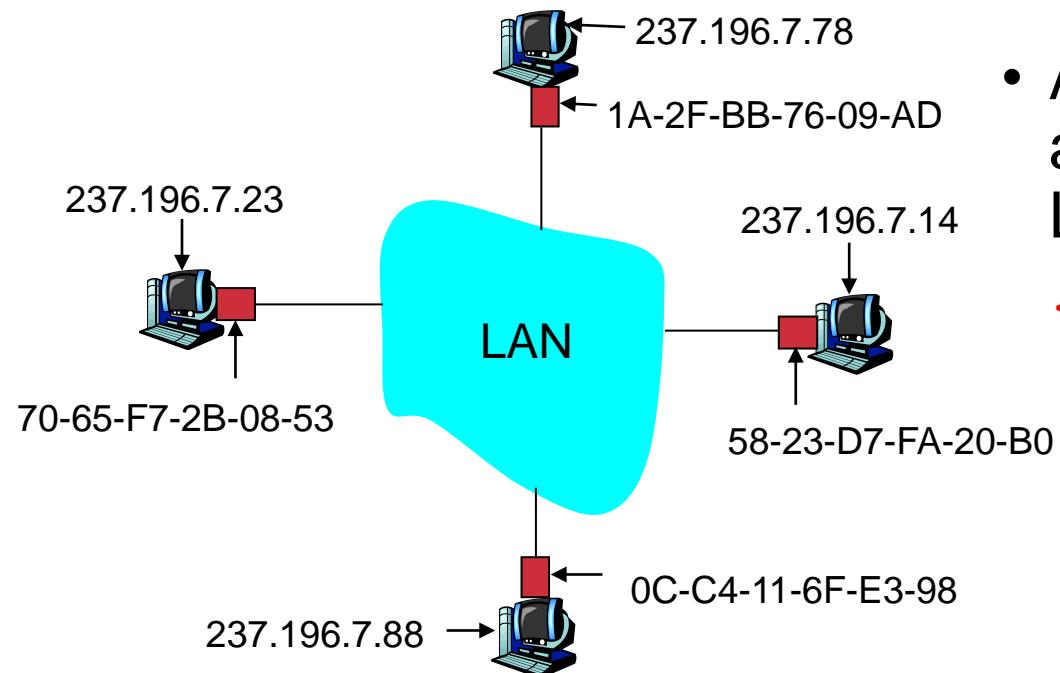
- Tildeling av MAC-adresser administreres av IEEE
- produsent kjøper del av MAC-adresserommet
 - Første **24 bit** forteller hvem som er **produsent** av nettverks-kortet

```
Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
Ethernet II, Src: AewinTec_0a:ab:c8 (00:0d:48:0a:ab:c8), Dst: Hewlett-77:a0:3f (d8:d3:85:77:a0:3f)
  Destination: Hewlett-77:a0:3f (d8:d3:85:77:a0:3f)
  Source: 158.36.131.1 (00:0d:48:0a:ab:c8)
  Type: IP (0x0800)
Internet Protocol Version 4, Src: 162.159.241.165 (162.159.241.165), Dst: 158.36.131.51 (158.36.131.51)
Transmission Control Protocol, Src Port: https (443), Dst Port: 55862 (55862), Seq: 2037027459, Ack: 26020
Secure Sockets Layer
```

- Analogi:
 - (a) MAC-adresse: personnummer
 - (b) IP-adresse: postadresse
- MAC har “flat” adressestruktur → portabilitet
 - kan uproblematisk flytte et nettverkskort fra et lokalnett til et annet, forutsatt at det ikke er satt opp filtrering av MAC-adresser i AP eller switch
- IP-adresser er hierarkiske og derfor ikke portable
 - Prefix-delen av IP-adressen angir hvilket nett maskinen henger på

ARP: Address Resolution Protocol

Hvordan finne MAC-adressen til en node man kjenner IP-adressen til?



- Hver IP-node (maskin og ruter) på et LAN har en **ARP-tabell/cache**
- ARP-tabell: IP/MAC adressemappinger for noen LAN-noder
 - <IP-adresse; MAC-adresse; TTL>
 - TTL (Time To Live): tiden mappingen skal ligge i ARP-tabellen (typisk 20 min)

ARP (Address Resolution Protocol)

- A ønsker å sende et datagram til B og kjenner Bs IP-adresse
 - Anta at Bs MAC-adresse ikke er i As ARP-tabell
- A **kringkaster** en ARP forespørsel som inneholder Bs IP-adresse
 - alle maskiner på LAN mottar ARP-forespørselen
- B mottar også ARP-pakken og svarer A med sin MAC-adresse
 - ramme sendes direkte til As MAC-adresse
- A cacher (lagrer) IP-til-MAC adresseparet i sin ARP-tabell inntil informasjonen blir foreldet
 - “soft state”: informasjon som forsvinner dersom den ikke oppfriskes
- ARP er “plug-and-play”:
 - en node lager sin ARP-tabell uten hjelp fra noen

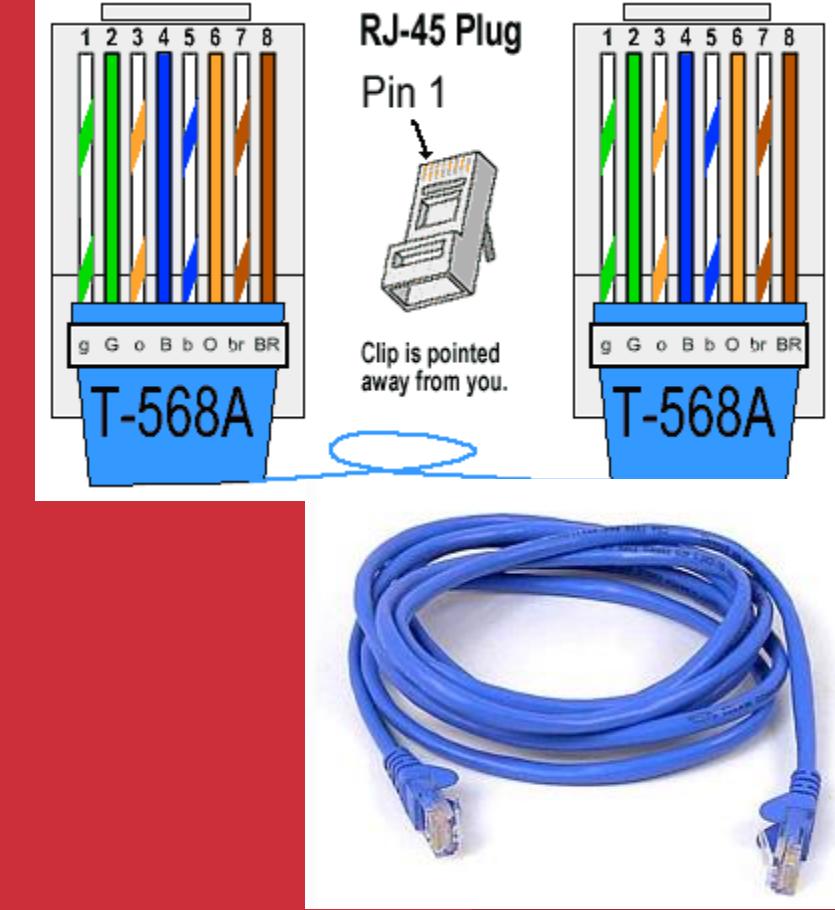
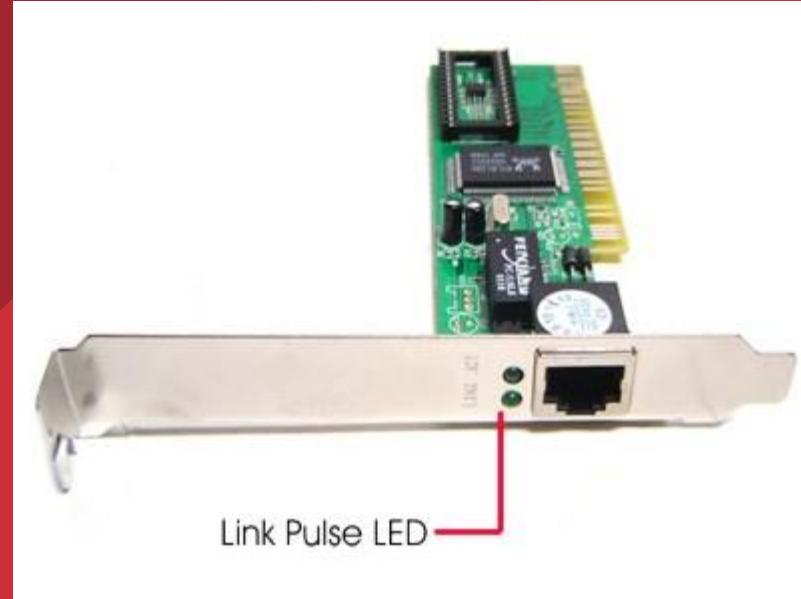
- [arp](#)-kommandoen kan brukes til å
 - vise frem ARP-cache ([-a](#))
 - tømme ARP-cache ([-d](#))
 - Legge til faste IP-MAC koplinger ([-s](#))

```
C:\>arp -a
Interface: 158.36.131.51 --- 0xc
  Internet Address          Physical Address          Type
  158.36.131.1              00-22-55-3e-da-ba  dynamic
  158.36.131.127            ff-ff-ff-ff-ff-ff  static
  255.255.255.255          ff-ff-ff-ff-ff-ff  static
```


 Terminal — bash — 80x24


```
NITHs-MacBook-Pro:~ foreleser$ arp -a -n
? (10.21.24.1) at 0:22:55:3e:da:ba on en1 ifscope [ethernet]
? (10.21.26.33) at 0:1b:77:76:6c:c6 on en1 ifscope [ethernet]
? (10.21.26.102) at 20:7c:8f:5:d2:cc on en1 ifscope [ethernet]
? (10.21.27.255) at ff:ff:ff:ff:ff:ff on en1 ifscope [ethernet]
NITHs-MacBook-Pro:~ foreleser$
```

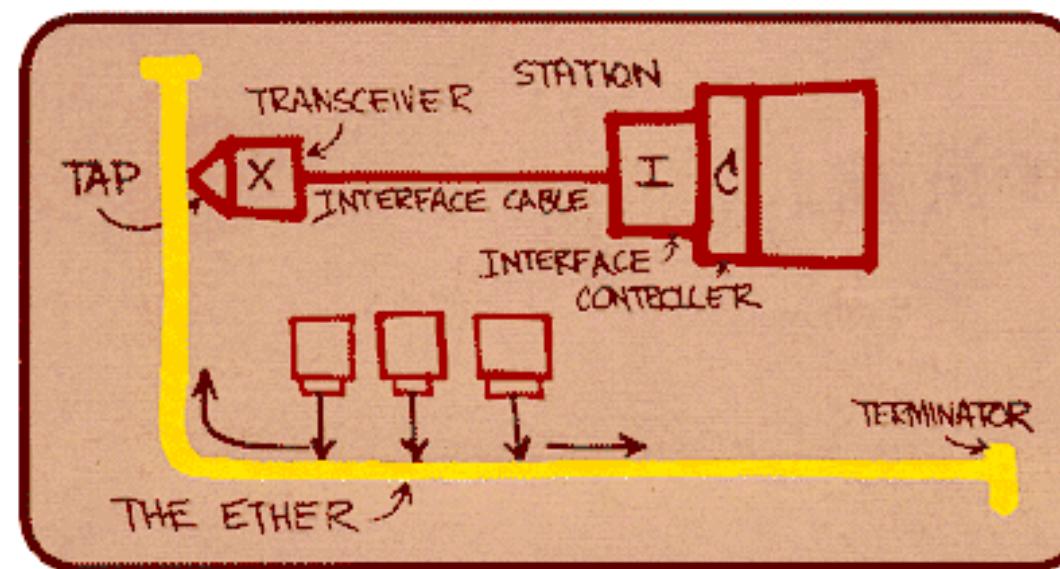
Ethernet



Ethernet

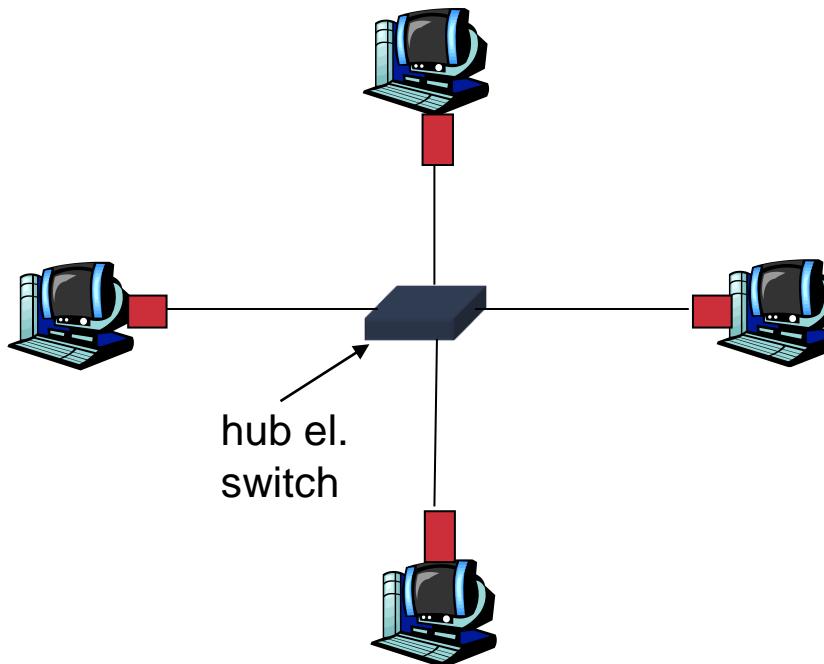
Dominerende lokalnetteknologi:

- billig
- første LAN-teknologi i utstrakt bruk
- Enklere og billigere enn token passing LAN
- Har holdt tritt i hastighetskappløpet: 10, 100, 1000 Mb/s



Bob Metcalfes
Ethernetskisse

- Busstopologi var populær til midten av 90-tallet
 - Ethernet er fremdeles definert med forutsetning om buss-topologi og hvordan løse kollisjoner.
- **Nå** er det stjernetopologi som “går og gjelder”
- Valgmulighet: (hub eller) **switch** (mer senere)



Upålitelig, forbindelsesløs tjeneste

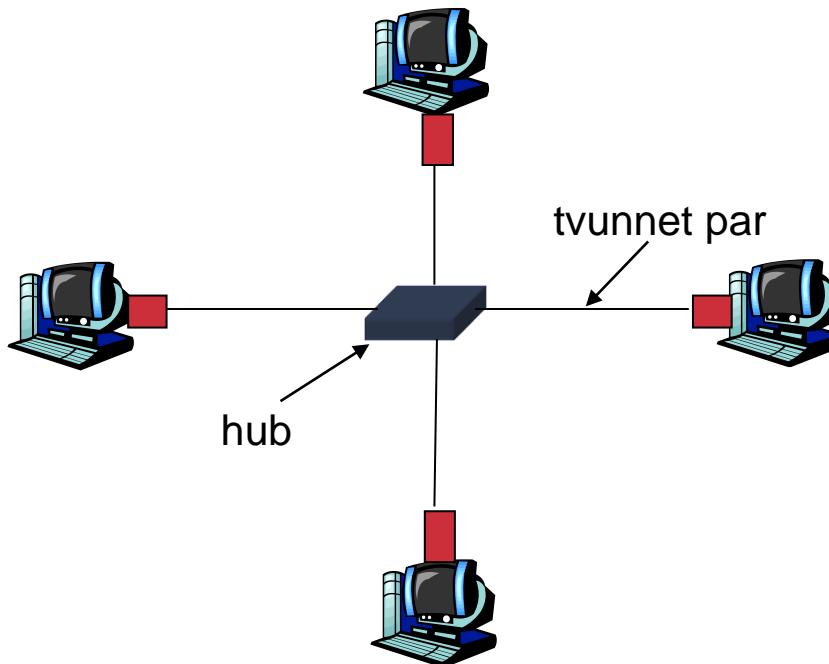
- **Forbindelsesløs:** Ingen håndhilsing mellom sender og mottager
 - Derimot så fremforhandler nettverkskortene hvilken IEEE 802-versjon og bitrate de skal benytte første gang de er i forbindelse
- **Upålitelig:** mottager sender ikke ACK eller NAK tilbake til senderen
 - strømmen av datagrammer som leveres til nettlaget kan ha gap
 - dersom TCP benyttes, sørger denne for å fylle eventuelle gap
 - ellers vil/må applikasjonen se gapene i datastrømmen

Ethernet benytter CSMA/CD

- Ingen tidsluker
- nettkort lytter på nettet før den skal sende (**carrier sense**)
 - sender ikke dersom noen andre allerede sender
- senderen fortsetter å lytte mens den sender og avbryter sendingen dersom den merker at en annen også sender (**collision detection**)
- Før senderen forsøker en retransmisjon, venter den en tilfeldig valgt tid (**random access**)

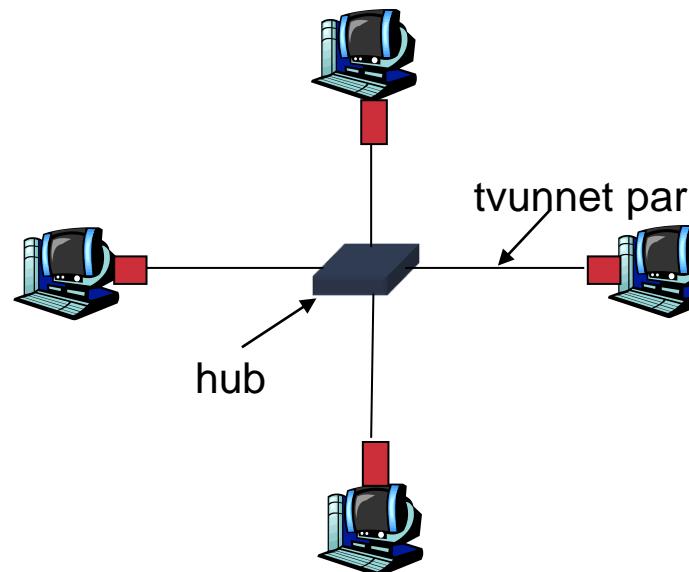
10BaseT og 100BaseT

- 10/100 Mbps rater; sistnevnte kalles “fast ethernet”
- 1 GbE (1000BASE-T)
 - Bruker alle trådparrene, og komplisert koding
- **T** står for “twisted pair” (tvunnet par)
- Noder forbundet med en “hub”(eller switch): stjernetopologi; maks avstand fra node til hub er ca 100 m



Huber er multiport repeatore (fysisk lag):

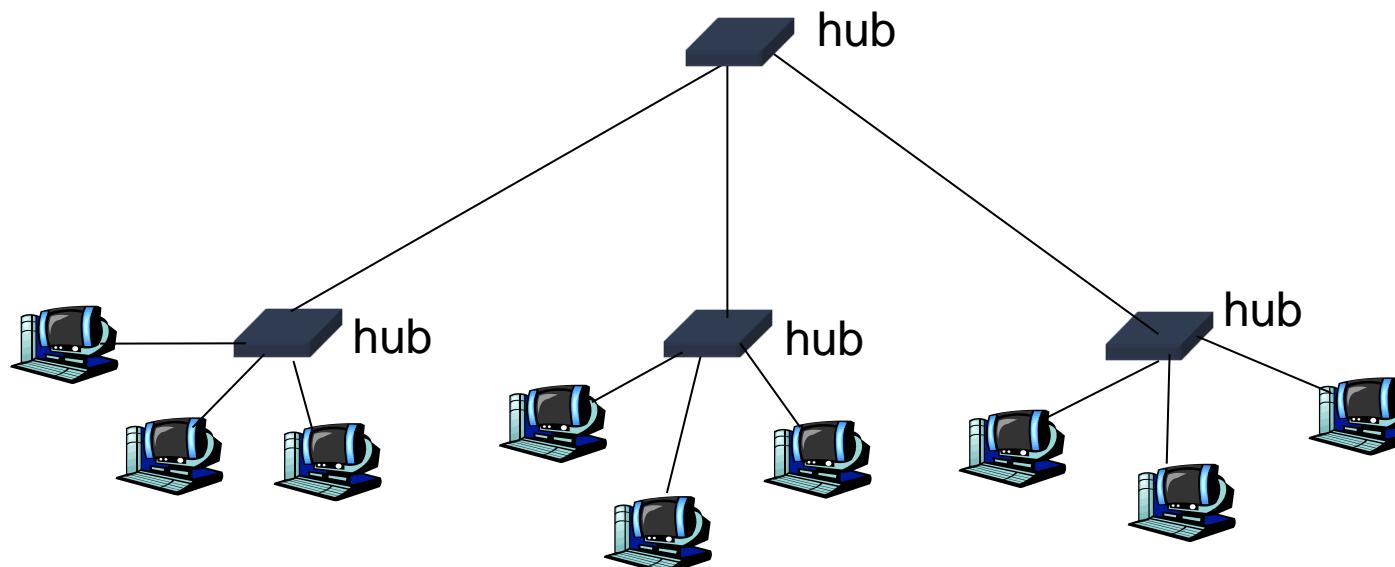
- bit som kommer inn på en link sendes ut på alle andre linker
- ingen buffring av rammer
- ingen CSMA/CD på huben: NIC detekterer eventuelle kollisjoner
- gir visse network management funksjoner



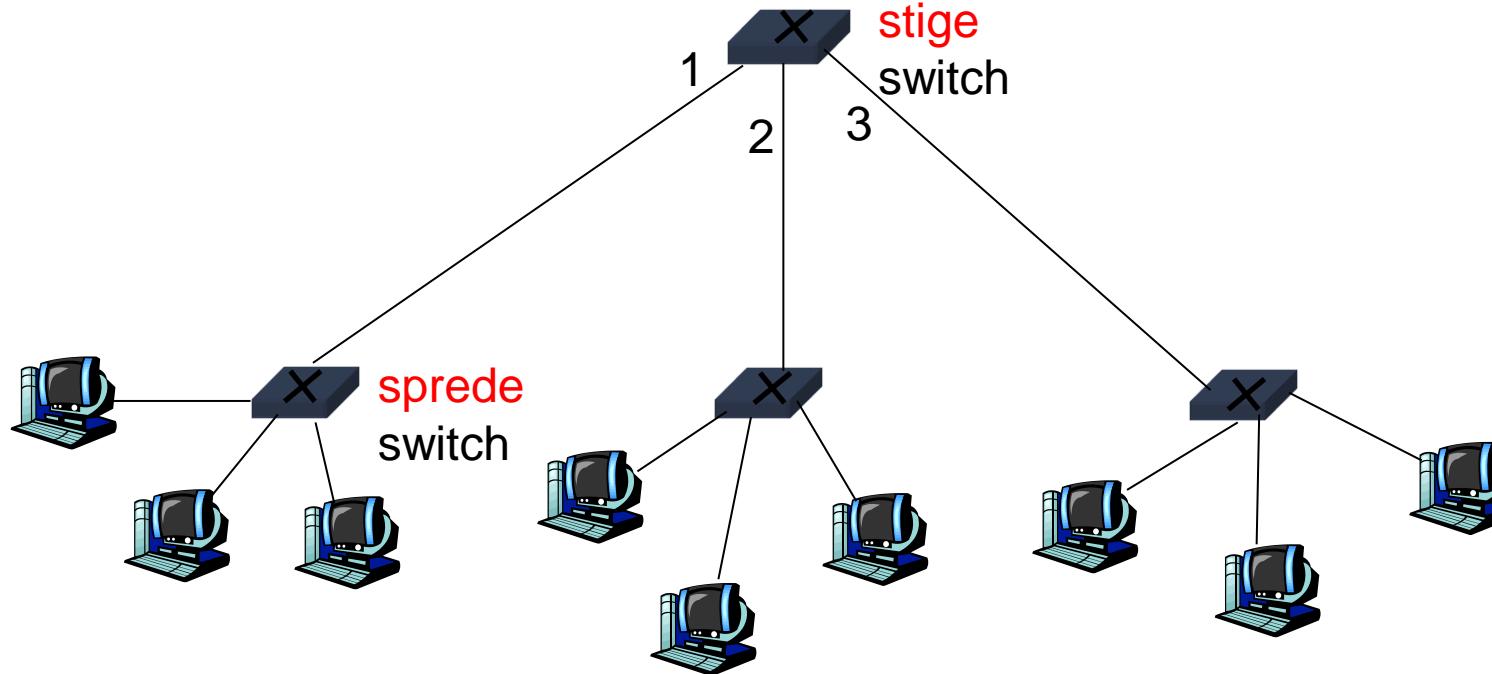


Switch

- Ryggradshub sammenkobler LAN-segmenter
- Utvider maks avstand mellom noder
- Gir ett stort **kollisjonsdomene** (ulempe)
- Kan ikke koble sammen 10BaseT og 100BaseT
 - For å kople sammen ulike fysiske og lag2-teknologier trenger man en **bridge**



- **Linklags-boks**
 - mellomlagrer og videresender ethernetrammer
 - sjekker rammeheader og videresender **når det er behov for det** – basert på mottagers MAC-adresse
 - når rammen skal videresendes på et segment, benyttes CSMA/CD
- **transparent**
 - maskiner er uvitende om svitsjens tilstedeværelse
 - **plug-and-play, selvlæring**
 - svitsjer trenger man ikke å konfigurere



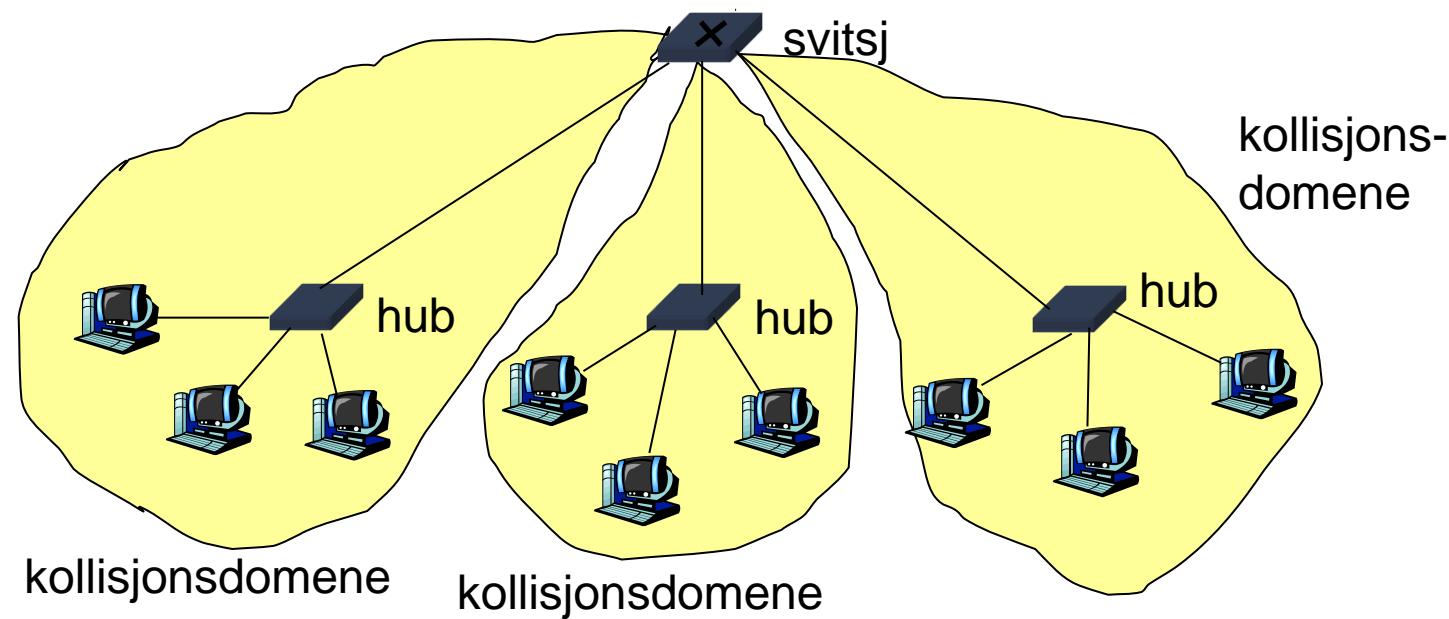
- Hvordan bestemme hvilket LAN-segment rammen skal sendes ut på?
- Ser ut som et routingproblem...

Selvlæring

- En switch har en **switchetabell**
- rader i svitsjetabellen:
 - (MAC-adresse, interface, tidsstempel)
 - foreldede innslag i tabellen slettes (TTL kan være 60 min)
- Switch'en **lærer** hvilke maskiner som kan nås på de ulike interface
 - når den **mottar** en ramme, vil svitsjen "huske" hvilket segment rammen kom fra kombinert med avsenders MAC-adresse
 - lagrer dette i sin svitsjetabell

Destination Address	Address Type	VLAN	Destination Port
0000.001e.2a52	Dynamic	1	Fa1/1
0000.001e.345e	Dynamic	1	Fa1/1
0000.001e.bb3a	Dynamic	1	Fa1/1
0000.001e.eba3	Dynamic	1	Fa1/2
0000.001e.face	Dynamic	1	Fa1/3
0000.001e.3519	Dynamic	1	Fa1/4
0000.001e.2dcl	Dynamic	1	Fa1/5
0000.001e.8465	Dynamic	1	Fa1/5
0000.001e.1532	Dynamic	1	Fa1/5
0000.001e.8ab2	Dynamic	1	Fa1/6
0000.001e.15b1	Dynamic	1	Fa1/6
0000.005a.1b01	Dynamic	1	Fa1/6
0000.005a.4214	Dynamic	1	Fa1/7
0000.005a.5129	Dynamic	1	Fa1/8
0000.00cc.bbe2	Dynamic	1	Fa1/9
0000.00cc.2291	Dynamic	1	Fa1/10

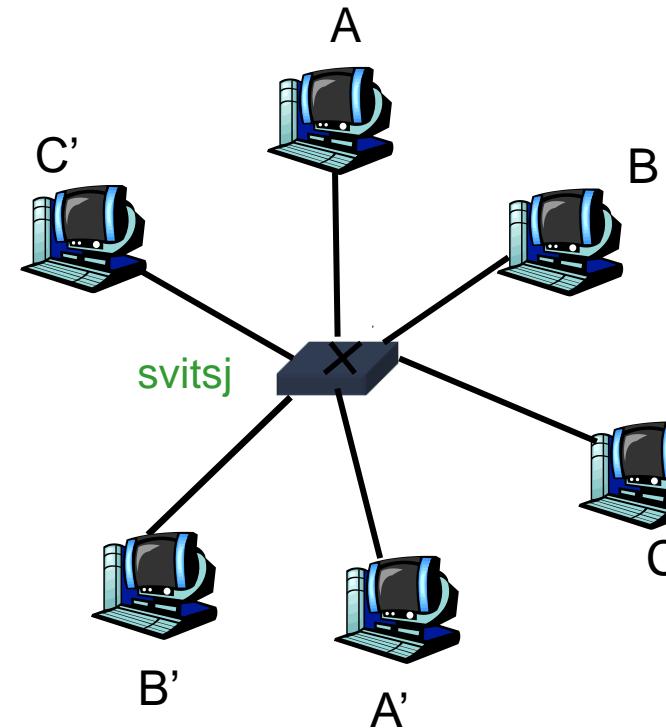
- installering av en svitsj vil dele lokalnettet i segmenter
- svitsjen **filtrerer** rammer:
 - rammer som skal til maskin på samme segment vil normalt ikke bli sendt til andre segmenter
 - segmentene blir separate **kollisjonsdomener**



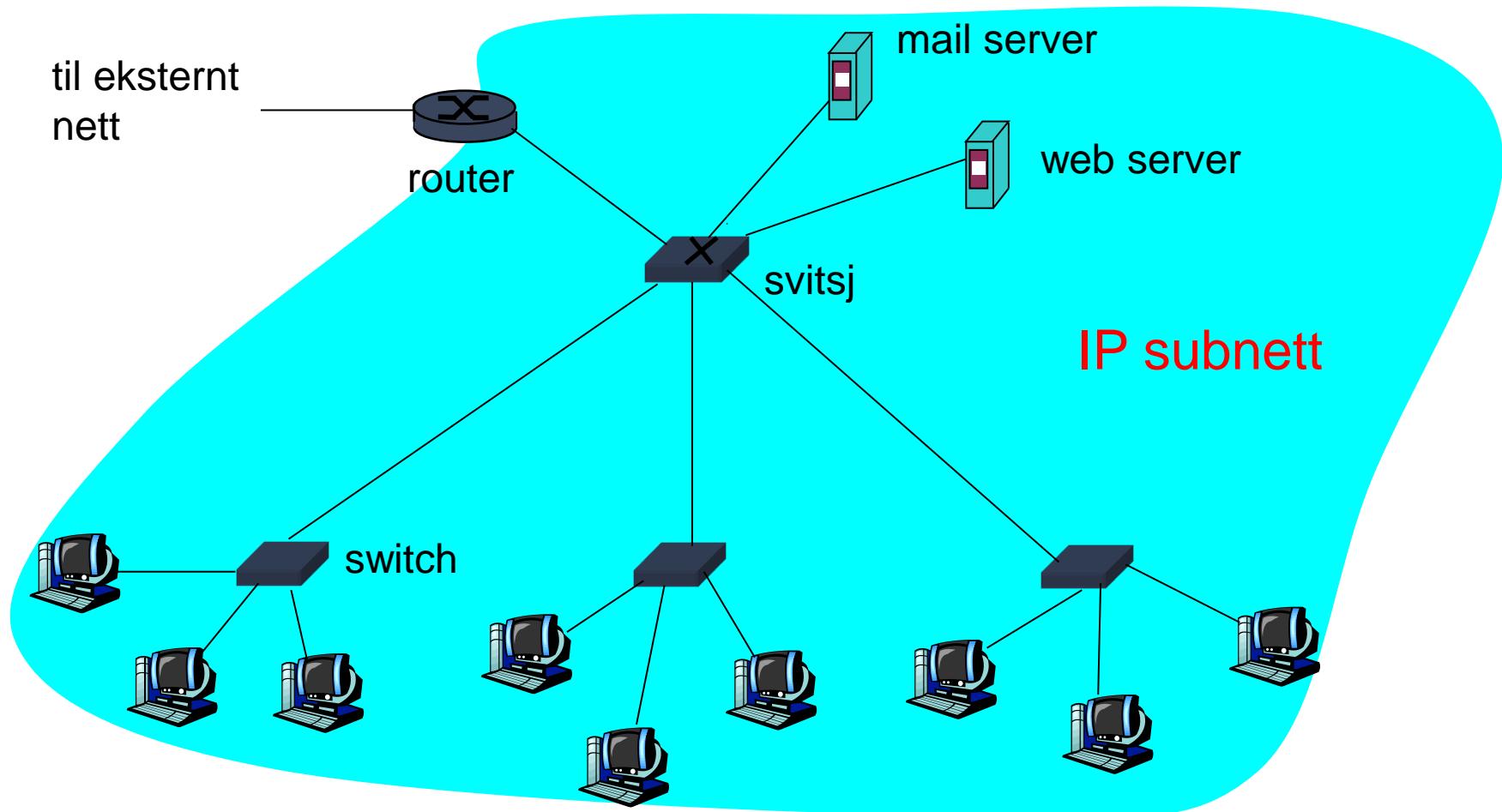
Svitsjer: dedikert aksess

- Svitsj med mange interface
- Maskiner har direkte forbindelse til svitsjen
- Ingen kollisjoner, full dupleks

Svitsjing: A-til-A' og B-til-B'
samtidig, ingen kollisjoner

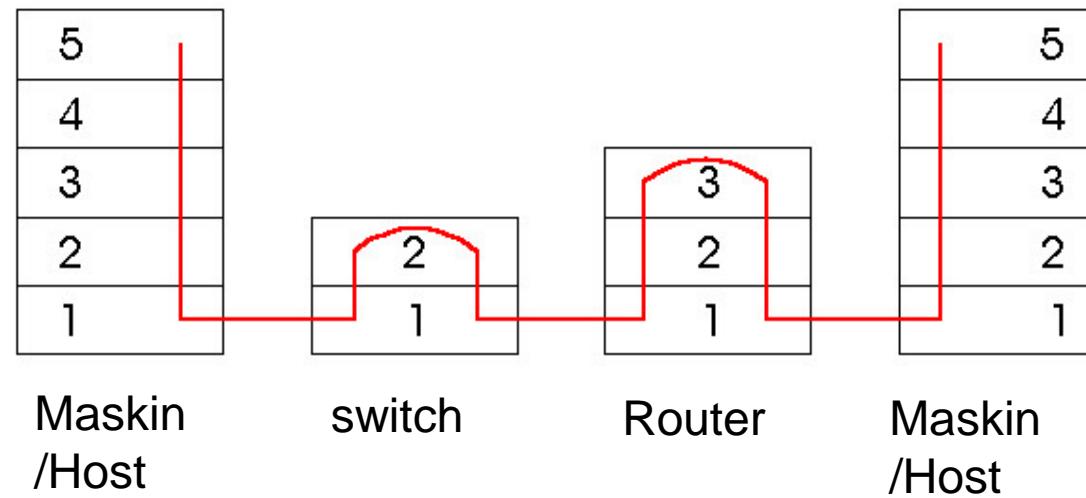


Nettverk for en institusjon



Svitsjer vs. routere

- begge er “store-and-forward” enheter
 - routere: nettlagsenheter (ser på nettlagsheadere)
 - svitsjer er linklagsenheter
- routere benytter routingtabeller og implementerer routingalgoritmer
- svitsjer benytter svitsjetabeller, gjør filtrering, og har selvlæring



Sammenligning - oppsummering

	<u>hub</u>	<u>ruter</u>	<u>svitsj</u>
trafikk- isolasjon	nei	ja	ja
plug & play	ja	nei	ja
optimal ruting	nei	ja	nei
cut through	ja	nei	ja

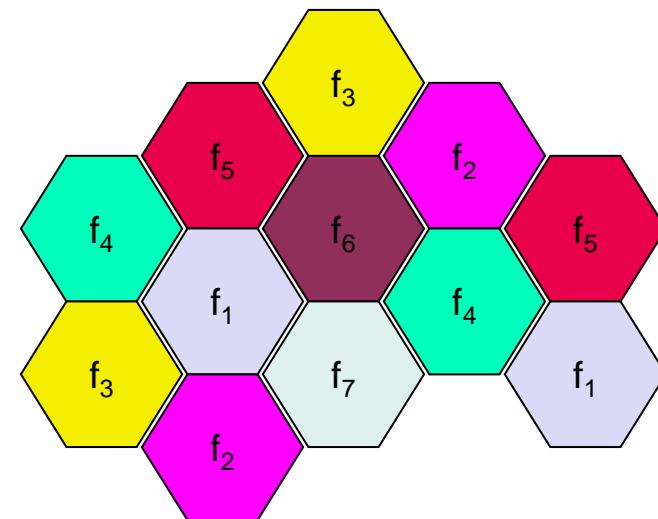
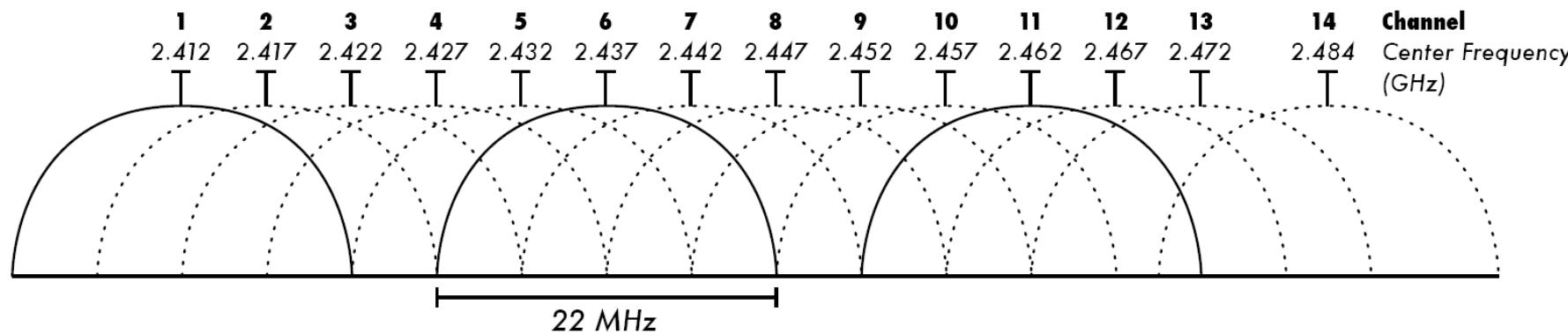


IEEE 802.11

IEEE 802.11 Wireless LAN (Wi-Fi)

- Alle bruker **CSMA/CA** for tilgang
- Alle tilbyr både base-stasjon (AP) og ad-hoc nettverk versjoner
- **802.11b**
 - 2.4 GHz lisensfritt radiobølgområde
 - opp til 11 Mbps
 - direct sequence spread spectrum (DSSS) i fysisk lag
 - Ligner CDMA, men alle vertsmaskiner bruker samme “chipping code”
 - Rekkevidde 38 / 140 m
 - Begynner å fases ut til fordel for **g** og **n**; men de fleste trådløse kort støtter den fremdeles.
- **802.11a**
 - 5-6 GHz
 - opp til 54 Mbps
 - OFDN
- **802.11g**
 - 2.4 GHz området
 - Opp til 54 Mbps
- **802.11n**
 - 2.4 og/eller 5 GHz området
 - Opp til 150 Mbps
 - Rekkevidde 70 / 250 m
 - Flere (4) antenner (**MIMO**)
 - *Forward Error Correction*

Kanaler: 1-13, 802.11 b/g/n



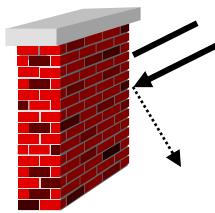
Signalforplantning

Signalforplantning i fritt rom samme som for lys (rett linje)

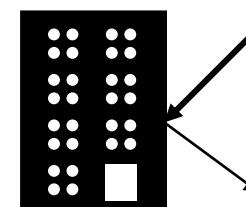
Mottatt effekt proposjonal med $1/d^2$
(d = avstand mellom sender og mottager)

Mottatt effekt/signal også påvirket av

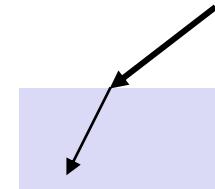
- **demping** (frekvensavhengig)
- hinder, ting og tang
- **refleksjon** fra store hindringer
- **refraksjon** avhengig av tettheten på mediet
- **spredning** fra små hindringer
- **diffraksjon** på kanter



hinder



refleksjon



refraksjon

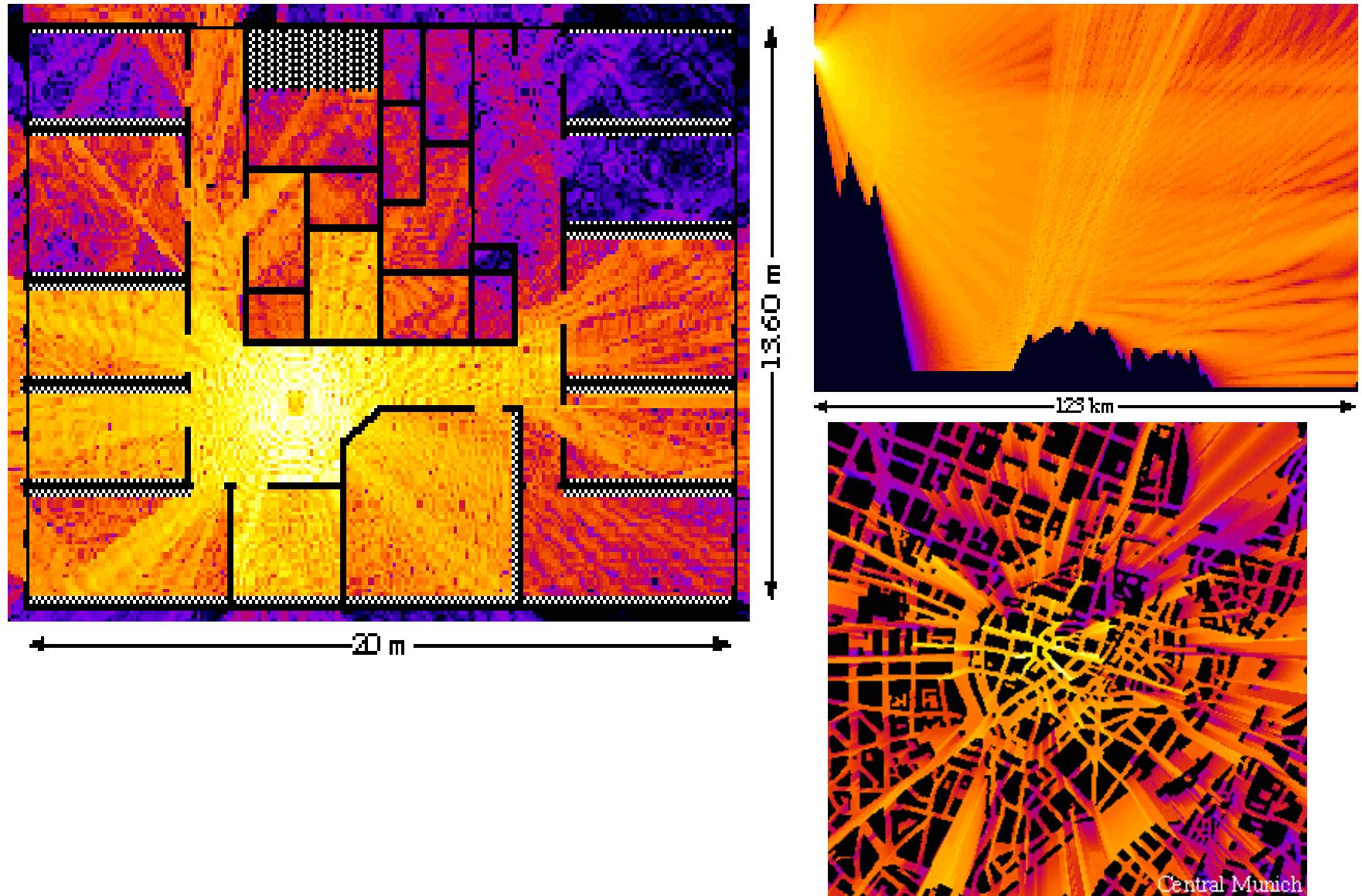


spredning

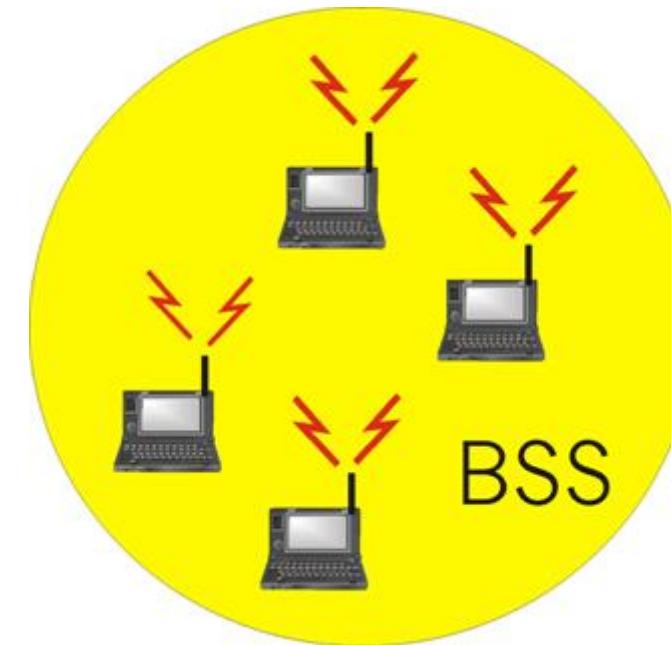


diffraksjon

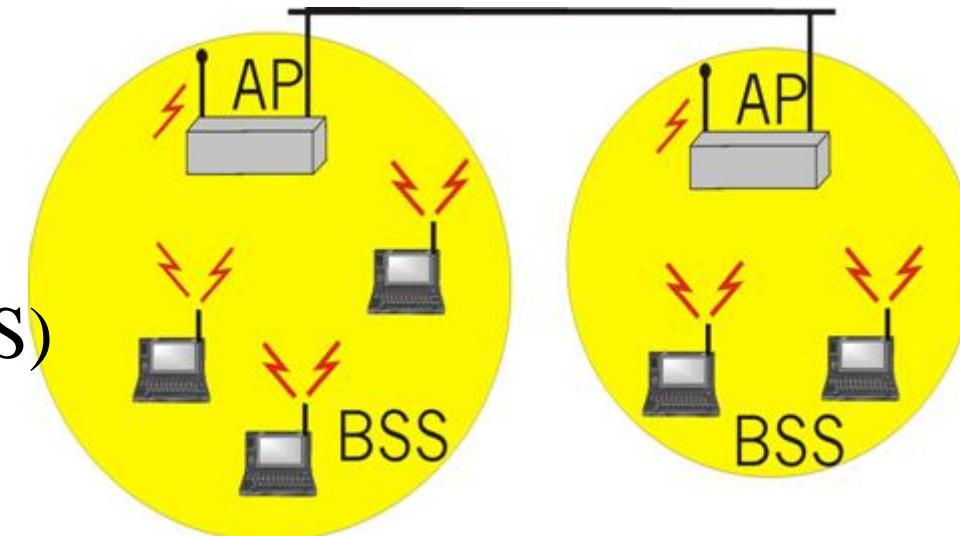
Eksempel: radiostråling / intensitet



- Intet AP (i.e., ingen basestasjon)
- Trådløse vertsmaskiner kommuniserer direkte og fungerer som switcher/routere for hverandre.
- Anwendelser
 - “laptop” møter i bilen, på hytta e.l.
 - slagmarken

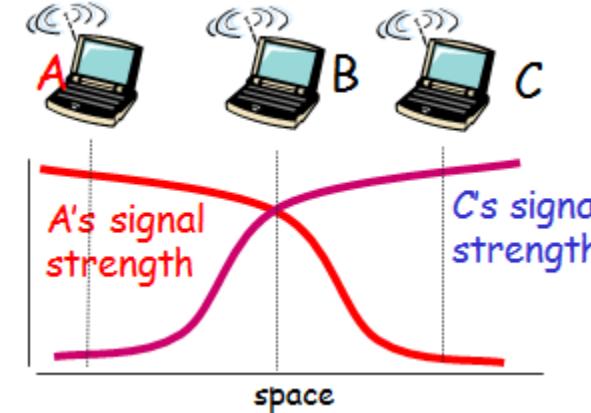
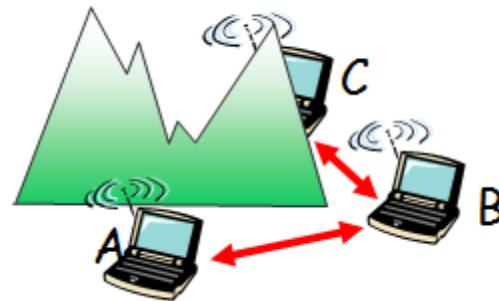


- Vertsmaskinene kommuniserer via en basestasjon
 - basestasjon = **access point (AP)**
 - **Basic Service Set (BSS)** (tilsvarer “celle” i mobiltelefonsammenheng) består av:
 - Trådløse vertsmaskiner
 - Access Point (AP)
- BSS’er kombineres til et DistribusjonsSystem (DS) (WLAN)
- BSS har en SSID



Trådløst LAN: Problem ->CA

- Hvis to terminaler er "gjemt" for hverandre kan det oppstå kollisjoner



- Collision Detection er dermed ikke tilstrekkelig
- Bruker CSMA/CA (Collision Avoidance)
- Kan benytte Request-To-Send og Clear-To-Send signaler for å reservere forbindelse

- Problem:
 - To noder, som er skjult for hverandre, sender ut komplette pakker til basestasjonen der de interfererer og går tapt.
 - Bortkastet båndbredde for alle!
- Løsning:
 - Små reservasjonspakker!
 - Nodene holder selv orden på reservert tid med egen “network allocation vector” (NAV)

- **Mål**
 - Tilgangskontroll
 - Kun den som blir gitt tilgang skal få det
 - **DataIntegritet**
 - Ingen skal kunne endre innholdet i datapakkene under overføring («man in the middle attacks»)
 - **Konfidensialitet**
 - Forhindre avlytting og sesjons-kapring
- **Metode**
 - Kryptering av trafikken mellom AP og brukermaskin
- **Teknikker**
 - **WEP**
 - Lett å knekke pga repeterende 40 bit nøkler mm; bedre med 128 bit nøkkel men fremdeles lett å knekke
 - Statisk nøkkel
 - **WPA 1 & 2**
 - Ny krypteringsnøkkel for hver enkelt pakke = bedre kryptering
 - WPA 1 lar seg cracke,
 - WPA 2 lar seg også cracke
 - WPA 2 lar deg også benytte autentisering-server (sikrere en PSK = felles passord))
 - CCMP krypering er sterkere en TKIP
 - **EAP** (Extensible Authentication Protocol)
 - Skal gjøre ulike WPA-Enterprise metoder interoperable.

Hvordan sikre eget trådløst nett

- Ikke bruk default-innstillinger inn mot eget WLAN/ISP
- Oppdater firmware på AP og drivere for WNIC (minimum årlig)
- Slå av «Remote Administration» på AP
 - Velg komplisert/sikkert Admin-passord
- Velg «uvanlig» IP-nett
 - gjerne fra 172.16.0.0/12 området
 - begrens DHCP-pool
- Filter tilgang på MAC-adresser
- Slå av SSID-broadcast
- Sørg for å sjekke rekkevidden på signalet!
- Bruk WPA2 med langt og komplisert passord

AVSLUTTNING

Hva skal vi kunne?

- Hvilke **oppgaver** løses på linklaget?
- Hvilke mekanismer for feildeteksjon finnes og benyttes?
 - Beregne paritetsbit og bruke 2D-paritet, beregne **CRC**
- Hvilke måter finnes for å dele medium (MA)?
 - Hvordan unngå, eller håndtere **kollisjoner**?
- **MAC-adresser**
 - Oppbygging, multicast, broadcast
- **IEEE 802.3 / Ethernet 2**
 - **Oppbygging av rammen**
 - Kjenne til de ulike typene.

Hva skal vi kunne?

- Hvorfor **ARP** trengs og hvordan den virker
 - Herunder bruk av `arp`-kommandoen
- Kjenne til de forskjellige LAN-topologiene
 - Buss, stjerne, ring
- Vite forskjellen på **hub** (nav), **bridge** (bro), **switch** (svitsj) og **router** (ruter).
- Kunne forklare hvordan en **switch** fungerer
 - Hvordan bygges switche-tabellen opp?
- Kunne forklare hvilke nye **problemer** som må løses i trådløse nett (**IEEE 802.11**)
 - Forklare rollen til AP, hva er SSID

Om eksamen (samme vi gikk gjennom tidligere)

Om eksamen

Det er 24 timers frist på denne hjemmeeksamen, men forventet arbeidsmengde er 4-6 timer så det er ikke meningen å «jobbe gjennom natten». Vær obs på at eksamen MÅ leveres innen fristen som er satt, og må leveres via eksamensplattformen WISEFLOW. Det vil ikke være mulig å få levert oppgaven etter fristen – det betyr at du bør levere i god tid slik at du kan ta kontakt med eksamenskontoret eller brukerstøtte hvis du har tekniske problemer.

Da dette er en hjemmeeksamen er det viktig å vise helhetlig forståelse, og oppgavene har et større preg av drøfting. Det forventes derfor utfyllende og forklarende svar på alle oppgaver. Figurer og skisser kan du velge å tegne i tekstbehandleren, eller ved å tegne på papir og laste opp bilde – husk å sette inn bilde på riktig sted i besvarelsen. (Bilder som er vedlegg, men ikke satt inn i besvarelsen anses ikke som en del av besvarelsen.)

Om eksamen

Det presiseres at studenten skal besvare eksamen selvstendig og individuelt, samarbeid mellom studenter og plagiat er ikke tillatt.

I regneoppgaver er det vesentlig at du legger vekt på å vise hvordan du kommer frem til svaret. Svar på regneoppgaver uten å vise fremgangsmåte er å betrakte som ubesvarte oppgaver.

OBS: Besvarelsen skal ikke være på mer enn 15 A4 sider, med font størrelse 12, normale marger og linjeavstand 1.0.

Om eksamen

Oppgave 1: Teori spørsmål (15%)

Oppgave 2: Tall og binære data (35%)

Oppgave 3: Praktiske oppgaver (30%)

*De praktiske oppgavene vil teste din forståelse av **verktøy** og av protokoller, de vil ikke være lik de vi har hatt i øvingene, men øvingsoppgavene vil være veldig relevante!*

Oppgave 4: “Tyngre” teori og forståelse (20%)

Dagens øving

- Øvingsoppgaver på Canvas
- Skriv ned 3 emner du føler du sliter med, og som du må jobbe mer med før eksamen
- Husk repetisjonsforelesning dagen før eksamen!
- Gjennomfør en tidligere eksamen
 - På Canvas under oppgaver har jeg lagt ut eksamen fra 2020
 - På en hjemmeeksamen vil det være litt mer fokus på å vise forståelse, og litt færre rene «pugge» oppgaver. I år vil det som dere vet også være noen praktiske oppgaver.
 - Figurer og utregninger kan tegnes på papir, sett inn bildet i besvarelsen :-)

Eksempler praktiske oppgaver

- Bruke et standard nettverksverktøy på kommandolinje og kopiere inn output fra verktøyet i besvarelsen; for eksempel **traceroute**
- Avanserte verktøy:

*I denne oppgaven skal du demonstrere forståelse for bruk av et raw-socket emulator verktøy som **PuTTY** (Windows) eller **telnet** (Linux/OSX), god forståelse for SMTP protokollen kreves også for å fullføre denne oppgaven.*

Du skal koble deg til SMTP på standard port (Raw connection type), på host `send.one.com`. Du skal sende en EPOST fra `[kandidatnummer]@h-ck.me` til `besvarelse@h-ck.me`, eposten skal ha tittel «OPPGAVE 42» og body «OK».

Hvilket svar får du tilbake fra denne serveren (oppgi fullstendig svar fra server)? Forklar fremgangsmetoden du brukte for å løse oppgaven.

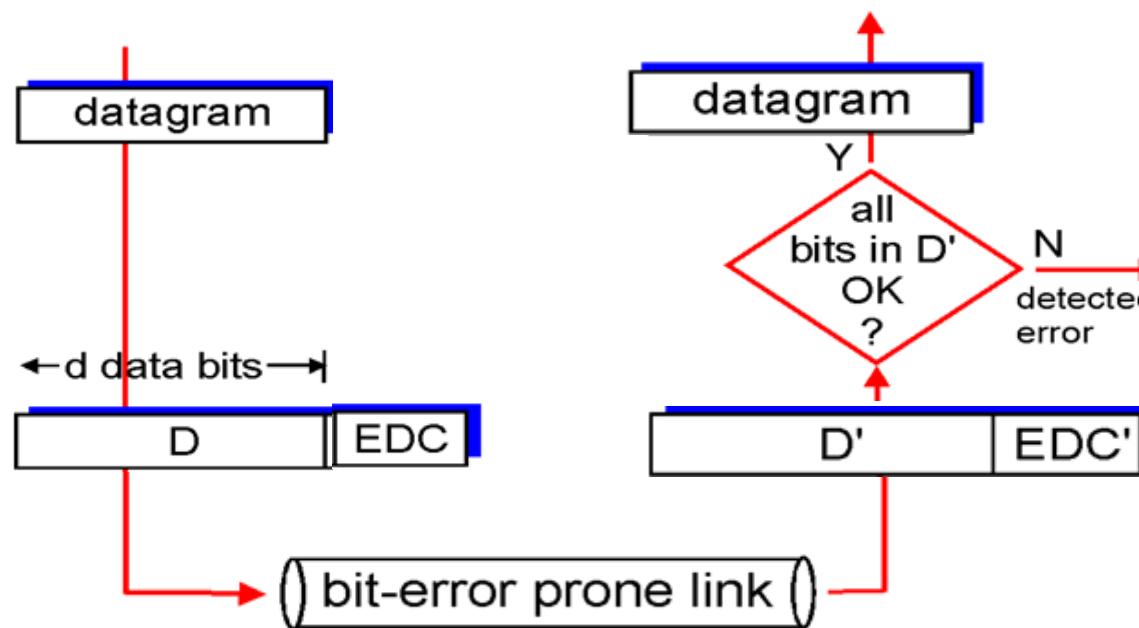
For valgfritt egenstudie

For de som ønsker å lære noen emner mer i dybden for å forstå det bedre er det her samlet noen ekstra temaer relatert til dagens undervisning, det må forventes en del egenarbeid for å forstå disse emnene.

Det vil ikke komme spørsmål på eksamen fra disse, og dette er altså ikke ansett for å være en del av pensum.

Feildeteksjon

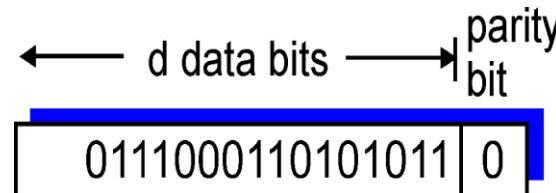
- **D** = Data beskyttet av feilsjekk, kan omfatte header-felter
- **EDC** = Error Detection and Correction bits (redundante bit)
- Feildeteksjon er ikke 100% pålitelig
 - protokollen kan overse feil (selv om det er sjeldent)
 - **flere feildeteksjonsbit** gir **bedre** deteksjon og mulighet for **retting**



Paritetssjekk

Ett-bits paritet:

Kan oppdage dersom ett bit er feil



Like paritet:

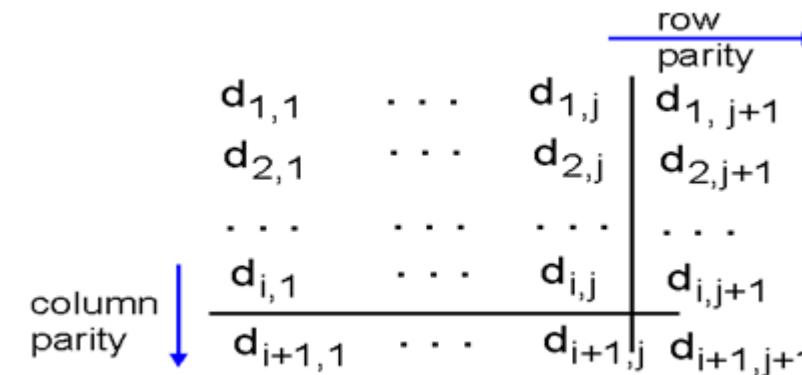
Det totale **antall enere** (inkl paritetsbit) skal være et **partall**

Odde paritet:

Det totale **antall enere** (inkl paritetsbit) skal være et **oddetall**

Todimensjonale paritetsbit:

Kan oppdage **og rette** dersom ett bit er feil



101011
 111100
 011101
 \hline
 001010

no errors

101011
 $1\cancel{0}1100$ → parity error
 011101
 001010

101011
 $1\cancel{0}1100$
 011101
 001010

correctable single bit error

Internett-sjekksum (repetisjon)

Mål: oppdage bitfeil i mottatt segment

Sender:

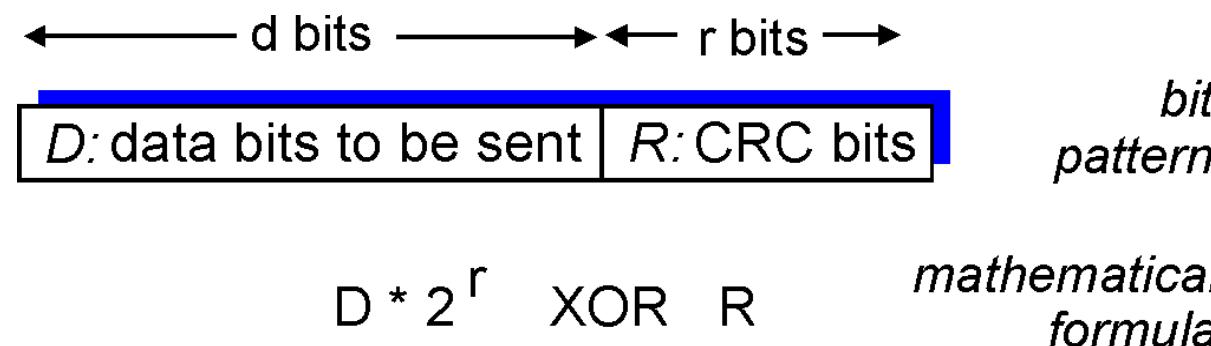
- behandler innholdet i segment/datagram som sekvens av 16-bits tall
- sjekksum: addisjon (eners komplement) av innholdet i segmentet
- sender legger sjekksum inn i UDP/TCP og IPv4 sjekksum-felt

Mottager:

- beregner sjekksum av mottatt segment
- ser om beregnet sjekksum er korrekt:
 - NEI → feil oppdaget
 - JA → ingen feil oppdaget. *Men det kan allikevel finnes feil...*

Cyclic redundancy check (CRC)

- ser databitene, D , som et binært tall
- velger et bitmønster, $r + 1$ bit langt → generator, G
- mål: velge r CRC-bit, R , slik at
 - DR er delelig med G (modulo 2)
 - mottager kjenner G og dividerer DR med G .
Dersom divisjonen gir en rest, er det bitfeil
 - oppdager alle **skur-feil** mindre enn $r+1$ bit
- mye benyttet i praksis (ATM, HDCL, Ethernet, zip, ...)



CRC eksempel

Ønsker:

$$D \cdot 2^r \text{ XOR } R = nG$$

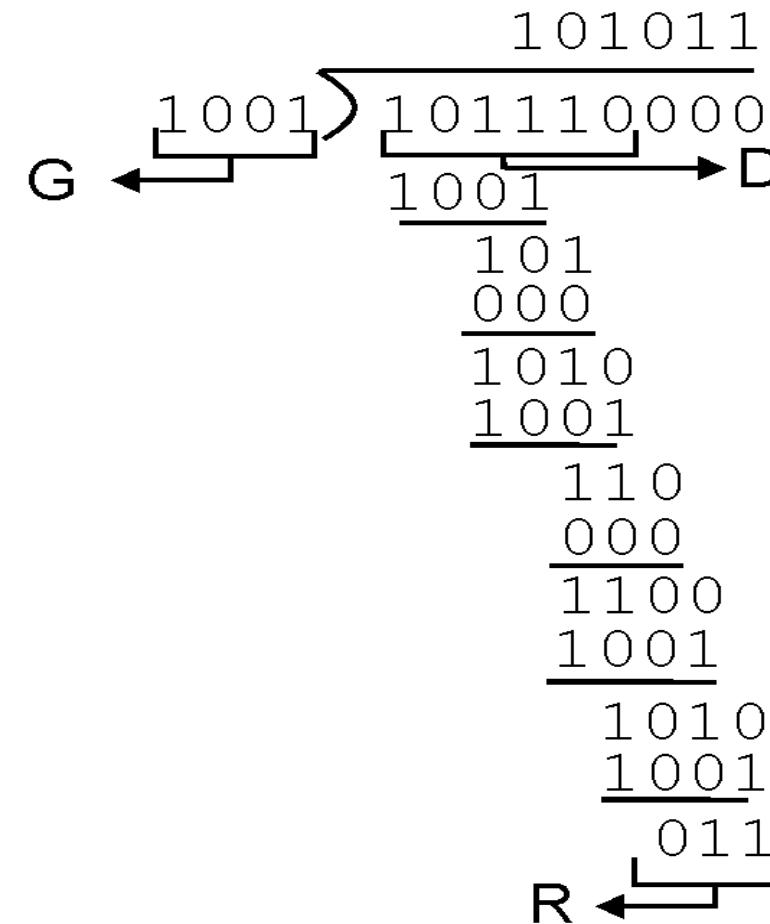
ekvivalent:

$$D \cdot 2^r = nG \text{ XOR } R$$

ekvivalent:

hvis vi deler $D \cdot 2^r$
med G er det *resten*,
 R , vi søker

$$R = \text{rest}\left[\frac{D \cdot 2^r}{G}\right]$$



CRC-32

- Bruker noen forskjellige nøkler
 - $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Enkel å beregne i hardware
 - XOR-porter og skift-registre
- Oppdager alle burst-**feil** som er på 32 bit eller færre
- Oppdager $1 - 2^{-32} = 99,9999999767\%$ av alle feil som består av flere enn 32 bit
- Brukes også av ZIP, MPEG, PNG, m.fl.

“Etter tur” MAC-protokoller

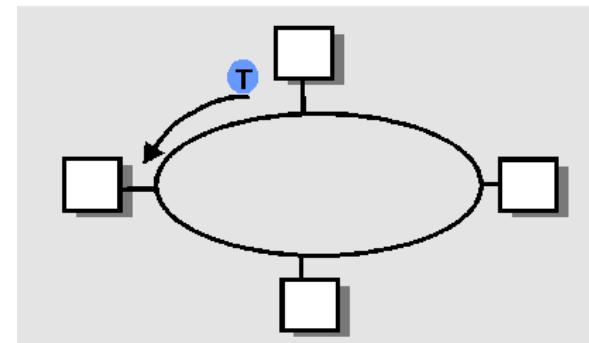
Alternativer til *tilfeldig tilgang*, finnes og brukes noen steder, f.eks. IEEE 802.5

Polling:

- master-node “inviterer” slavenoder til å sende – en om gangen
- ulemper:
 - overhead pga pollingen
 - latens (forsinkelse) – må vente på tur
 - single point of failure (master)

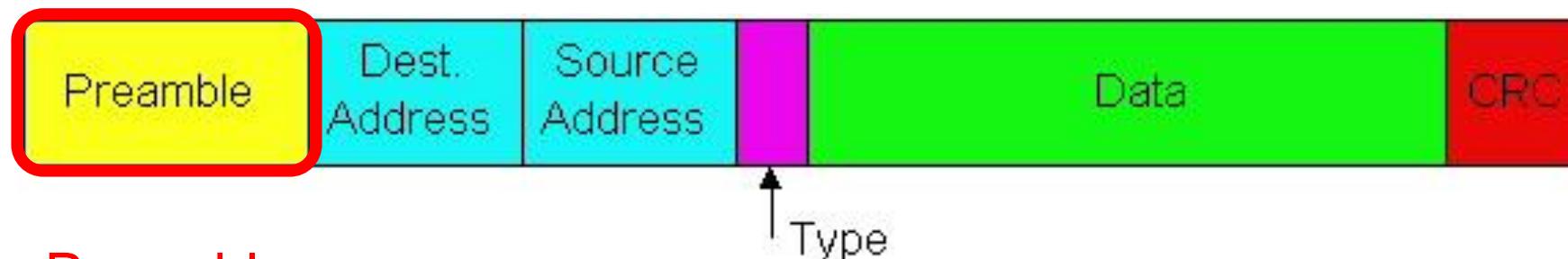
Token passing:

- en spesiell ramme – **token** (stafettpinne), sendes fra node til node
- token message
- ulemper:
 - token overhead
 - latens (forsinkelse)
 - single point of failure (token)



Ethernets rammestruktur

Nettverkskort (NIC, adapter) legger IP-datagrammet (eller annen nettlags-PDU) i en **Ethernetramme**

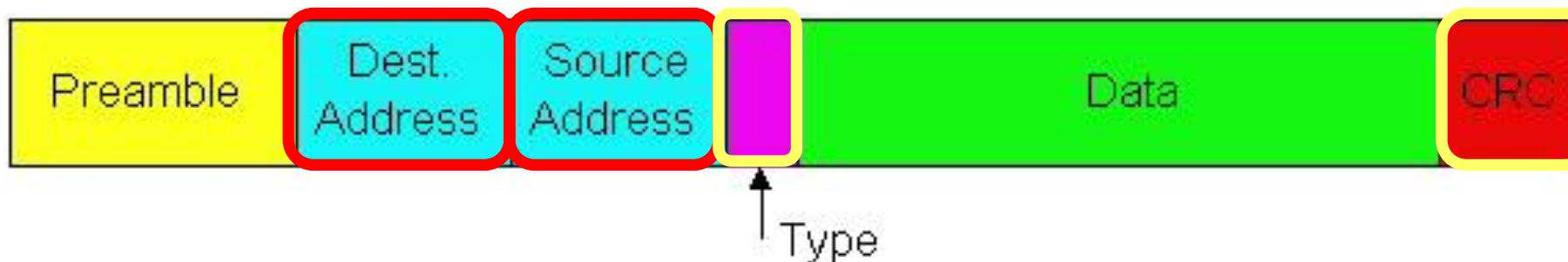


Preamble:

- 7 oktetter (7 byte) med bitmønster 10101010 fulgt av én oktett med bitmønster 10101011
- benyttes for å synkronisere mottagers "klokke" med senderens

Ethernets rammestruktur (forts)

- **Adresser:** 6 oktetter (48 bit)
 - hvis NIC mottar ramme med egen adresse som destinasjonsadresse eller en kringkastingsramme (f eks ARP-pakke), leverer den data i rammen til nettlags-protokollen
 - ellers kaster den rammen
- **Type:** indikerer hvilken nettlagsprotokoll data tilhører (normalt IP, men også andre muligheter, f. eks. Novell IPX eller AppleTalk)
- **CRC:** feildeteksjon (cyclic redundancy check) – hvis feil oppdages, kastes rammen



Ethernet CSMA/CD algoritmen

1. Nettkort får datagram fra nettlag og lager en ramme
2. Sender lytter på mediet for å se om det er ledig. Hvis ingen andre er å høre, vil nettkortet starte sendingen. Hvis mediet er opptatt, venter den til det blir ledig og sender deretter
3. Hvis hele rammen er sendt uten kollisjon, er nettkortet ferdig med rammen
4. Hvis senderen oppdager at en annen sender samtidig med den selv, avbryter den sendingen og sender i stedet et jamme-signal
5. Etter avbruddet vil senderen foreta en **“exponential backoff”**: etter kollisjon nr m, velger senderen tilfeldig en K fra mengden $\{0,1,2,\dots,2^m-1\}$. Så venter den $K \cdot 512$ bit-tider og returnerer til trinn 2.

Ethernets CSMA/CD (forts)

Jammesignal: for å forsikre seg om at alle er oppmerksom på kollisjonen; 48 bit

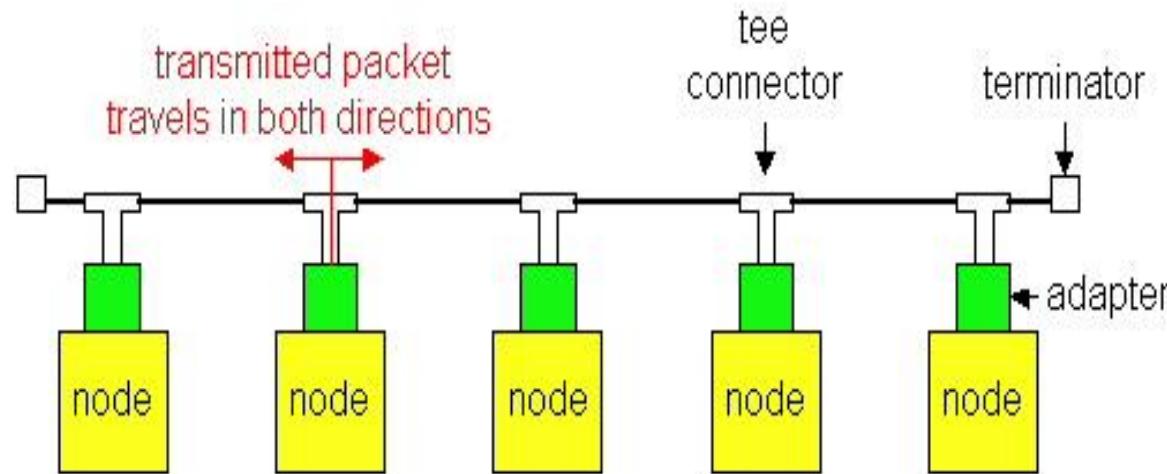
Bit-tid: 10 ns for 100 Mb/s Ethernet; for $K = 1023$ vil følgelig ventetiden være omkring 5 ms

Eksponential Backoff:

- **Mål:** tilpasser forsøk på retransmisjon etter estimert last for øyeblikket
 - stor belastning: tilfeldig ventetid ofte lenger
- første kollisjon: velg K fra $\{0, 1\}$; ventetid er $K \cdot 512$ bit-tider
- etter andre kollisjon: velg K fra $\{0, 1, 2, 3\}$
- etter ti kollisjoner: velg K fra $\{0, 1, 2, 3, 4, \dots, 1023\}$
- Dersom fremdeles ikke sendetid: Gi opp..

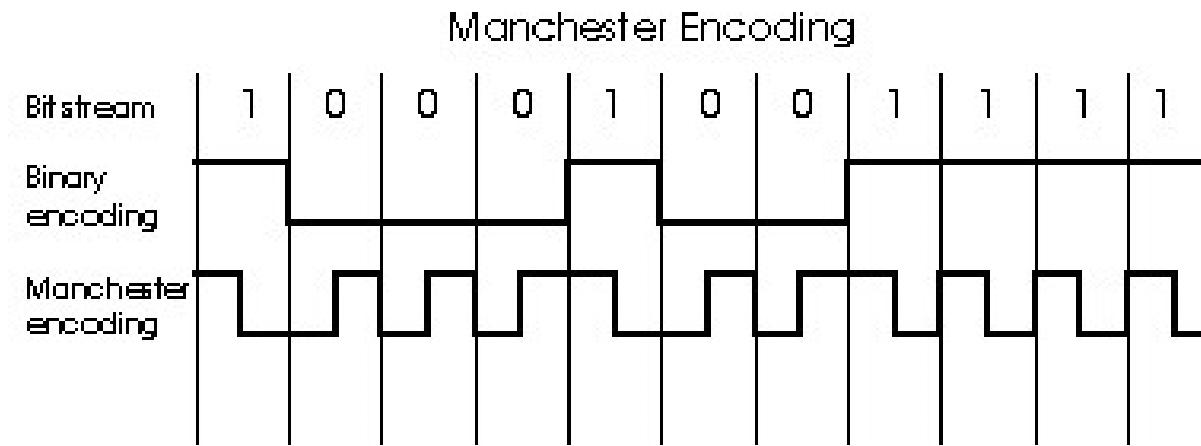
Ethernet-teknologier: 10Base2

- 10: 10 Mb/s; Base: basisbånd; 2: maks 200 meters kabel
- tynn koaksialkabel i en busstopologi



- maks 30 noder pr *segment*
- repetere brukes for å knytte sammen flere segmenter
- repeater gjentar bit den hører på ett interface på sine andre interface: opererer på fysisk lag!

Manchester-koding



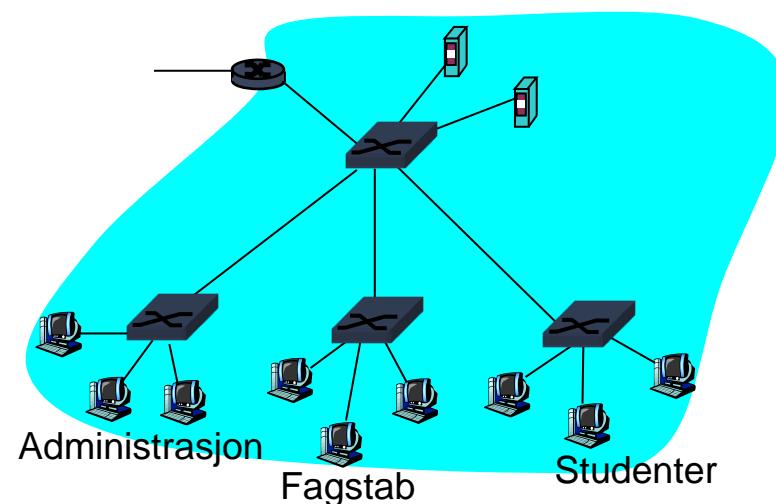
- Brukes i 10BaseT og 10Base2
- Hvert bit har en overgang (lav-til-høy el. høy-til-lav)
- Muliggjør synkronisering av sender og mottaker
 - mottager må vite hvor hvert bit starter
 - ikke behov for sentralisert, global klokke
- (Dette tilhører **fysisk** lag – ikke linklaget)

Gigabit ethernet (1000Base-T)

- benytter standard ethernet rammeformat
- Benytter alle **fire** trådparrene i UTP-kabelen
- ved delte kanaler brukes CSMA/CD; bør ha korte avstander mellom noder for topp ytelse, men kan brukes opp til 100 m
- Benytter flere avanserte kodingsteknikker: 5 nivå puls-amplitude-modulering m.fl.
- Full-dupleks ved 1 Gb/s for punkt-til-punkt linker
- Finnes også for fiber mm
- 10 Gbps finnes og vinner stadig terreng
 - Mange ulike fysiske standarder (PHY)
 - 10GBASE-T for Cat 6, 6A eller 7 UTP med RJ45

VLAN: motivasjon

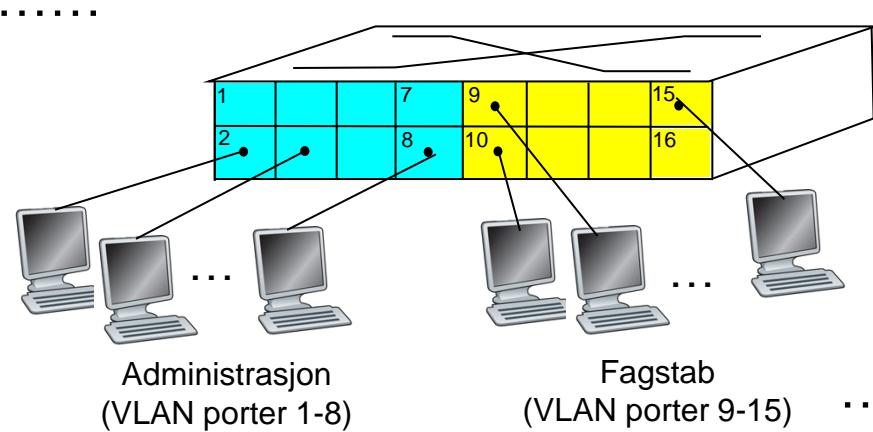
Hva er feil med dette LANet?



- Hva skjer dersom: Adm-bruker bytter kontor til Fagstab-gangen, men vil fortsette å henge på Adm-switch?
- Ett enkelt **broadcast** domene:
 - all lag-2 broadcast trafikk (ARP, DHCP) sendes til hele LANet (sikkerhet/privatliv, ineffektivt)
- Sprede-switchene bruker bare et fåtall av portene sine..

Virtuelle LAN

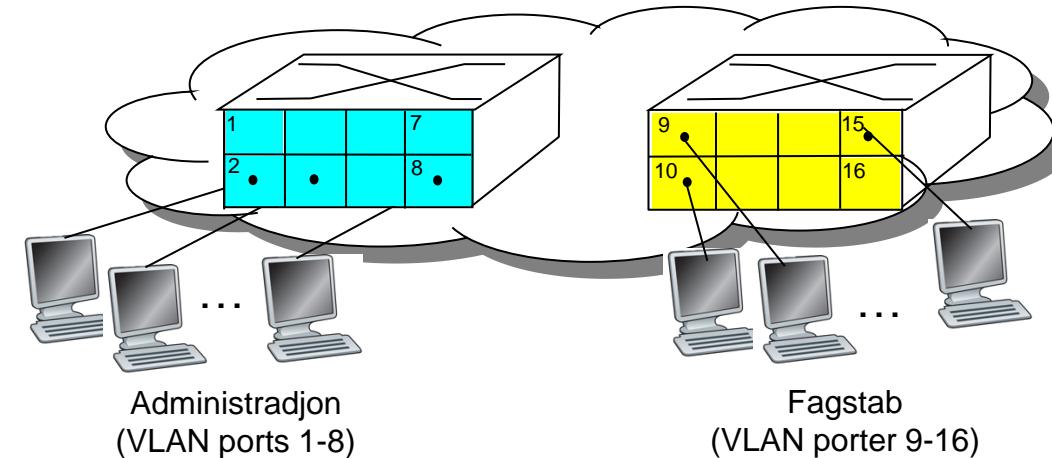
Port-basert VLAN: switch porter grupperes (med switch management software) slik en *enkelt* fysisk switch



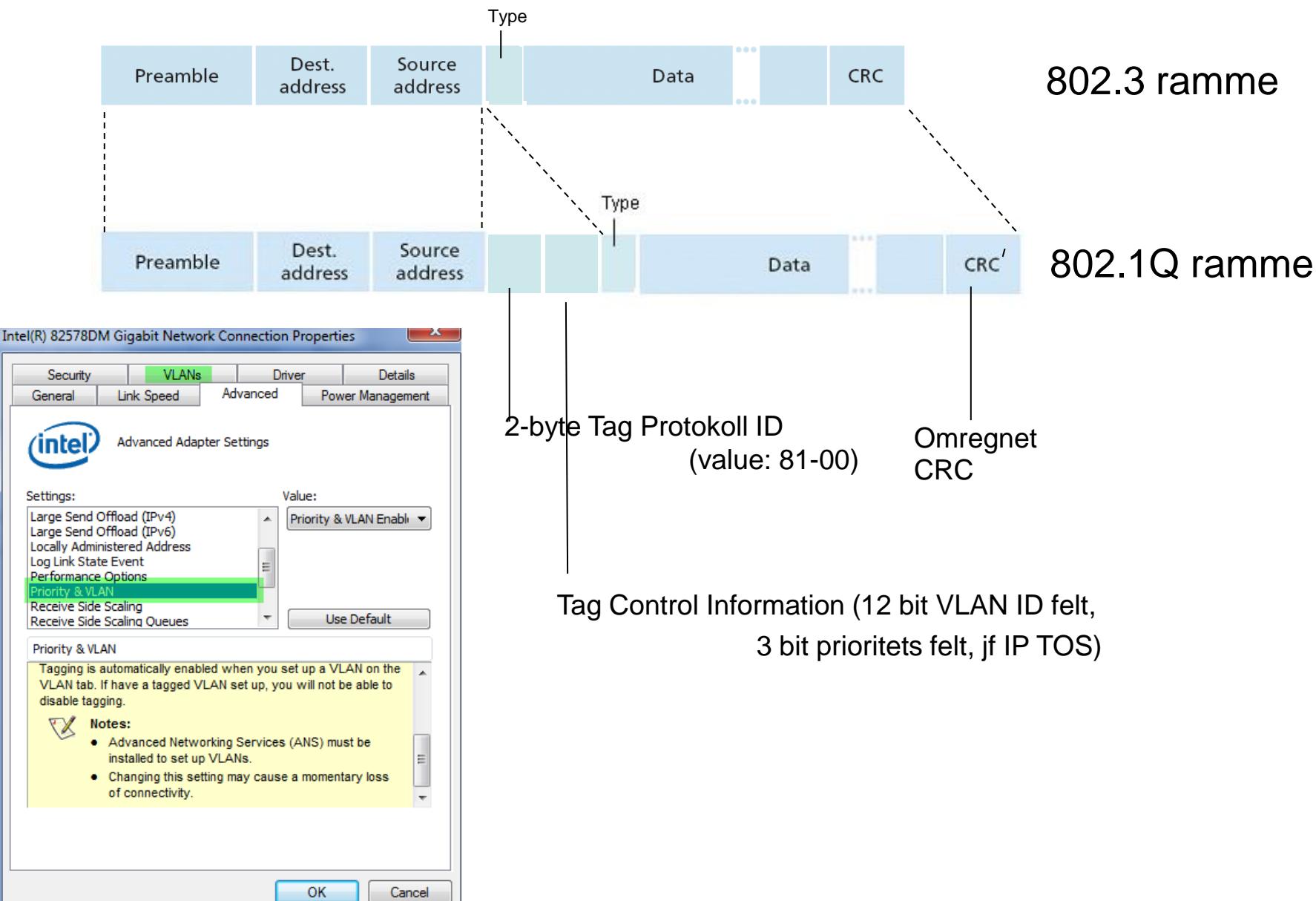
Virtual Local Area Network

Switch(er) som støtter VLAN kan konfigureres til å definere flere **virtuelle** LAN i ett enkelt fysisk LAN.

... fungerer som **flere** virtuelle switcher



802.1Q VLAN ramme format



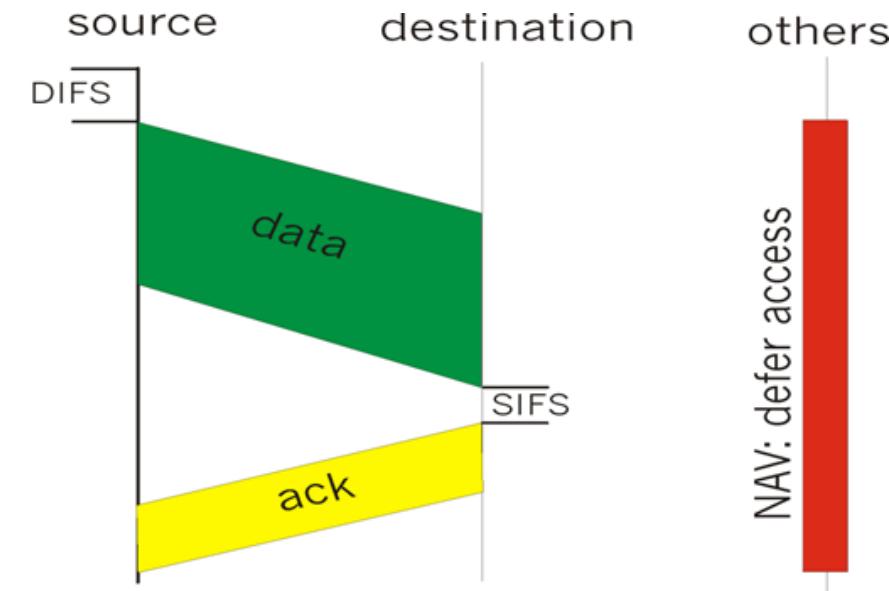
IEEE 802.11 MAC Protocol: CSMA/CA

802.11 CSMA: sender

- if sense channel ledig for **DIFS** sekunder.
then transmit entire frame
(ingen kollisjonsdeteksjon)
- if sense channel busy
then binary backoff

802.11 CSMA receiver

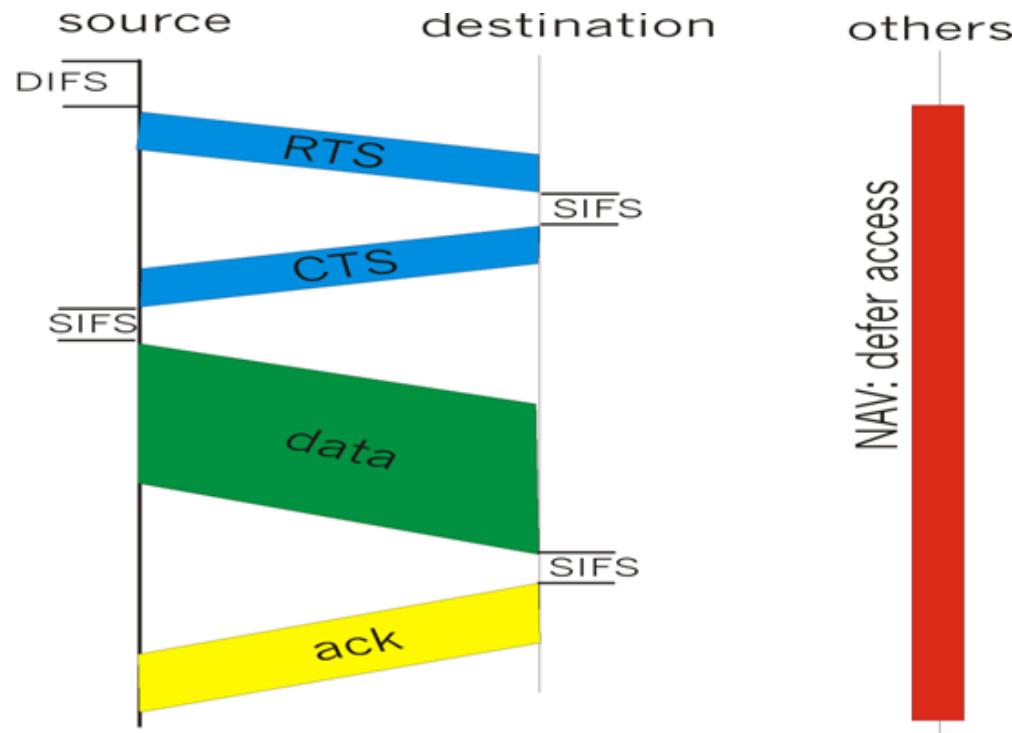
- if received OK
send ACK etter **SIFS**
(ACK nødvendig pga
“skjulte noder problemet”)



DIFS = Distributed Inter Frame Spacing
SIFS = Short Inter Frame Spacing

RTS-CTS utveksling

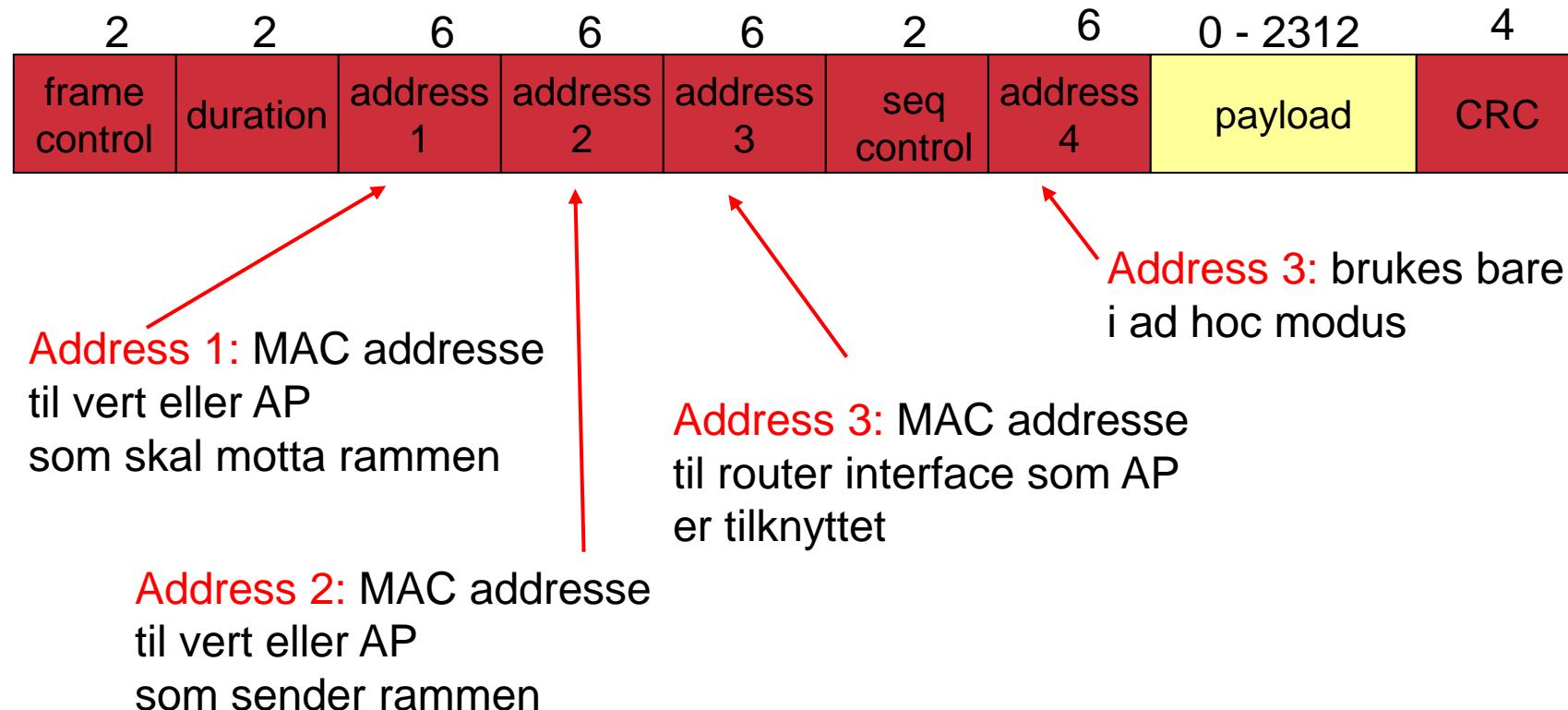
- Sender overfører kort RTS (request to send) pakke: antyder varigheten på overføringen
- Mottager svarer med kort CTS (clear to send) pakke
 - varsler (muligvis skjulte) andre noder
- Skjulte noder avstår da fra å sende i det reserverte tidsrommet: NAV



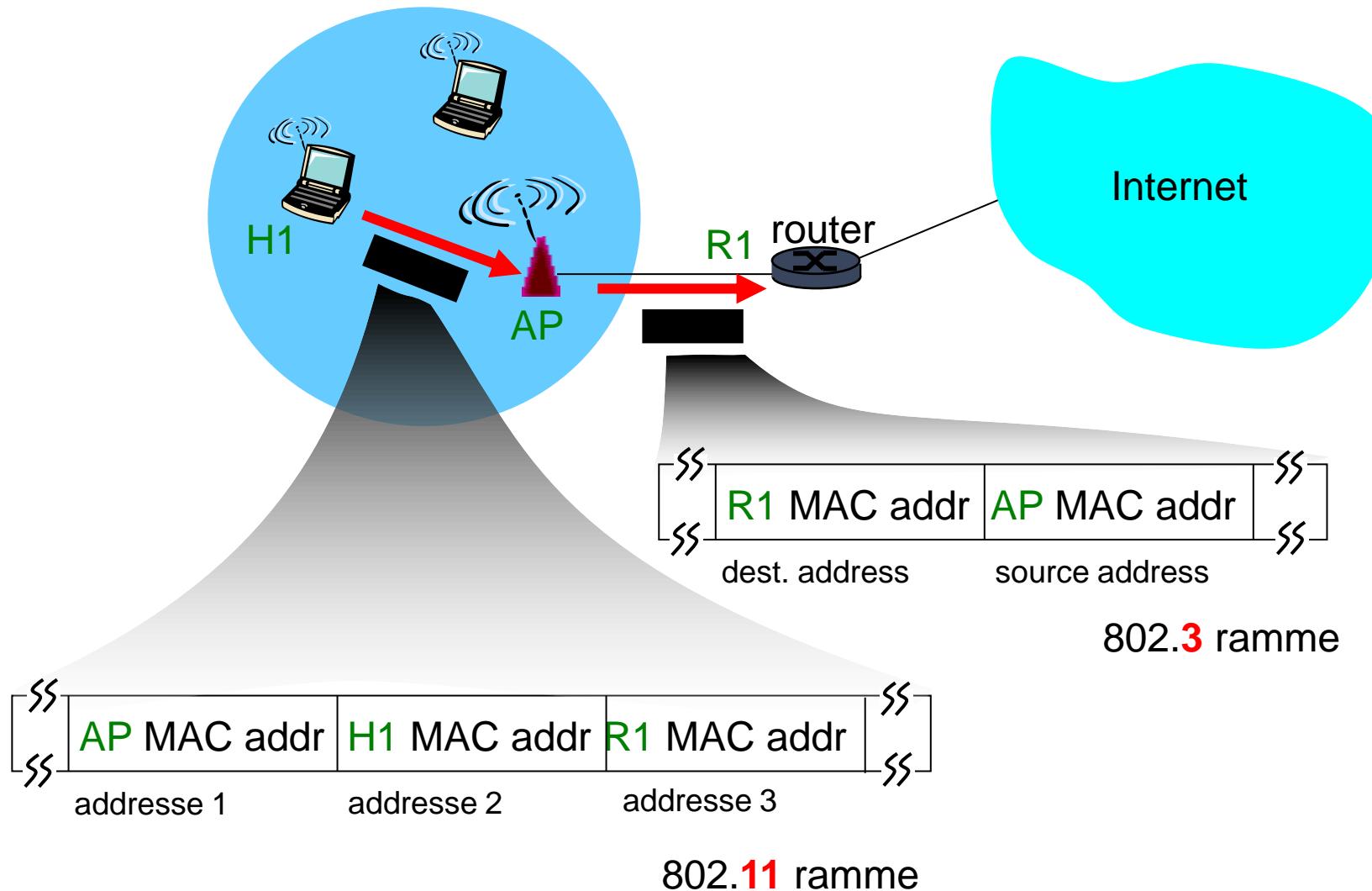
Ulike CA'er

- IEEE 802.11 tillater altså:
 - “ren” CSMA
 - CSMA/CA
 - Reservasjoner
 - Polling fra AP
 - AP tildeler tidsrom til hver node på rundgang (RR)

802.11 ramme: addressing



802.11 ramme: adressering



802.11 ramme: mer

