



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

0xB forelesning:

Repetisjon

I dag

- Om eksamen
- Spørsmål fra studentene – fordel for dere om dagen i dag blir mest mulig interaktiv; jeg kan fortelle hva som er viktig, men vet ikke hva dere syntes var vanskelig 😊

For alle de viktigste emnene se følgende slidedeck:

TK1104_Leksjon_0x0B_Viktige-emner.pptx

Om eksamen

Det er 24 timers frist på denne hjemmeksamen, 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 hjemmeksamen 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.

KILDEHENVISNING

Som dere sikkert har lest om i forskjellige medier de siste årene kan man bli utvist fra universitet/høgskoler for plagiat (juks) på eksamen.

JEG er ikke «slavisk» når det gjelder plagiat (man jeg stryker studenter som åpenbart har sittet sammen og levert en eller flere helt identiske oppgaver)...

Hvis eksamenskontoret tar i bruk standardiserte verktøy for å avdekke plagiat er det viktig at studenter oppgir kilder på alle ressurser de har hentet fra Internett, lærebok og forelesninger (tekst og bilder).

Dere kan bruke hvilken kildehenvisning dere ønsker, men hvis dere spør meg anbefaler jeg APA7: <https://kildekompasset.no/referansestiler/apa-7th/>

- den flyter bedre i tekst enn mange andre standarder

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%)

- Du har tilgang til alle hjelpemidler da det er en hjemmeeksamen, men du må svare selv (ikke lov til å samarbeide med andre) og det er ikke lov til å plagiere (ikke lov til å kopiere tekst fra for eksempel internett).
- Det er en FORDEL (for dere selv) om dere oppgir kilder, følg i så fall APA7 standarden, men jeg er ikke strengt på det formelle. Hvis dere ikke oppgir kilder kan eksamenskontoret sine IT systemer flagge eksamenen som fusk.
- Husk at hvis du blir tatt i å samarbeide med medelever risikerer dere stryk og kanskje også utvisning, ikke sett dere selv, meg og skolen i en situasjon hvor vi mistenker juks.

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 😊

Repetisjon

Tolking av binære koder (noen få)

	0011 1110	0010 0000	0111 0010	0011 0111
(Hexadesimalt)	0x3E	0x20	0x72	0x37
32-bit heltall	1 042 313 783			
16-bit heltall	15 904		29 239	
32 bit flyttall	0.156686			
BCD	Umulig!	20	72	37
IPv4-adresse	62.32.114.55			
ASCII	>	mellomrom	r	7
Scancode(USB)	F5 	3 # 	F23	
UTF-16	桃		爷	
JVM bytekode	istore_3	Istore_2	frem	Istore
X86 opkode	DS: DS	AND	JNO	? AAA

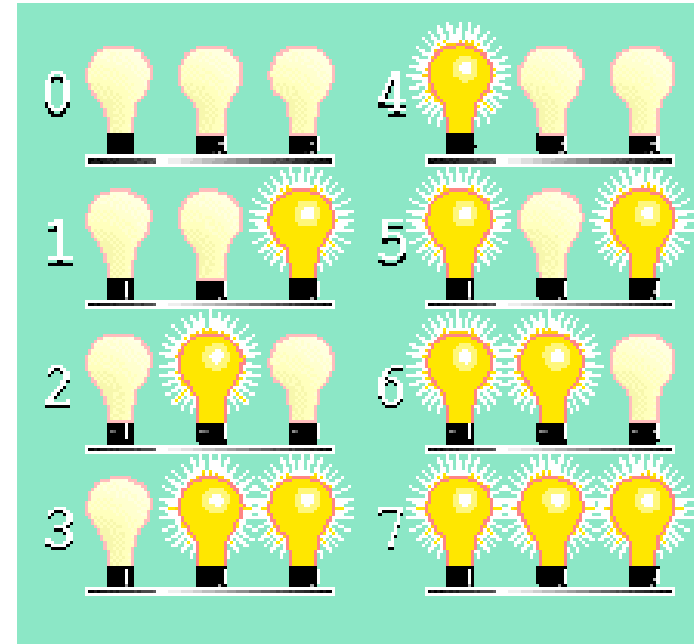
Hvordan vet man hvilken tolkning som er riktig?

• Hvordan vet man hvilken tolkning som er riktig?

- Det bestemmer (bruken i) programmet!

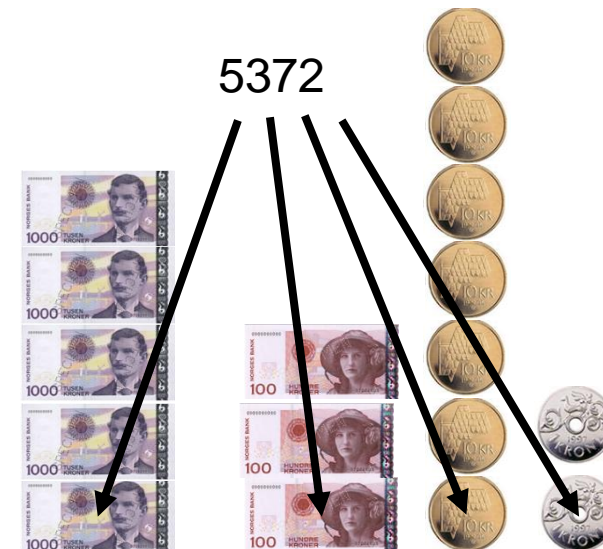
Binær tall-representasjon

- Med 3 lamper kan vi representere $2^3 = 8$ kombinasjoner av av/på
- Med Av=0 og På=1 får vi
 - 0 = 000 4 = 100
 - 1 = 001 5 = 101
 - 2 = 010 6 = 110
 - 3 = 011 7 = 111
- Dette kan utvides til å bruke flere lamper (transistorer), f. eks. 8, 16, 32, 64,



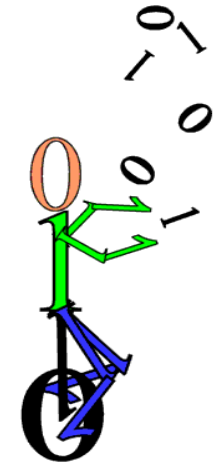
Desimale tall - posisjontall

- Det «vanlige» (**desimale**) tallsystemet bruker **10 sifre**/symboler (0 – 9), mens det **binære** systemet bare bruker **2** sifre/symboler (0-1).
- Prinsippene bak det binære tallsystemet er imidlertid de samme som for det desimale systemet
 - **Posisjonen** til et siffer i et tall **avgjør** hvilken **verdi** sifferet representerer («tyngde»)



Tallsystemer

- Alle **posisjonelle** tallsystemer bruker en **base**
- Det **desimale** tallsystemet har base **10**
- Hver **posisjon** i tallet tilsvarer en **potens** av basen
- I prinsippet kan en bruke en hvilken som helst base
 - $407_{23} = 4 \cdot 23^2 + 0 \cdot 23^1 + 7 \cdot 23^0 = 2116 + 0 + 7 = 2123_{10}$
 - $= 6122_7$
- Det fine med posisjonelle tallsystemer er at fravær av en potens-verdi («vekt») kan representeres med **0**
 - **407** er ikke det samme som 47



Konvertering fra desimal til binær

$$851 = 512 + 339 = 2^9 + 339$$

$$339 = 256 + 83 = 2^8 + 83$$

$$83 = 64 + 19 = 2^6 + 19$$

$$19 = 16 + 3 = 2^4 + 3$$

$$3 = 2 + 1 = 2^1 + 1$$

$$1 = 2^0$$

$$851 = 2^9 + 2^8 + 2^6 + 2^4 + 2^1 + 2^0$$

$$851 = 1 \cdot 2^9 + 1 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

$$851_{10} = 0000\ 0011\ 0101\ 0011_2$$

2^0	=	1
2^1	=	2
2^2	=	4
2^3	=	8
2^4	=	16
2^5	=	32
2^6	=	64
2^7	=	128
2^8	=	256
2^9	=	512

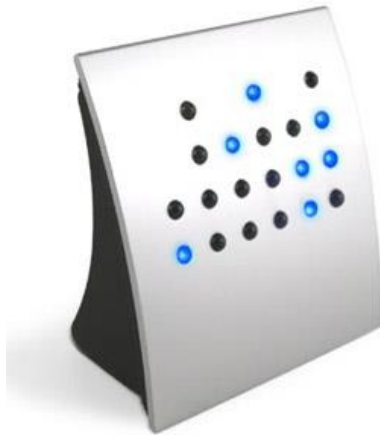
$$2^{10} = 1024$$

Konvertering fra binær til desimal

$$1101010011 = 1*2^9 + 1*2^8 + 0*2^7 + 1*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0$$

$$= 512 + 256 + 64 + 16 + 2 + 1$$

$$= 851$$



512 256 128 64 32 16 8 4 2 1

1101010011

512 256 64 16 2 1

2^0	= 1
2^1	= 2
2^2	= 4
2^3	= 8
2^4	= 16
2^5	= 32
2^6	= 64
2^7	= 128
2^8	= 256
2^9	= 512
2^{10}	= 1024

NB! Her mangler innledende nuller i 16 bit presisjon!

0000 0011 0101 0011

Regneoperasjoner - binærtall

Addisjon

$$\begin{array}{r} 1011 \\ + 0001 1010 \\ \hline 0010 0101 \end{array}$$

Multiplikasjon

$$\begin{array}{r} 10 * 10 \\ \hline 00 \\ 10 \\ \hline 0100 \end{array}$$

NB!

Å doble er dermed det samme som å legge til en null lengst til høyre...

Negative tall = toerkomplement

- Invertering bør ikke resultere i forskjell på +0 og -0
- Bruker 2'er komplement istedenfor 1'er komplement

0001 0011

1110 1100 1's komplement (flip (snu) alle bits)

1110 1101 2's komplement = 1's komplement + 1



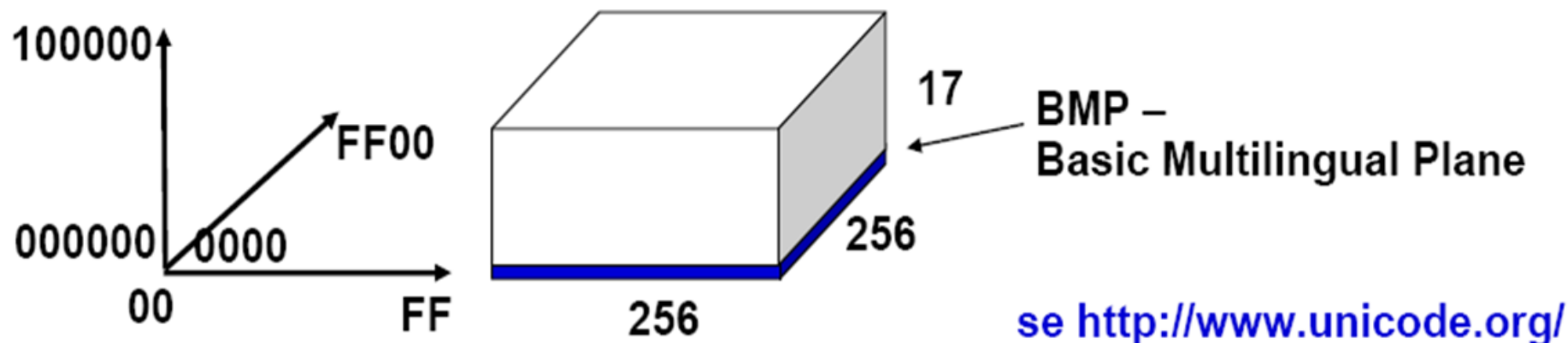
Subtraksjon

- Å trekke fra er dermed **alltid** det samme som å legge til toerkomplementet

46	=	0010 1110	=	0010 1110
-37	=	-0010 0101	=	+1101 1011
<hr/>				<hr/>
9				1 0000 1001
<hr/>				<hr/>
				<hr/>

Overflow (spillsiffer),
Det sløyfer vi!!
Pga 8 bit presisjon

UNICODE Struktur



- Bruker (foreløpig) **21 bit** til kodepunkter
- Delt i 17 plan på 65536 kodepunkter i hvert
 - Plan 0: Basic Multilingual Plane (BMP)
 - Alle tegn vi støter på i «vanlige» språk
 - Plan 1: SMP
 - Historiske språk (f.eks hieroglyfer), musikk (noter), emotikoner ☺, spillkort mm
 - Plan 2: «Uvanlige» kinesiske tegn
 - Plan 14: Spesialtegn for tagging av språk o.l.
 - Plan 15-16: Privat bruk
 - Firmalogoer o.l. kan registereres her.

Unicode UTF-8

- Enslig motorvogn = 0

+ ASCII-kode

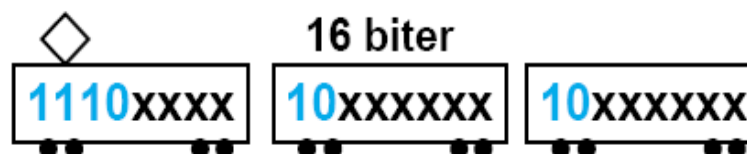


ASCII med en ekstra
ledende 0
er kompatibel med
UTF-8

- Motorvogn i tog
begynner alltid med et
antall 1er-biter
etterfulgt av en 0



- Antall 1er-biter i
motorvognen
= antall vogner i toget



- Vognene begynner
alltid med 10

- Disse bitmønstrene
brukes ikke for
vanlige tegn i UTF-8



Eksempel «Å» i UTF-8

- Tegnet «Å» har Unicode-punkt: U+00**C5**
- Binært: **0000 0000 1100 0101**
- I UTF-8 må vi da legge dette inn i **to byte**:

Mest signifikante (**MSB**)
starter med **110**

Minst signifikante (**LSB**)
starter med **10**

110x xxxx

10xx xxxx

1100 00**11**

1000 0101

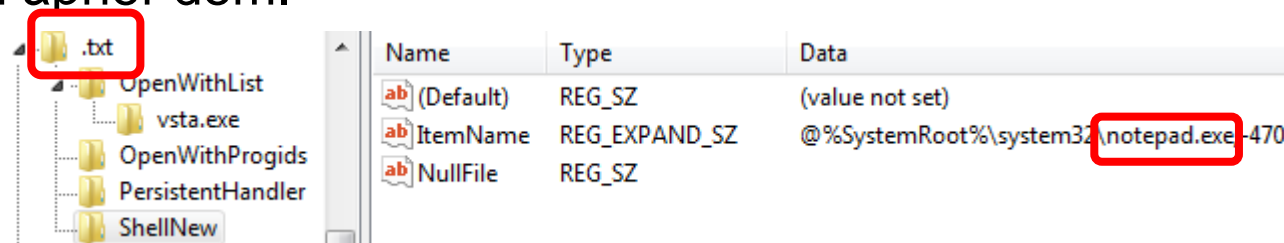
«Padding»

- **UTF-8** koding av U+00C5 er **C385** ...

«Magic Number»

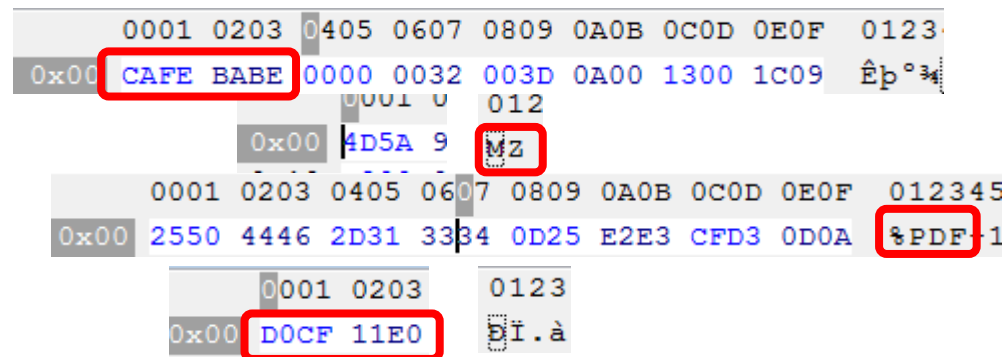
- Mange filformater starter med et «magisk tall» som fungerer som ID for filformatet

- **Windows** bruker også fil-enderelser som IDer og legger dem inn i Registry med info om hvilket program som åpner dem.

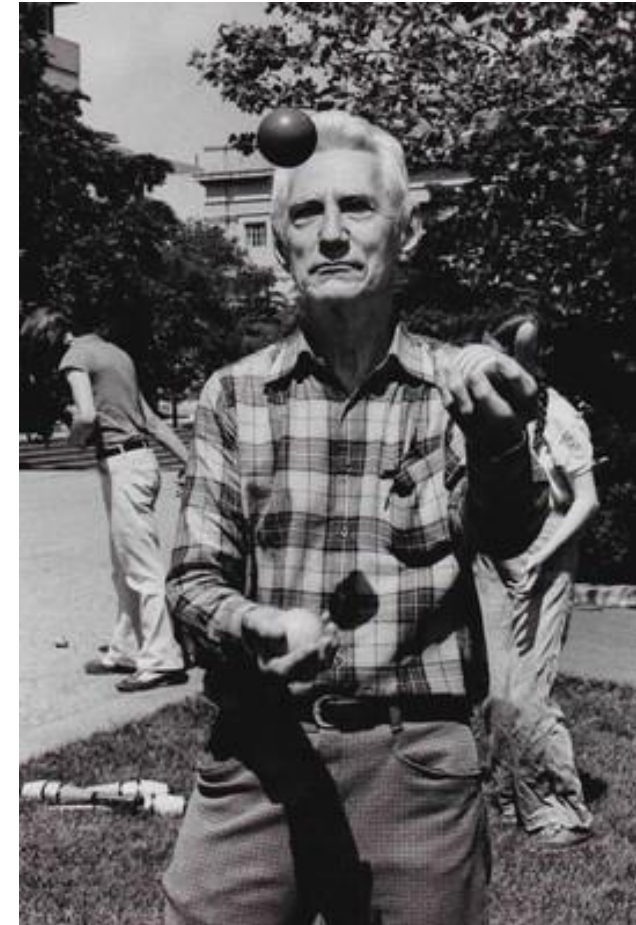


- **OSX** bruker et mer «distribuert» system
 - plister som kan endres med (bl.a.) `defaults` kommandoen
- Noen eksempler på magiske tall i filformat

- Java **.class** filer:
- Windows PE **.exe** filer:
- Adobes PDF filer:
- MS Office (gammelt):

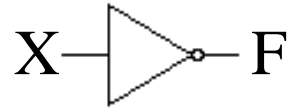


- Består av 3 grunnoperasjoner (porter, gates)
 - IKKE (NOT)
 - OG (AND)
 - ELLER (OR)
 - (EKSKLUSIV ELLER (XOR))
- "Gjenoppdaget" i 1939 av Claude Elwood Shannon
- Data-elektronikk foretrekker oftest «negativ» logikk
 - NAND, NOR, XNOR



NOT

10010110



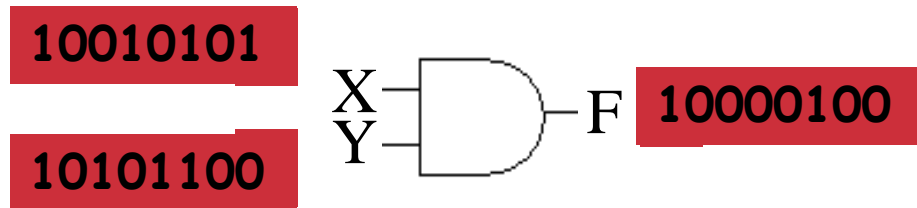
01101001

NOT

Sannhetstabell

X	F
0	1
1	0

AND



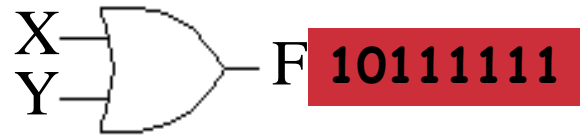
AND

X	Y	F
0	0	0
0	1	0
1	0	0
1	1	1

OR

10111001

10001110



OR

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

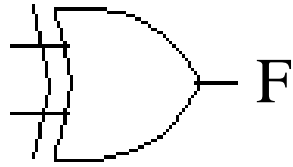
XOR

10110010

X

11001110

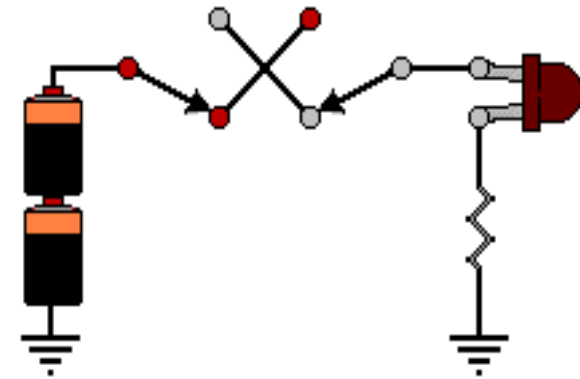
Y



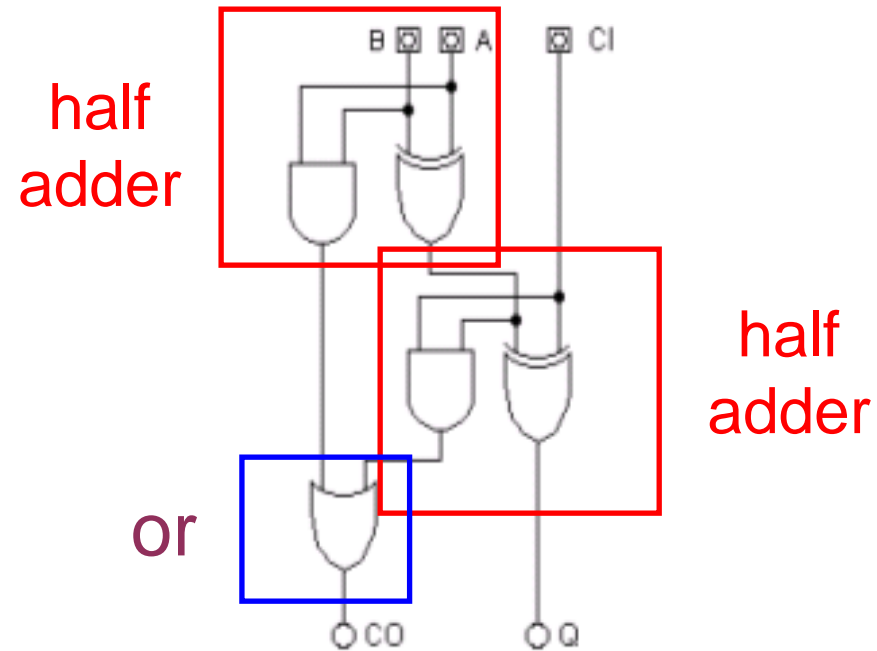
01111100

XOR

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0



Praktisk eksempel: Full adder



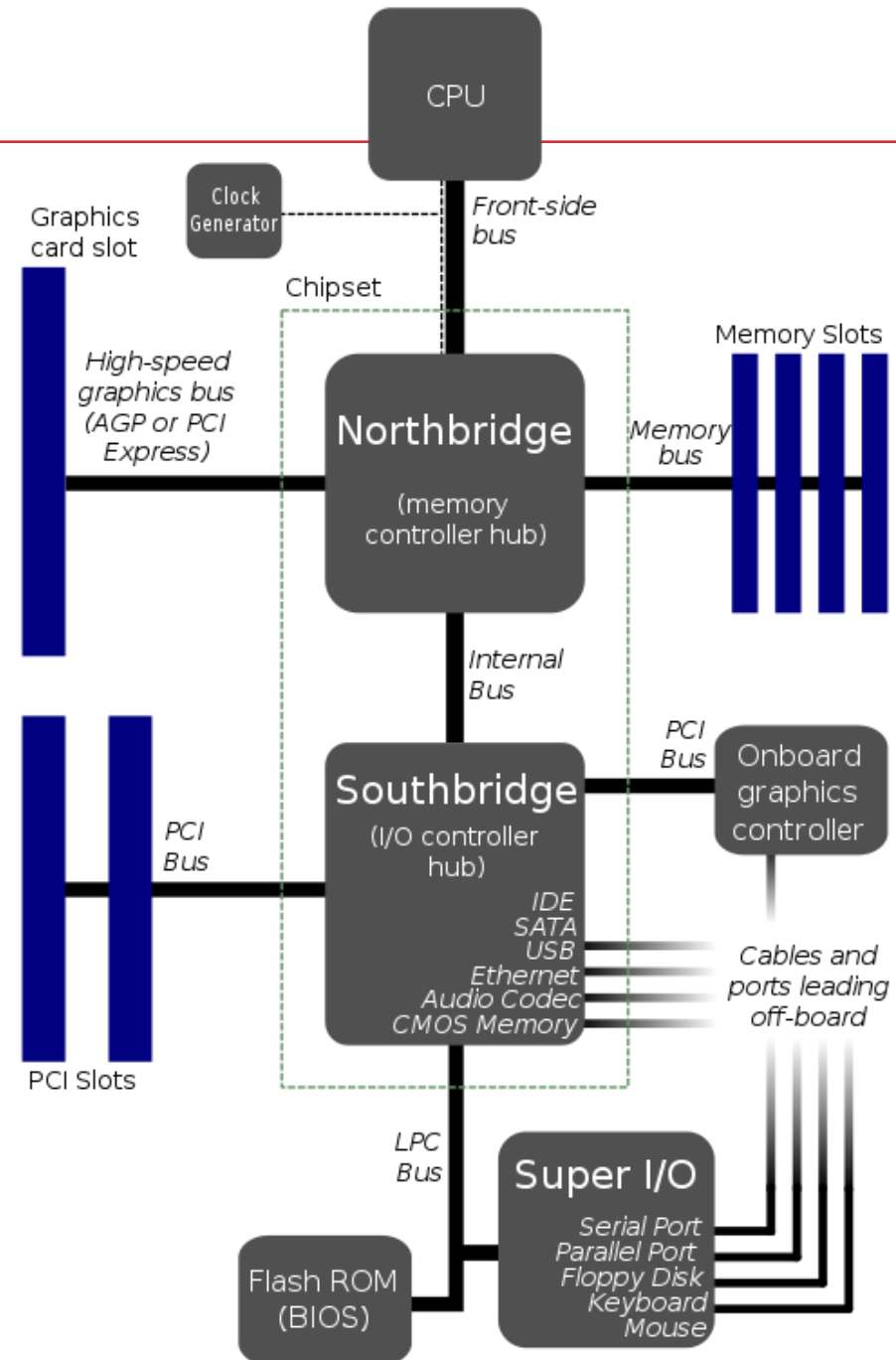
A	B	CI	Q	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- Legger sammen to bit, **A** og **B**, og eventuell mente inn på **CI**
- Får svaret i **Q** med menten ut i **CO**

- Intel og AMD
 - Konstruerer og produserer CPUer, chipset m.m.
 - Fokus: **Desktop**, **server**, mobil, embedded
 - **C**omplex **I**nstruction **S**et **C**omputing
- MIPS og ARM
 - Konstruerer og lenserer CPU-kjerner mm til ulike produsenter
 - Fokus: **Embedded**, **mobil**
 - **R**educed **I**nstruction **S**et **C**omputing

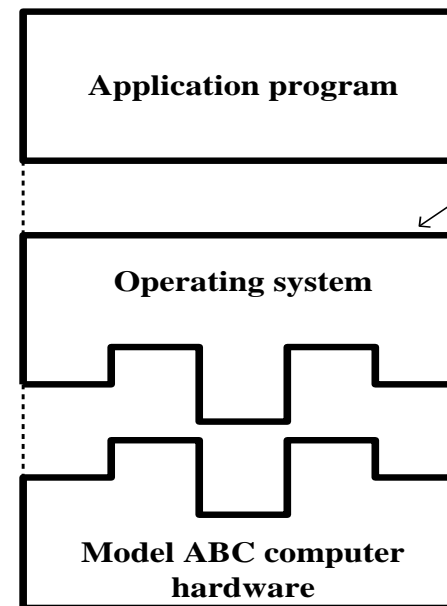
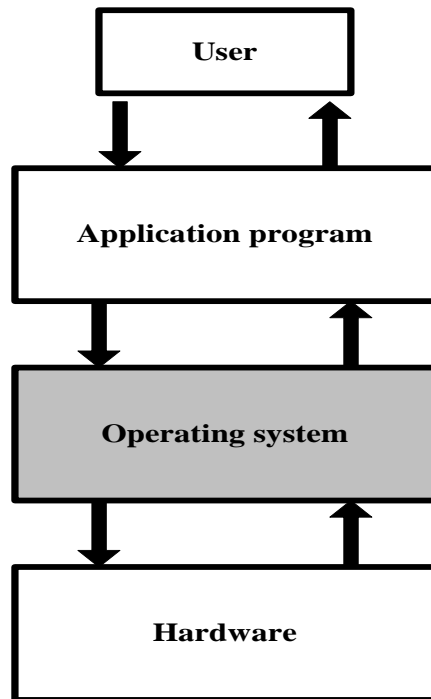
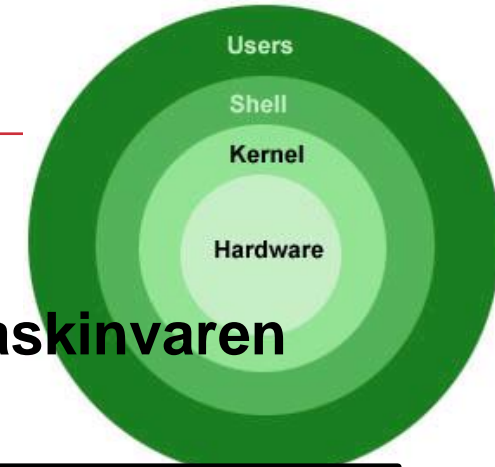
Hovedkort

- Kopler sammen
 - CPU
 - Chipset
 - Busser
 - Minnespor og RAM
 - Expansjons-spor og ekstrakort
 - Porter og «støpsler»

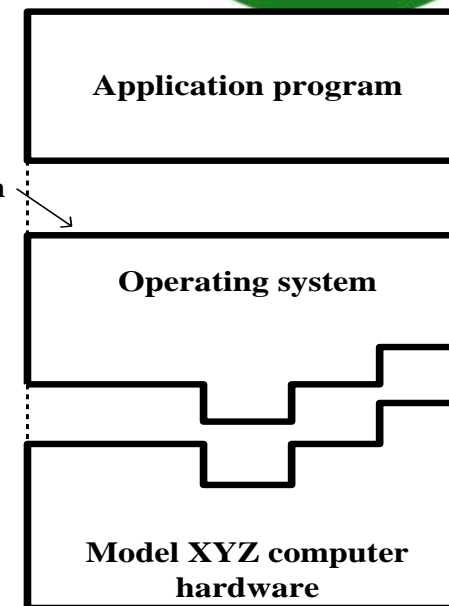


Hva er et operativsystem?

Operativsystemet er programvare som ligger mellom brukeren/programmereren og maskinvaren



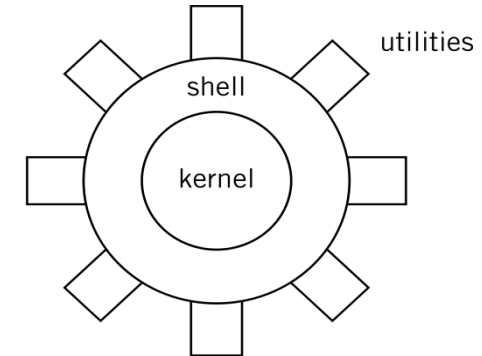
Platform



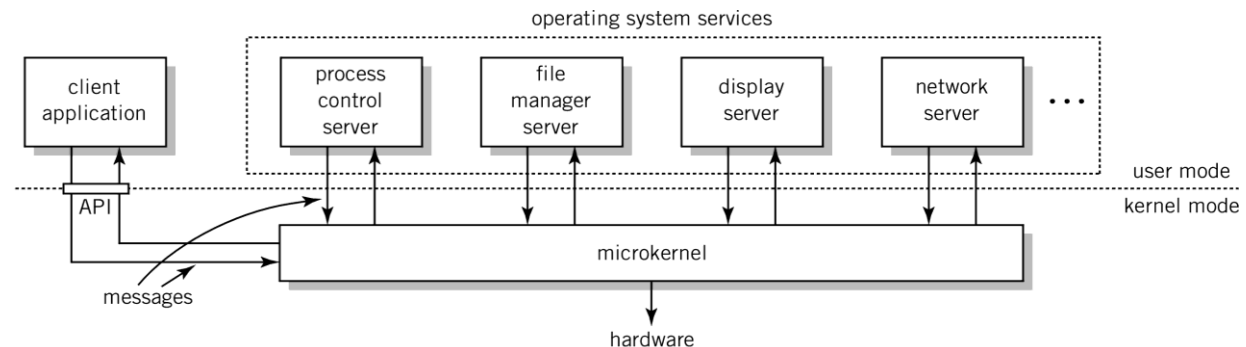
Samme applikasjon på ulike typer maskinvare (OS = “plattform”)

OS Organisering

- Flere mulig tilnærminger, 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, portabelt, ...
 - MACH, L4, Chorus, ...



- 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

- **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, ...

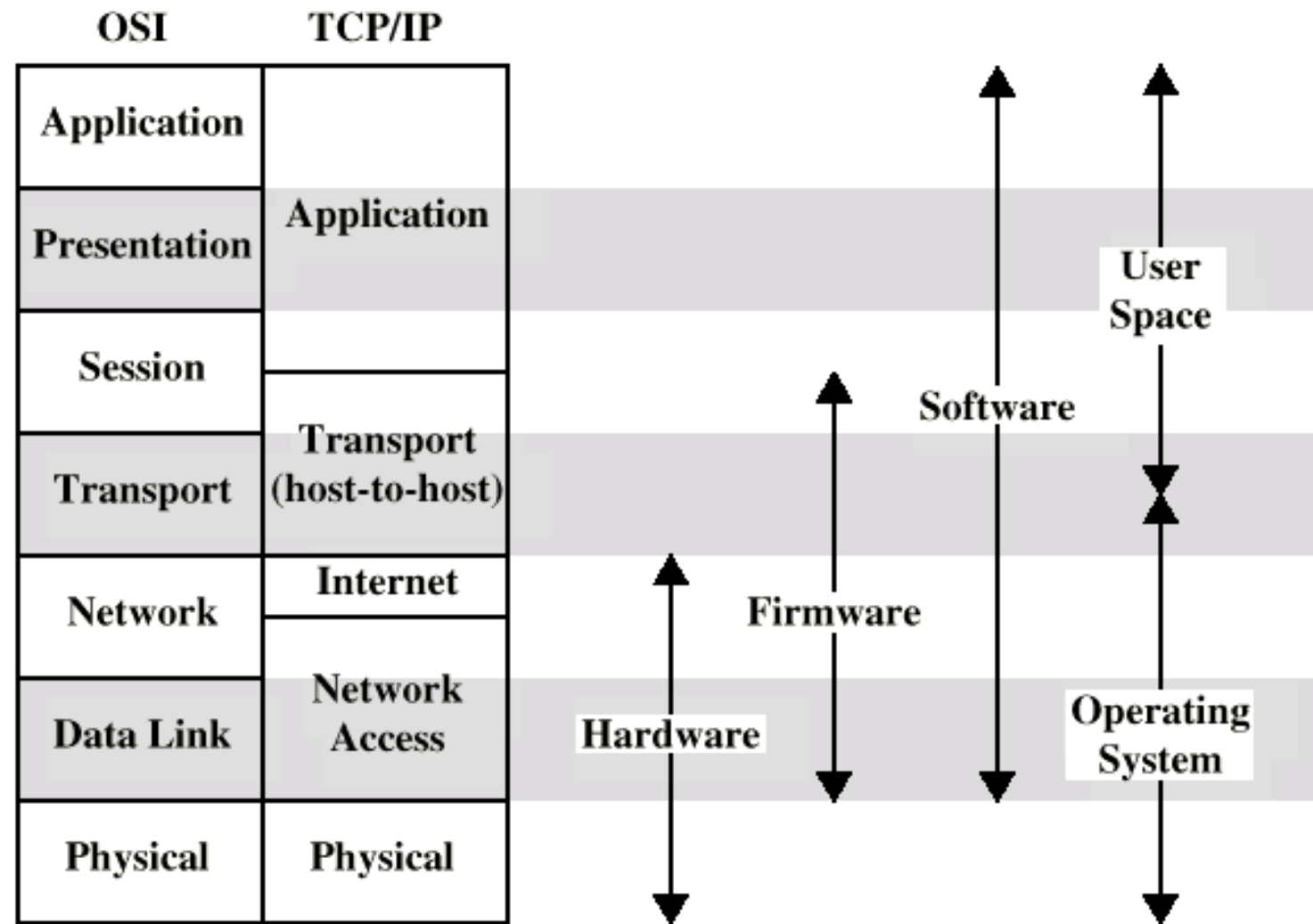
Bruker- vs kjernemodus

- For å oppnå sikkerhet og beskyttelse gir de fleste **CPU**er 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, utstyrsdrivere 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

- 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

OSI vs TCP/IP

- Hvilke modeller (arkitektur) brukes?



Internett historikk: Forspill

1961

- Kleinrock: **Pakkeswitching** som prinsipp

1964

- Baran: Pakkeswitching i militære nett

1967

- ARPAnet (Advanced Research Project Agency) unnfanget

1969

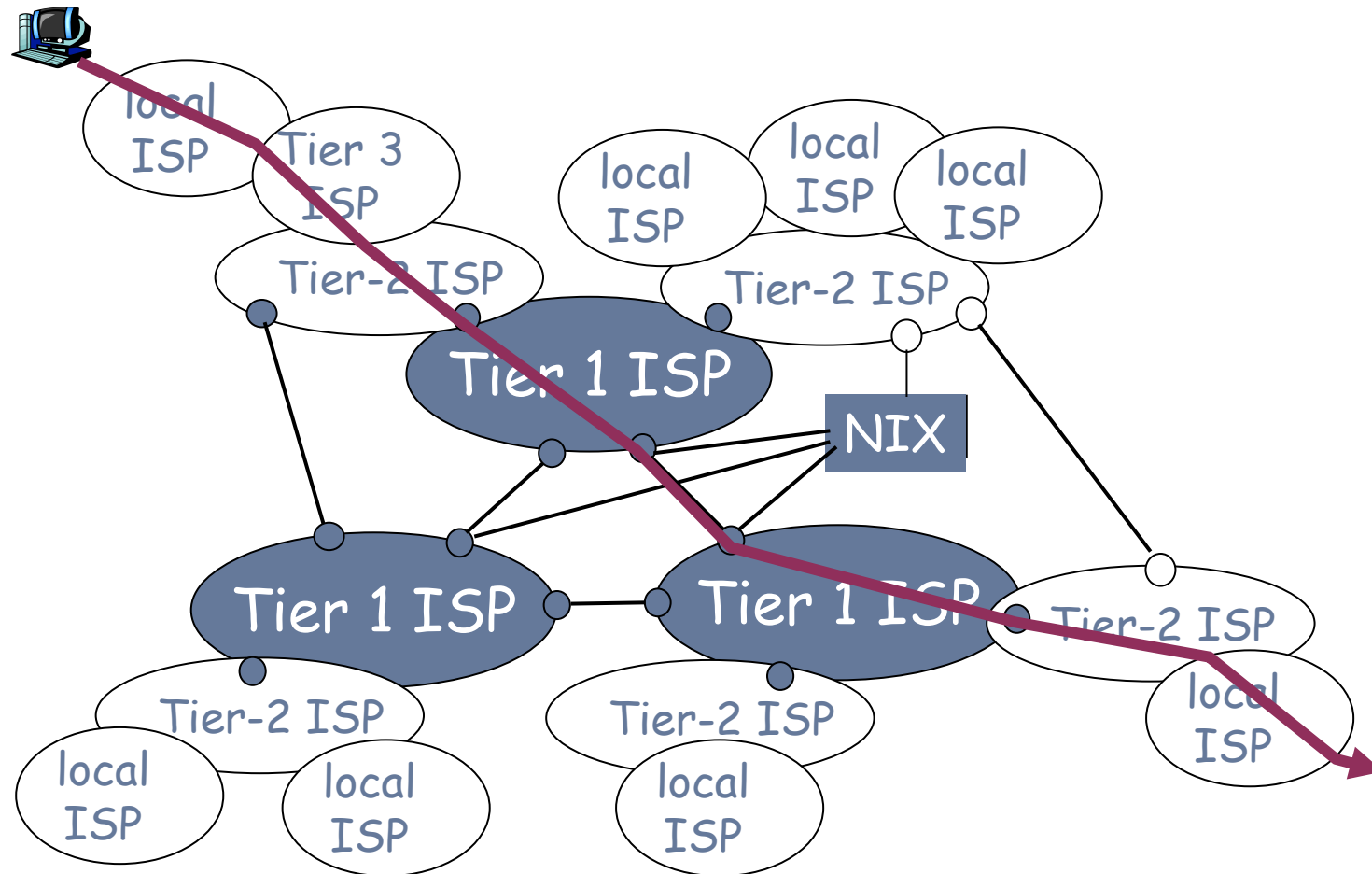
- Første **ARPAnet** node operativ

1972

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

Nettverk av nettverk!

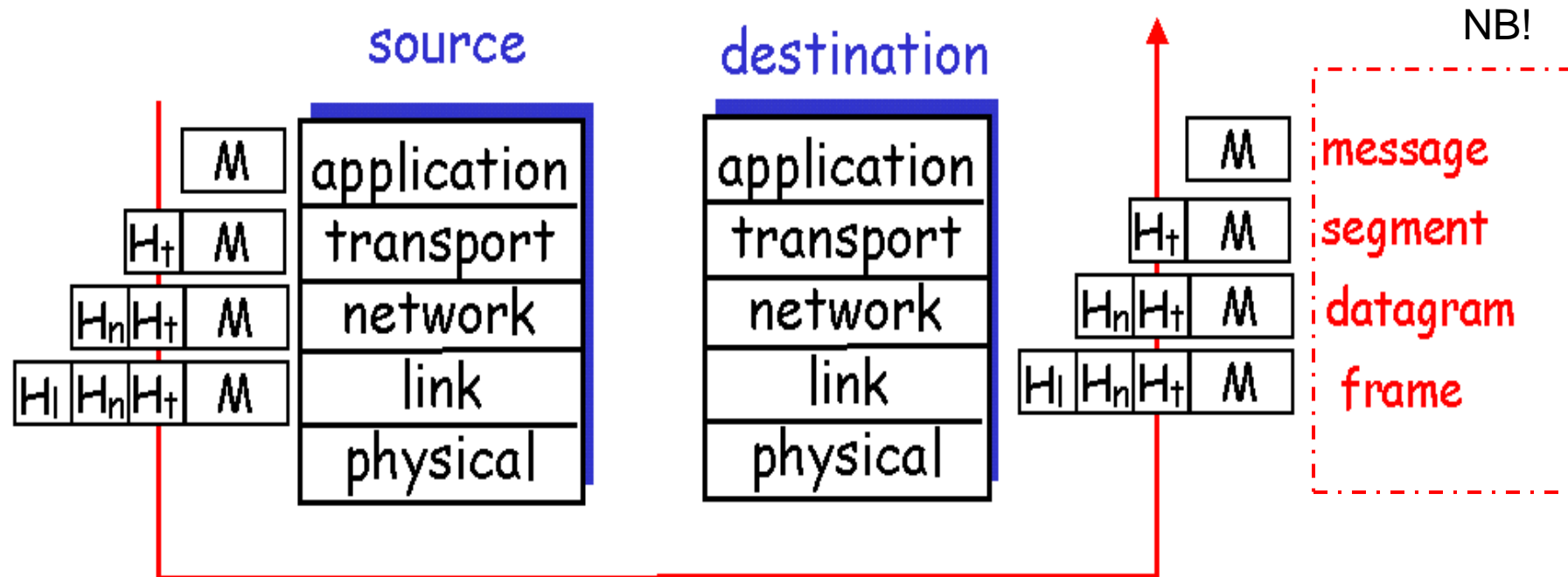
- Internett var opprinnelig laget for å kople sammen ulike typer lokalnettverk
- En datapakke passerer altså (ofte) gjennom mange ulike typer nettverk!



Navn på laget	Betegnelse på overføringsenhet	Viktigste oppgaver/funksjoner Eksempel på protokoller/standards
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, ...

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

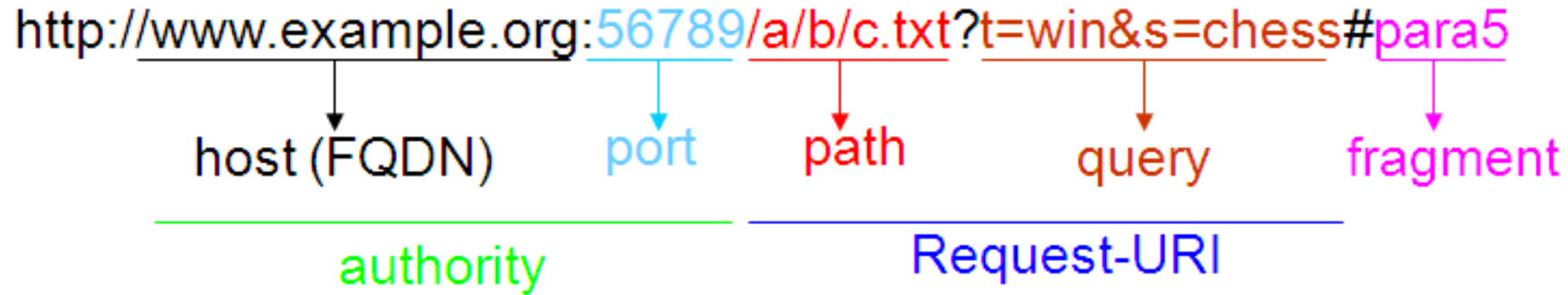


- Internett var designet for et minimum av pålitelighet og med svært lite tanke på sikkerhet
 - Har blitt den viktigste spredningsvektoren for malware
- Nettverk-sikkerhet
 - Hvordan beskytte kommunikasjon mot å bli avlyttet, utnyttet eller "kræsjet"
 - Virus, ormer, trojanske hester, spyware, spam,...
 - Denial of Service Angrep (DOS)
 - Mye mer om dette i TK2100 (etter jul)

- 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://home.nith.no/~blistog/minfil.txt
- Bruker-agenten på web er browseren
 - Netscape, Internet Explorer, Mozilla
- Tjeneren på web kalles web-server
 - Apache, MS IIS



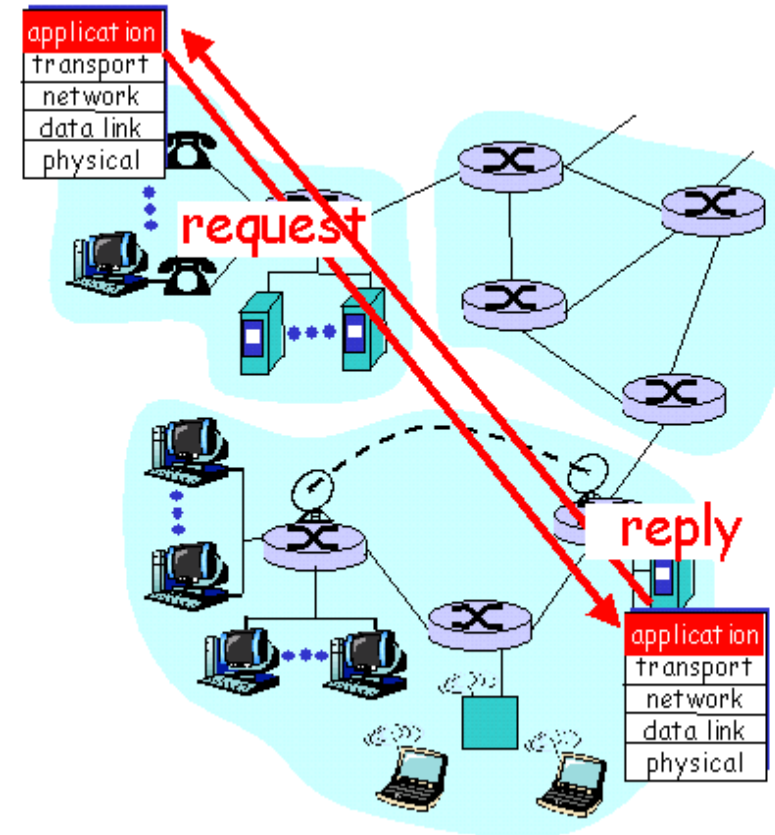
HTTP URL



- 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 **an**ker/posisjon innenfor ressurs ("dokument") (<a href=)

Klient/tjener

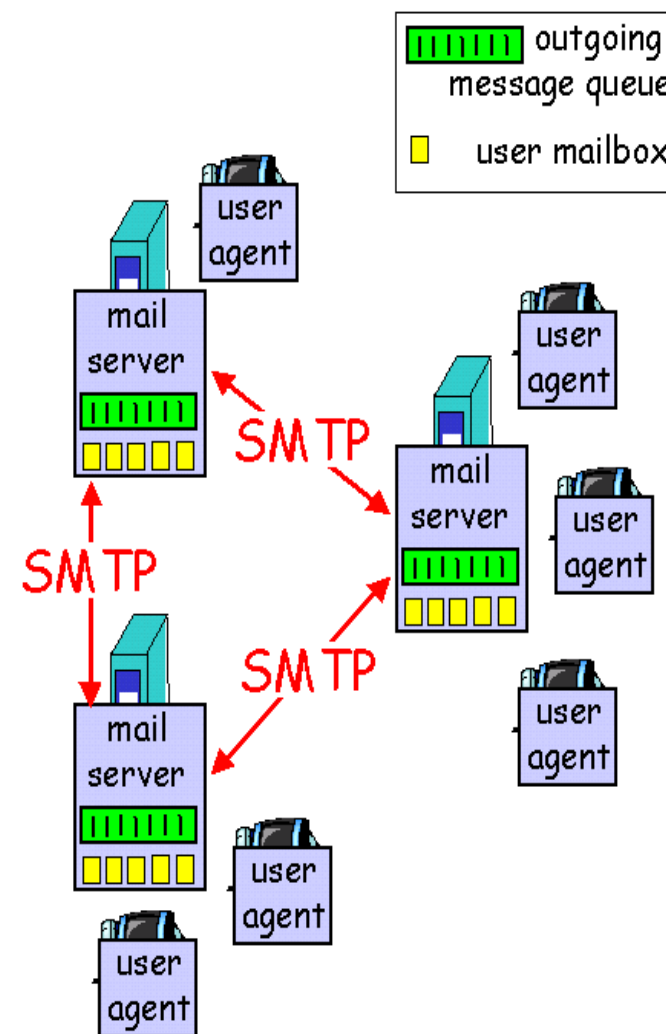
- Typisk oppsett i et nettverk
- **Klient**
 - Tar initiativet
 - Ber om en service fra tjeneren
 - På web er klienten i browseren
- **Tjener**
 - Leverer etterspurt service til klienten
 - Står «alltid på»
 - Har en fast, velkjent adresse
 - Er «flaskehals» fordi alle bruker den samme serveren/server-parken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



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.



Simple Mail Transfer Protocol

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

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»)

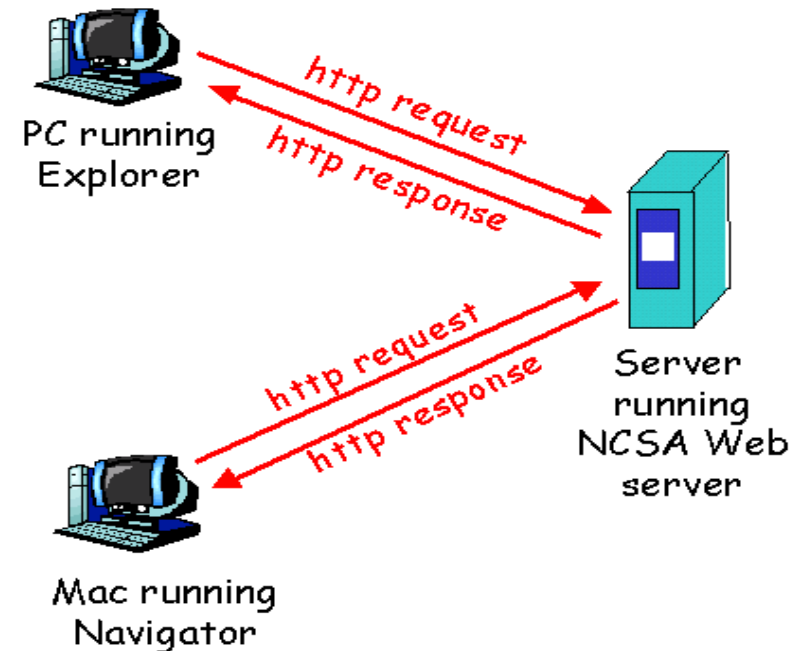
Post format

- 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



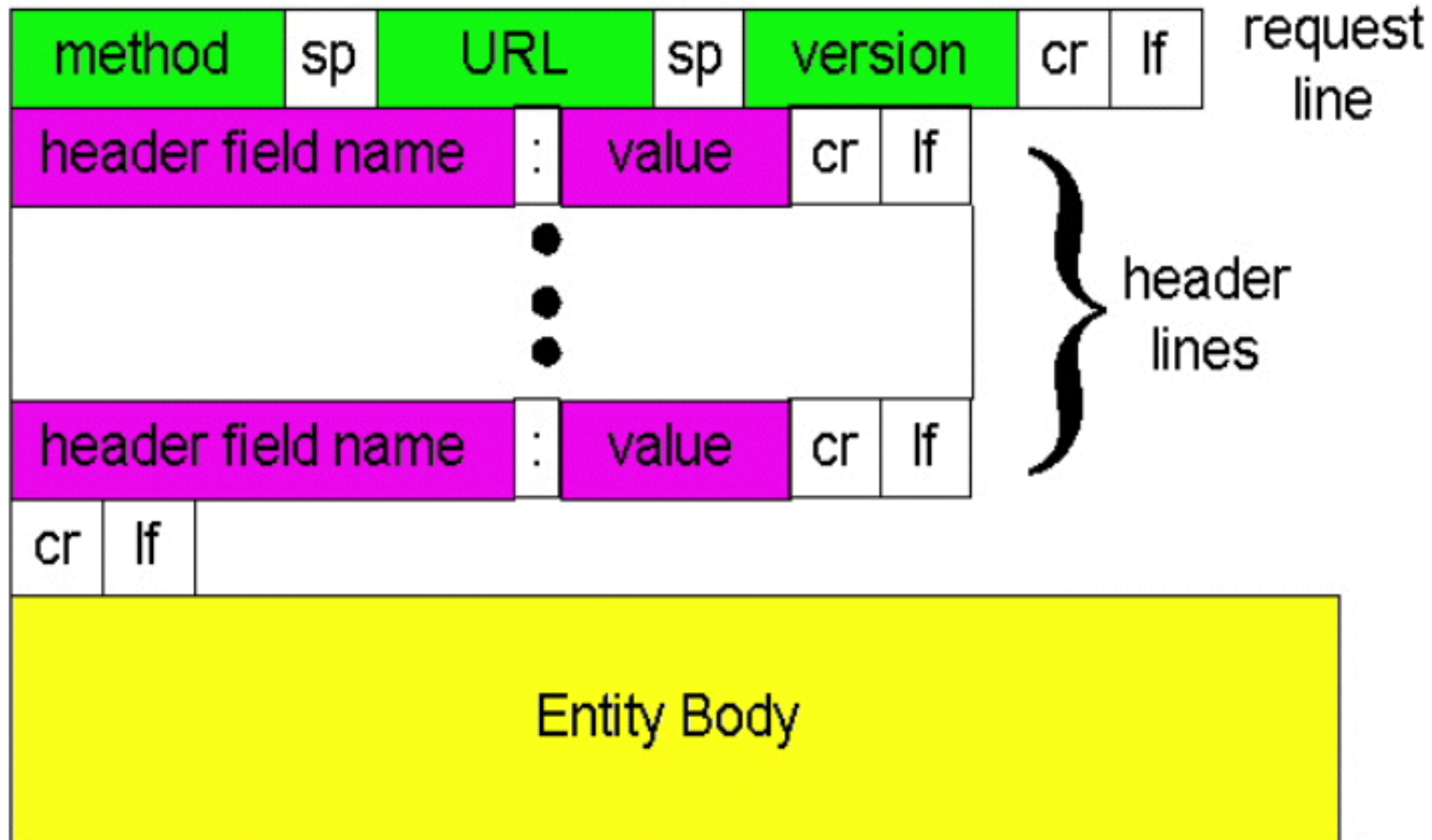
HTTP (HyperText Transfer Protocol)

- 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



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

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

1. Transportlagets tjenester

- **Multipleksing/demultipleksing**
 - Portnummer
- `netstat` (standard verktøy)
- Transport **uten** fast **forbindelse**: **UDP**

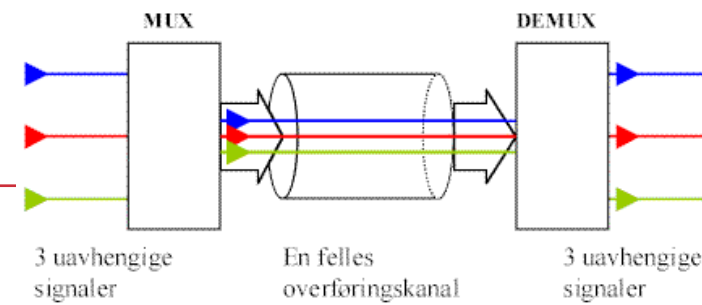
2. **Prinsipper** for pålitelig dataoverføring

3. Transport **med** «fast» **forbindelse**: **TCP**

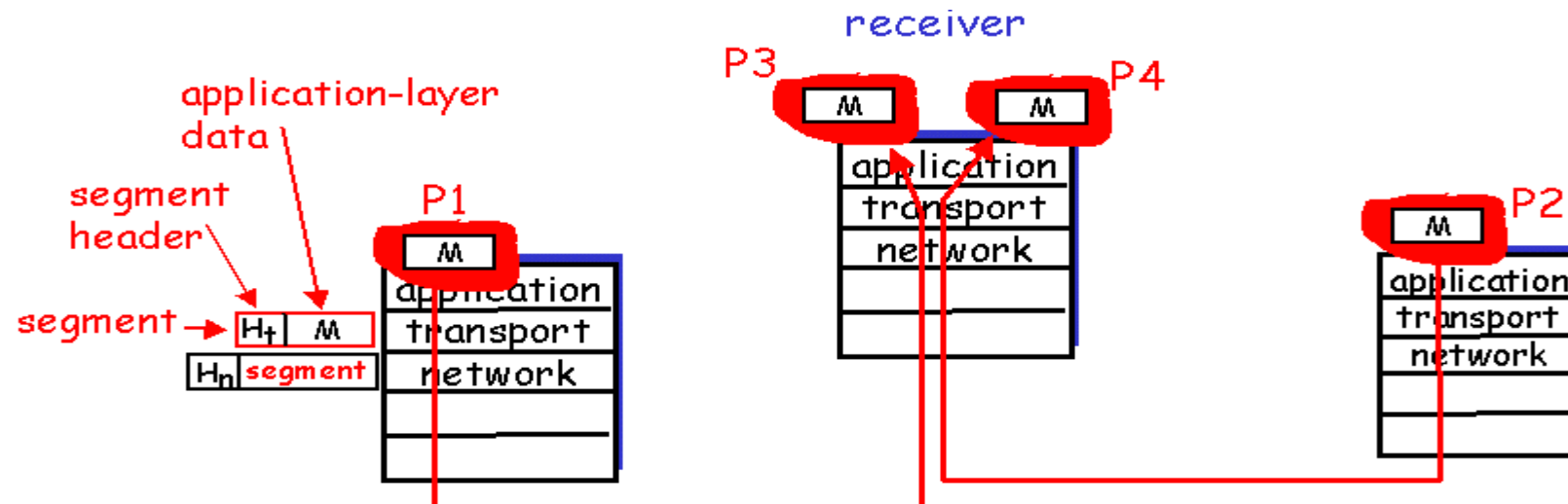
- Pålitelig overføring
- Flyt-kontroll
- Kontroll og styring av forbindelsen

- Internett bruker **nettverks**-protokollen **IP**
 - Gjør så godt den kan, men gir ingen garantier
 - «best effort»
- **UDP**
 - Sender et **datagram**, som kan bestå av flere deler, til mottaker og håper det kommer fra.
 - Forbedrer **IP** bare med **ende-til-ende** kontroll og **feil-sjekking**
- **TCP**
 - Oppretter en "fast" forbindelse
 - Legger inn **flyt-kontroll**,
sekvens-nummer,
kvittering,
tidskontroll,
feilsjekking og
kontroll av **trafikk-kork** (metningskontroll)

Multipleksing/ demultipleksing

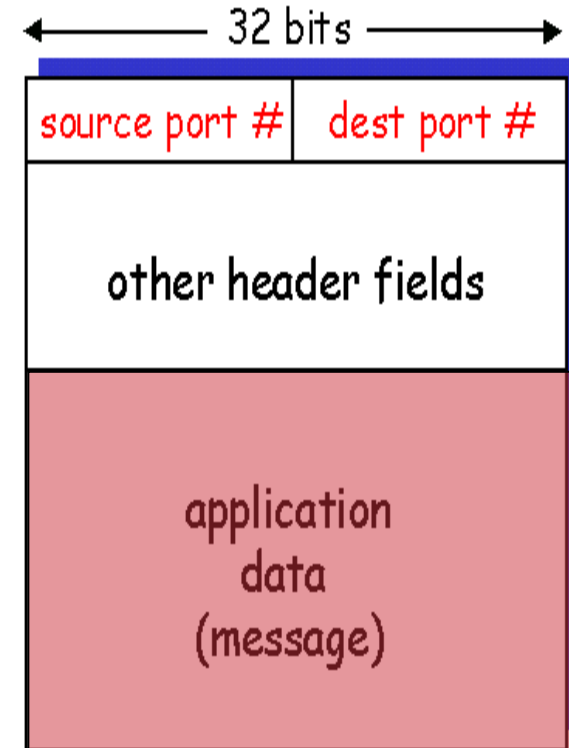


- **Segment**
 - Data-enhet som utveksles mellom transportlagene
 - TPDU (Transport Protocol Data Unit)
- **Demultipleksing**
 - Levere motatte segmenter til riktig **prosess**



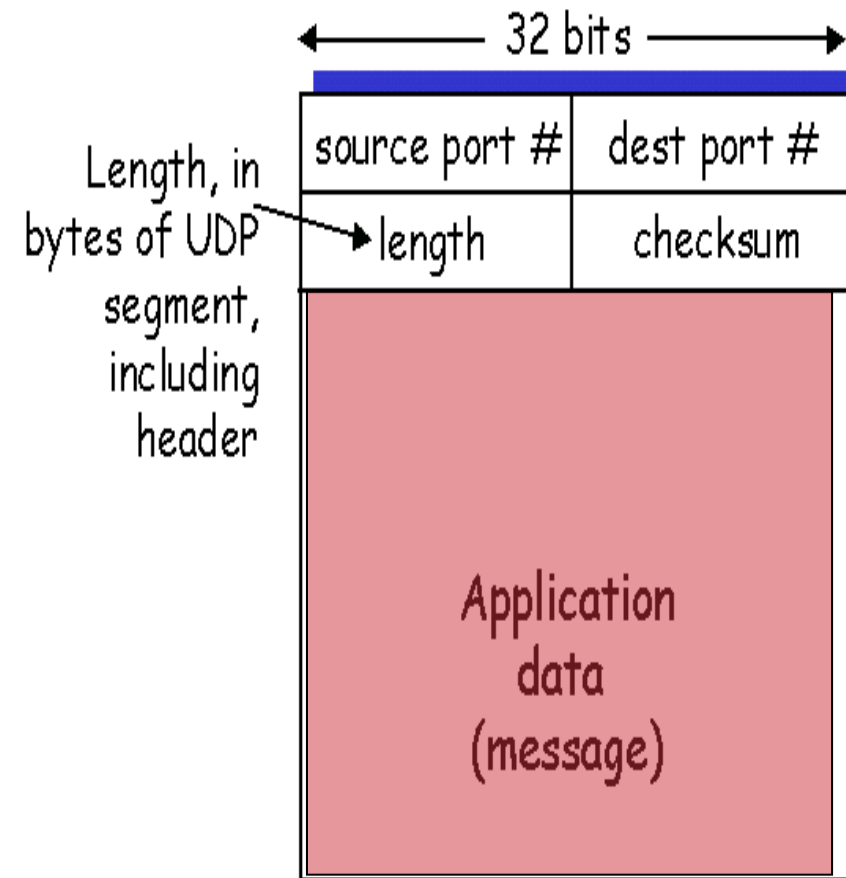
Multipleksing <- portnummer

- Samler data fra applikasjons-prosesser og pakker disse med et hode (header)
- Hodet inneholder **senders** og **mottakers portnummer**
- Portnummer = **16 bit** unsigned **heltall**
- Portene **0-1023** er «**well known**» (RFC 1700)
 - Secure Shell: port **22**
 - SMTP: port **25**
 - DNS: port **53**
 - HTTP: port **80**
 - HTTP over TLS/SSL: port **443**
- Andre porter deles opp i:
 - **Registrerte**
 - 1024-49151 (0x0400-0xBFFF)
 - Kan brukes til annet også, men er **registrert** for en tjeneste hos IANA
 - **Private/Dynamiske**:
 - 49152-65535 (0xC000-0xFFFF)



TCP/UDP segment format

- Brukes ofte i forbindelse med **multimedia** hvor den menneskelige hjerne kan korrigere feilene
- Andre bruksområder
 - DNS
 - SNMP, ICMP
- Mottakerens **applikasjon** kan besørge feil-håndtering



UDP segment format

- Avsender
 - Oppfatter segmentet som sammensatt av 16 bits ord
 - Summerer alle ordene
 - Tar 1's komplement av summen (flipper)
 - Setter sjekksummen inn i headeren på segmentet
- Mottaker
 - Summerer alle 16 bits ordene i mottatt segment, inkl. sjekksummen
 - dersom sum = 1111 1111 1111 1111 => alt OK
- I beste fall gir dette bare en indikasjon på om feil er oppstått under overføringen

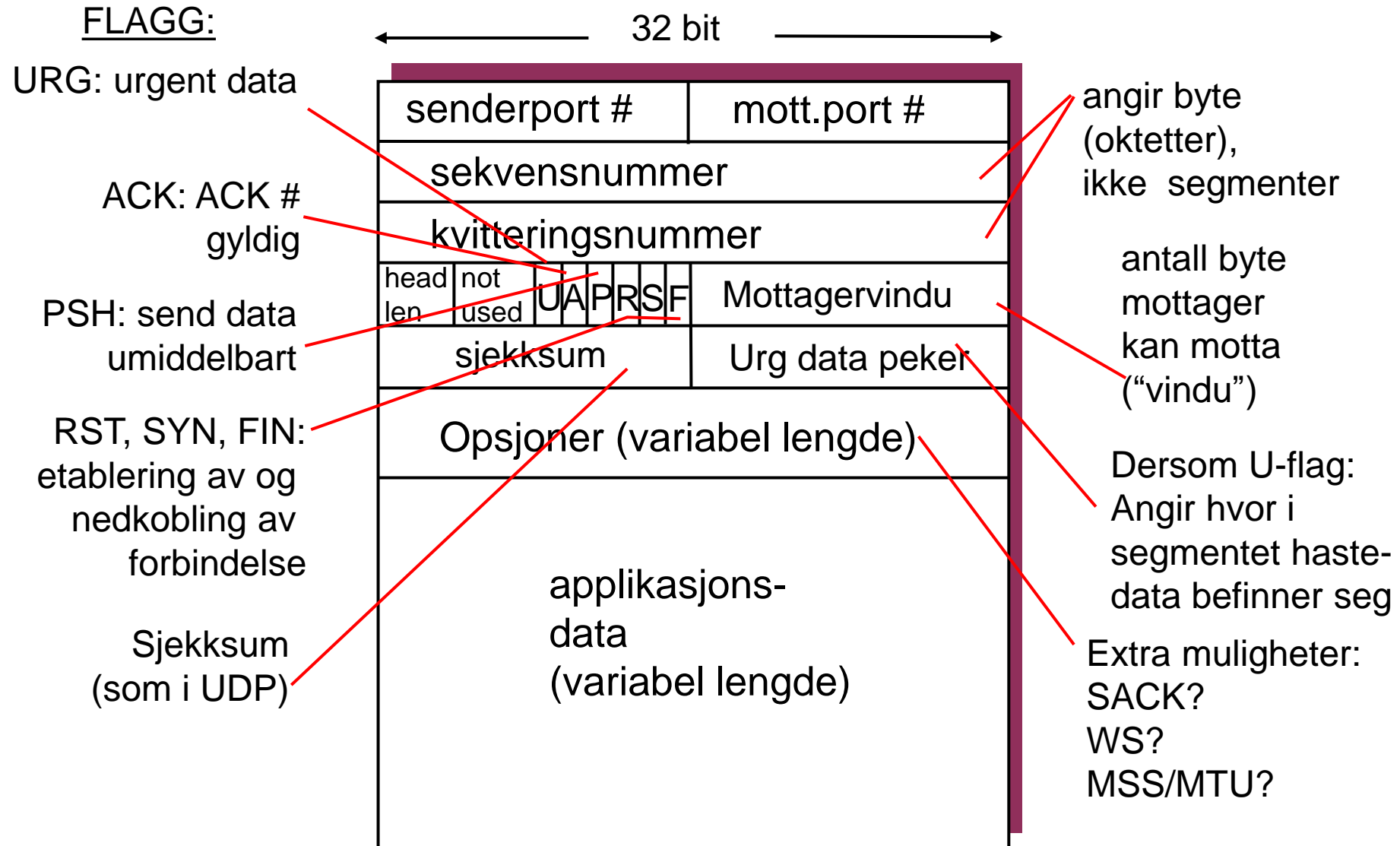
Ex: Internet sjekksummen

Merk: Mente i mest signifikante posisjon legges til LSb (Minst signifikante bit) !

Ex: To 16 bit deler av samlet pakke legges sammen

	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
<hr/>																
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1
<hr/>																
sum	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Oppbygging av TCP-header



TCP: Oppstart av forbindelsen

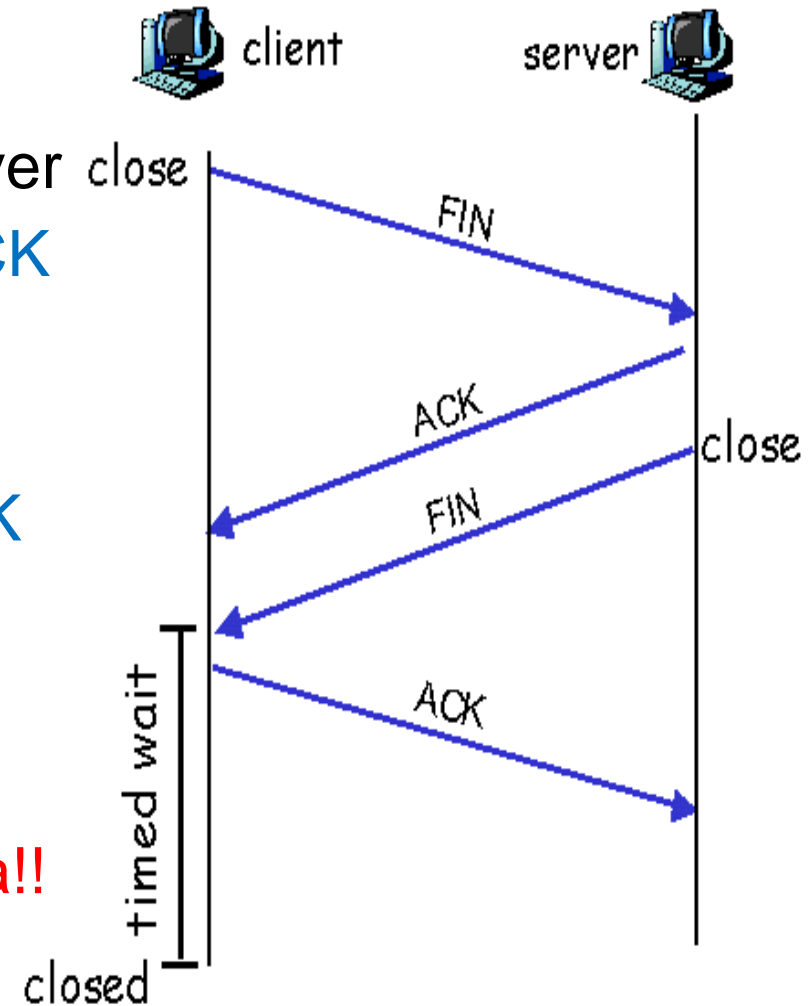
- Sender og mottaker etablerer en forbindelse før data-segmenter utveksles
 - Initialiserer TCP-variable
 - Sekvens-nummer, buffere, vinduer.....
- Klient -> avsender -> mottaker -> server
 - Setter opp socket
- Klient sender et spesielt TCP-segment med **SYN**
 - SYN-flagget i headeren satt
 - Spesifiserer start sekvens-nummer
- Server svarer med **SYN ACK**
 - SYN og ACK-flaggene i headeren satt
 - Setter opp start sekvens-nummer, buffere, vinduer mm

Nedkobling av forbindelsen

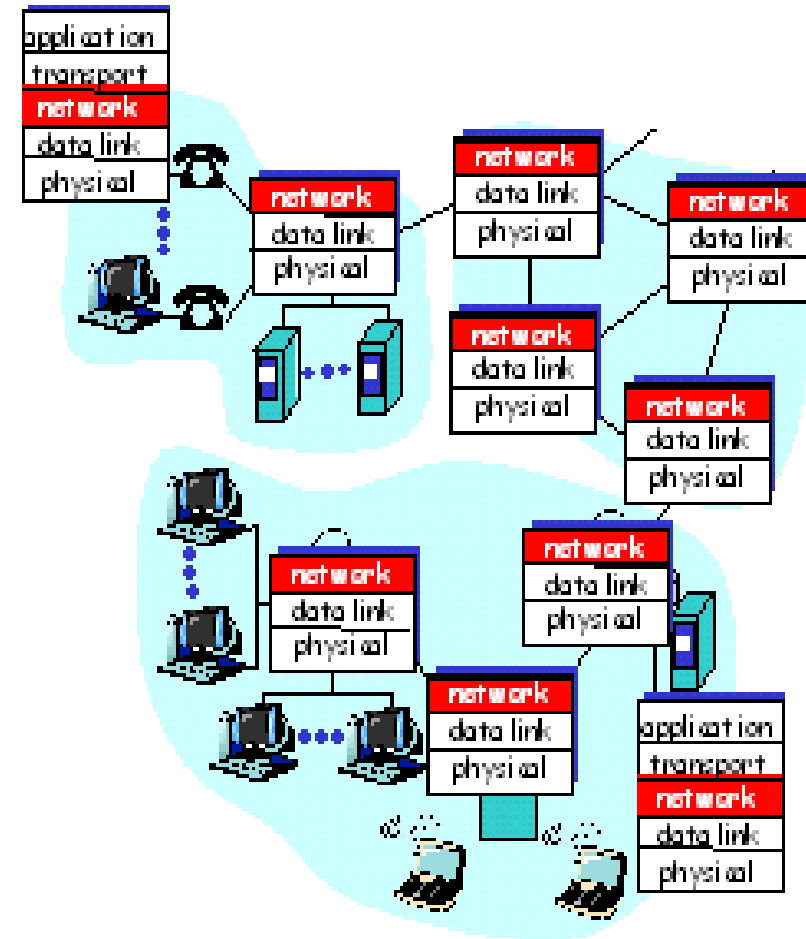
- Klient-app lukker socket
- Klient-OS sender TCP FIN til server
- Server-OS mottar FIN, sender ACK
- Server-app lukker socket
- Server-OS sender FIN til klient
- Klient-OS mottar FIN, sender ACK
- Server-OS mottar ACK
- Forbindelsen avsluttet

- **NB! Andre metoder benyttes også!!**

- F. eks RESET-flagget (fra Server)
- Three Way: FIN, FIN+ACK, ACK

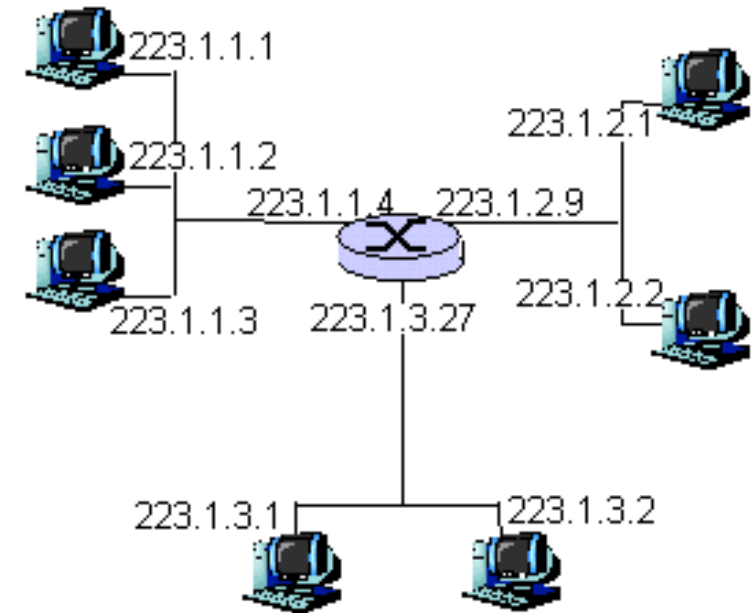


- Flytter pakker fra avsender til mottaker
- Nettverks-protokoll også på hver mellomlanding
- Routing fra avsender til mottaker
- Switching av pakker fra routers input-side til routers output-side
- Hvis nødvendig defineres router kall oppsett for hele ruten før pakke sendes



IPv4 adressering

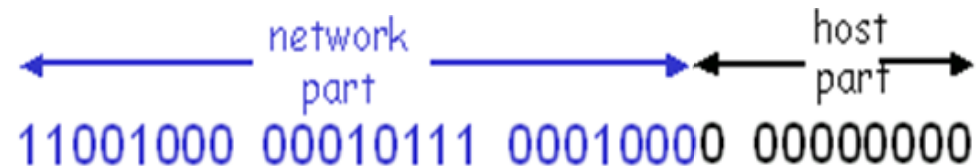
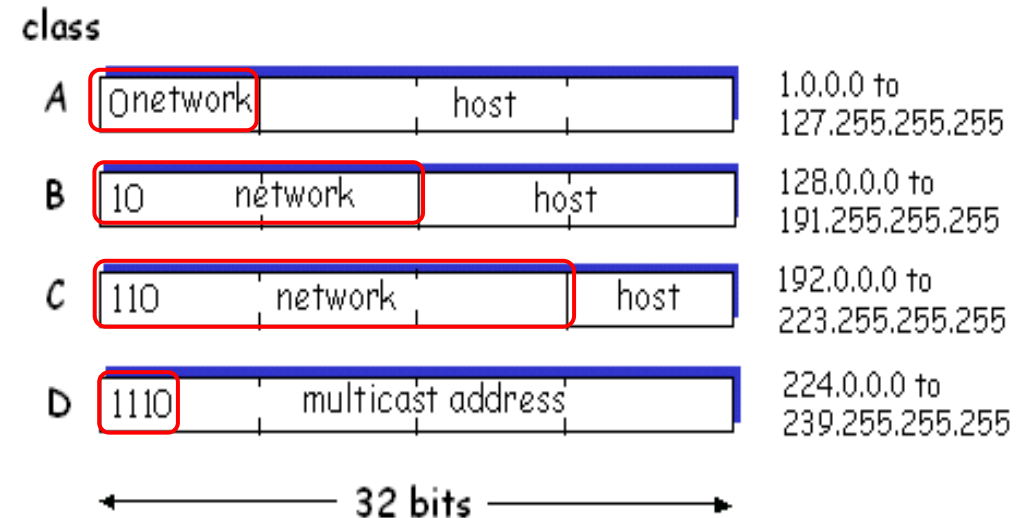
- IPv4 adresse: **32-bit** «id» for hver vertsmaskin og router **interface** (adapter)
- En vertsmaskin kan ha flere interface
- En router har vanligvis flere forbindelser, med hver sin interface
- **IP-adresse hører til hvert interface**



223.1.1.1 = $\underbrace{11011111}_{223} \underbrace{00000001}_1 \underbrace{00000001}_1 \underbrace{00000001}_1$

IP adresser: klasser og CIDR

- 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 adresser, 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



200.23.16.0 /23

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 nettverksparamerte (f.eks. DNS-tjener) + DHCP ack
- **Vertsmaskin settes opp med disse verdiene**

```
C:\Users\blistog>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 . :  
Link-local IPv6 Address . . . . . : fe80::50e5:4  
Default Gateway . . . . . :
```

```
C:\Users\blistog>ipconfig /renew
```

```
Windows IP Configuration
```

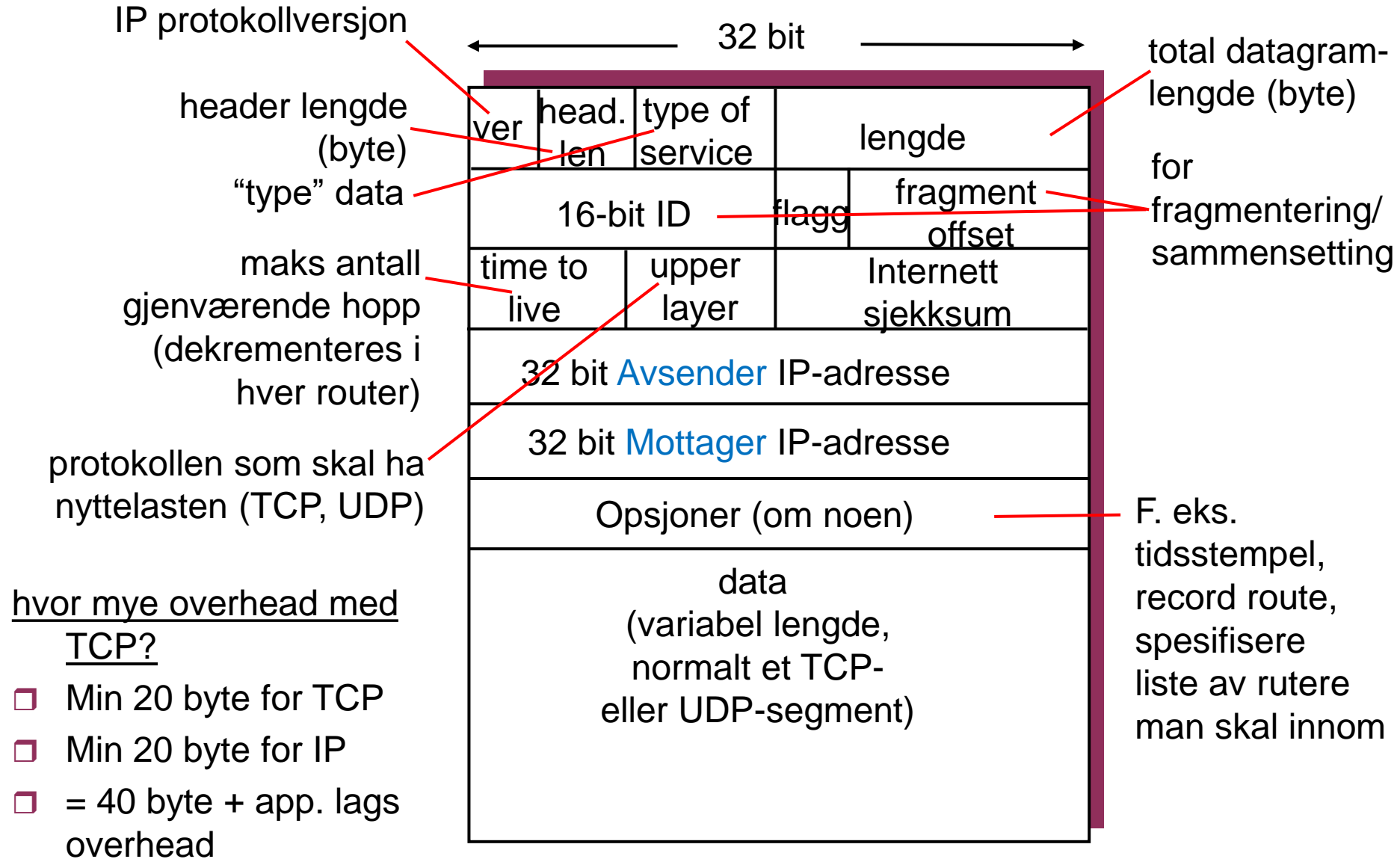
```
No operation can be performed on Bluetooth Network Connection while media disconnected.
```

```
Wireless LAN adapter Wireless Network Connection:
```

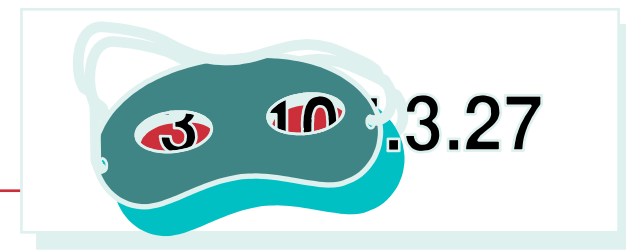
```
Connection-specific DNS Suffix . : ad.nith.no  
Link-local IPv6 Address . . . . . : fe80::50e5:40ff:6794:1d5a%19  
IPv4 Address. . . . . : 10.21.25.60  
Subnet Mask . . . . . : 255.255.252.0  
Default Gateway . . . . . : 10.21.24.1
```

- 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.

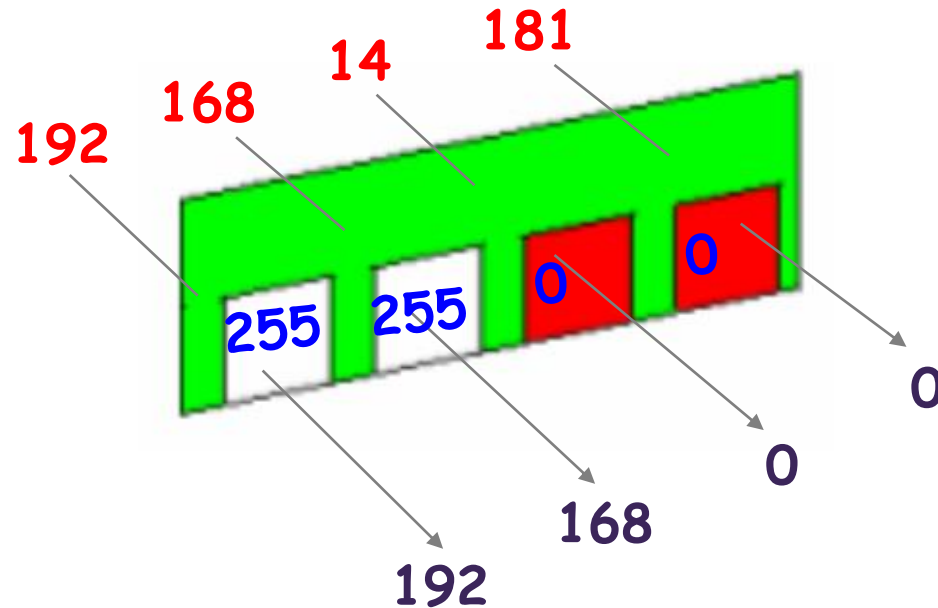
IPv4 datagram-format



Nettmaske



- 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



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

IP-adresse & Nettmaske = IP-Nettverk

- Maskiner/adaptore 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: C**lassless **I**nter**D**omain **R**outing
 - Nettverksdel (prefix) av adressen er av vilkårlig lengde
 - adresseformat: **a.b.c.d/x**, hvor x er antall bit i nettverks-delen av adressen



200.23.16.0/23

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!!!

ICMP - Internet Control Message Protocol

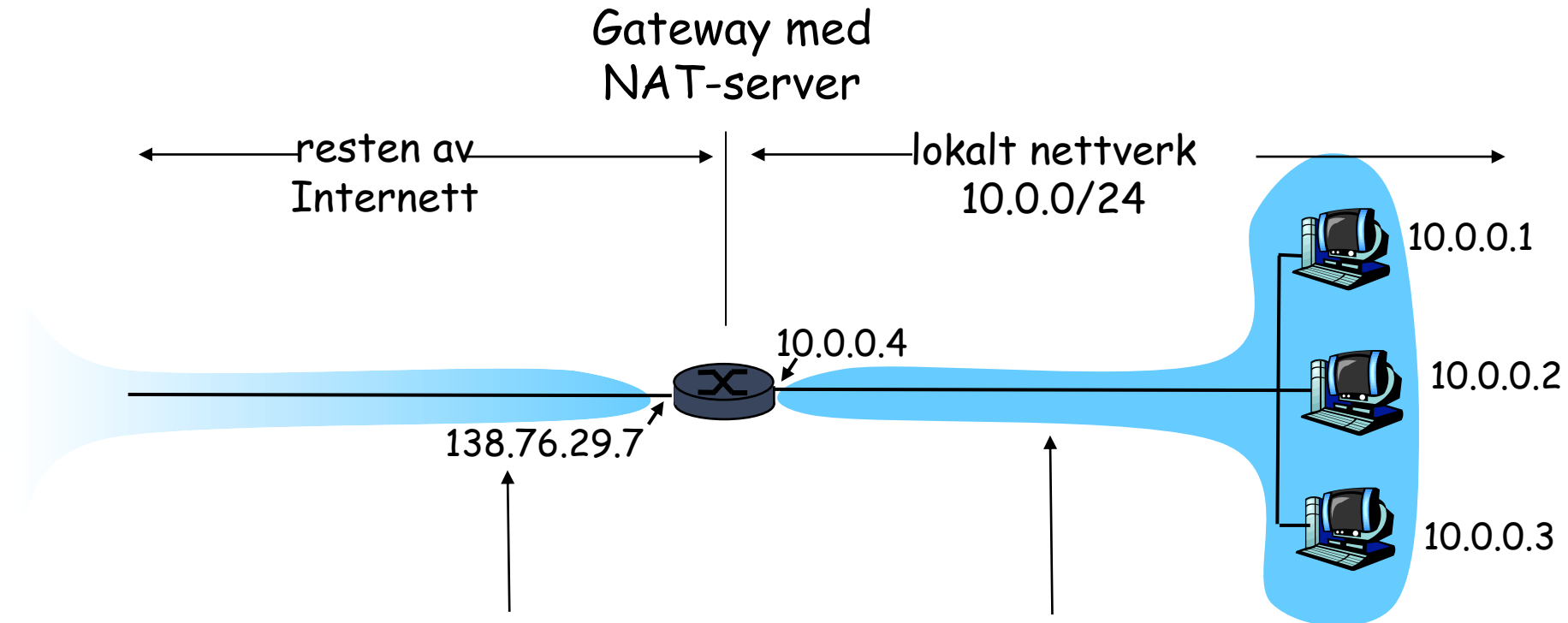
- Brukes av host, router og gateway
 - Feil-rapportering
 - Ekko forespørsel/svar (ping)
- Nettverkslag "over" IP
 - ICMP multiplekkes med datagrammet
- ICMP-melding
 - Type, kode og første 8 byte i datagrammet med feilen
- ping og traceroute utnytter ofte ICMP

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

NAT: Network Address Translation

- **Hvorfor?:** LANet har kun en/noen få IP-adresse fra Internetts perspektiv:
- ISP slipper å tildele et adresseområde:
 - kun en/noen få IP-adresse(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 ikke direkte adresserbare eller synlige for utenforstående (bedre sikkerhet)

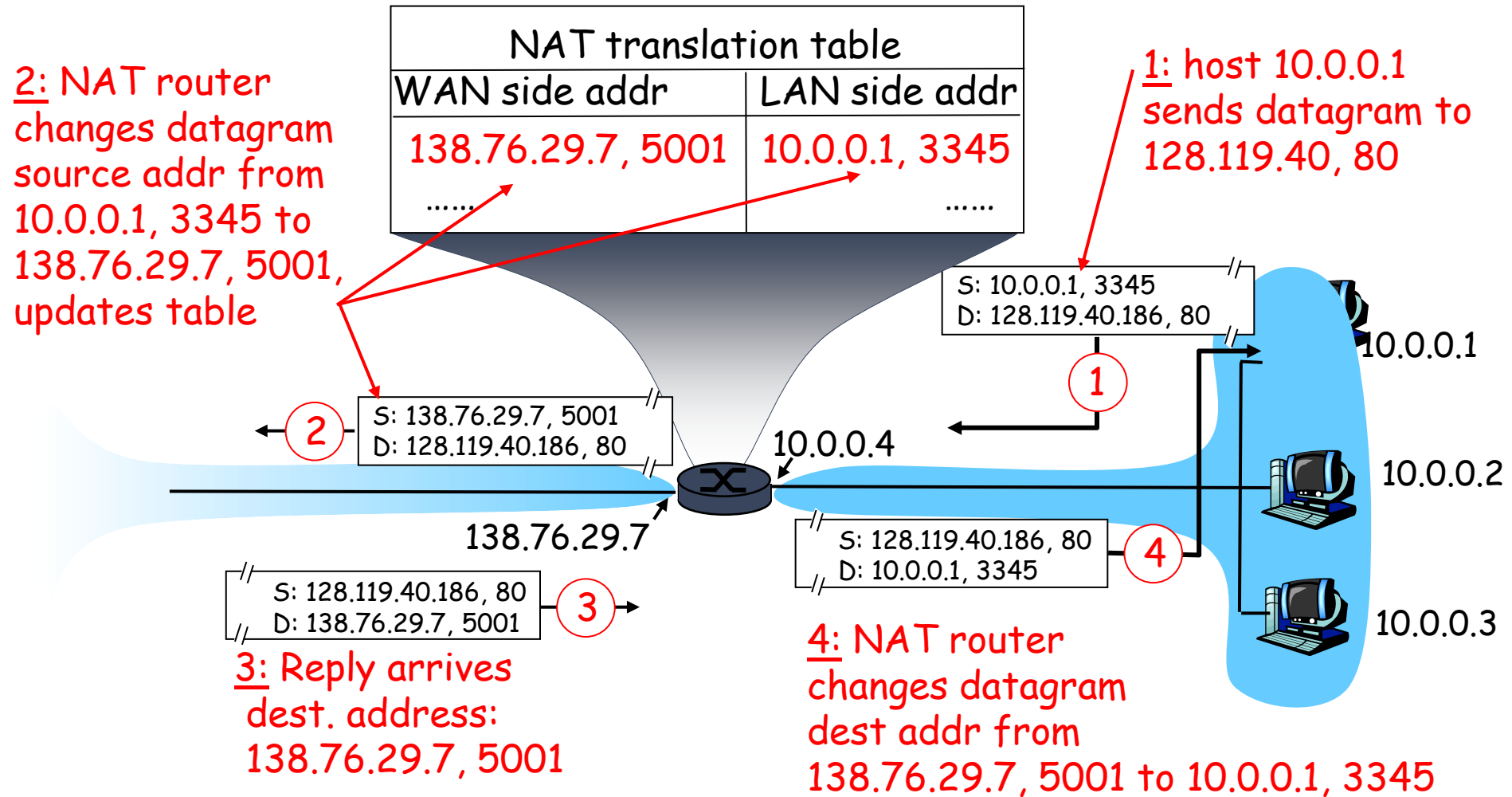
NAT: Network Address Translation



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



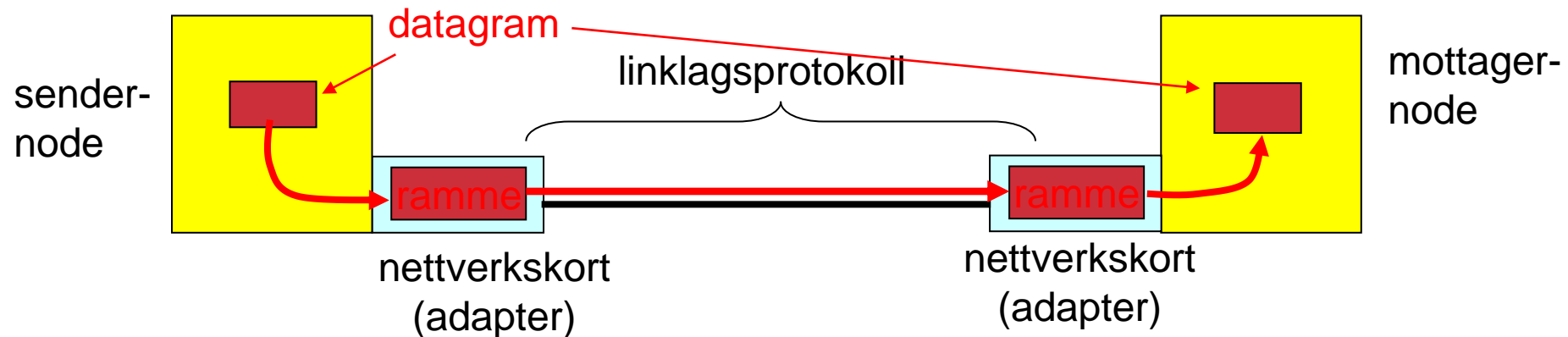
Linklaget (“Datalinjelaget”)

Mål:

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

- **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

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

CSMA/CD (Collision Detection)

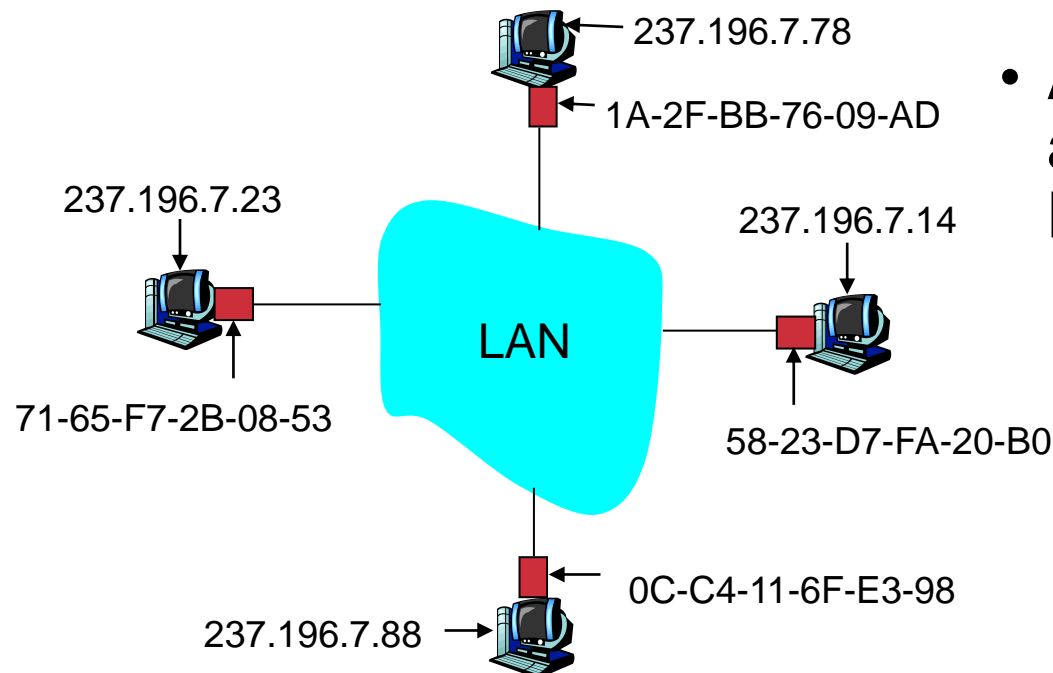
CSMA/CD: lytter på mediet før sending, venter hvis mediet er opptatt (som i CSMA)

- 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

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 adresse-mappinger for noen LAN-noder

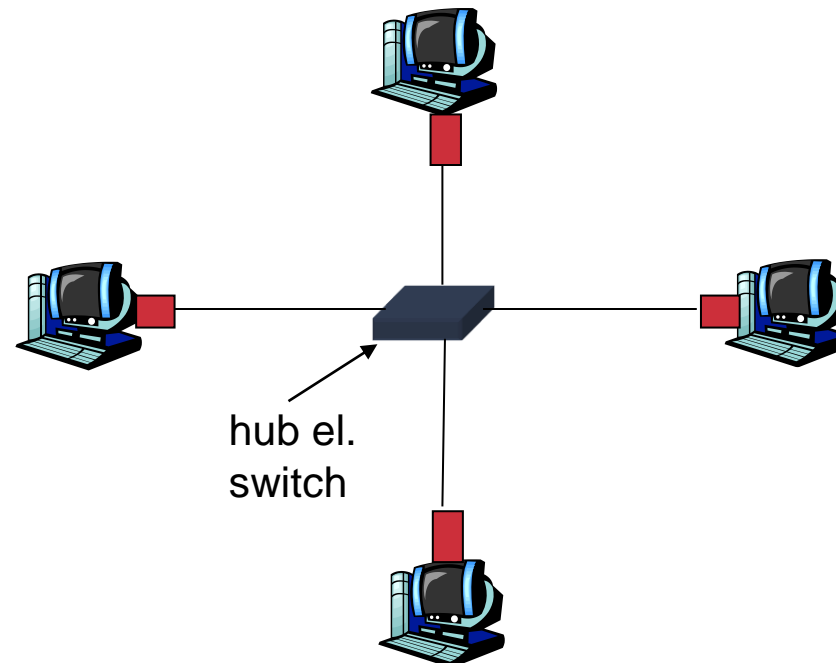
<IP-adresse; MAC-adresse; TTL>

- TTL (Time To Live): tiden mappingen skal ligge i ARP-tabellen (typisk 20 min)

ARP (Adress 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

- 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

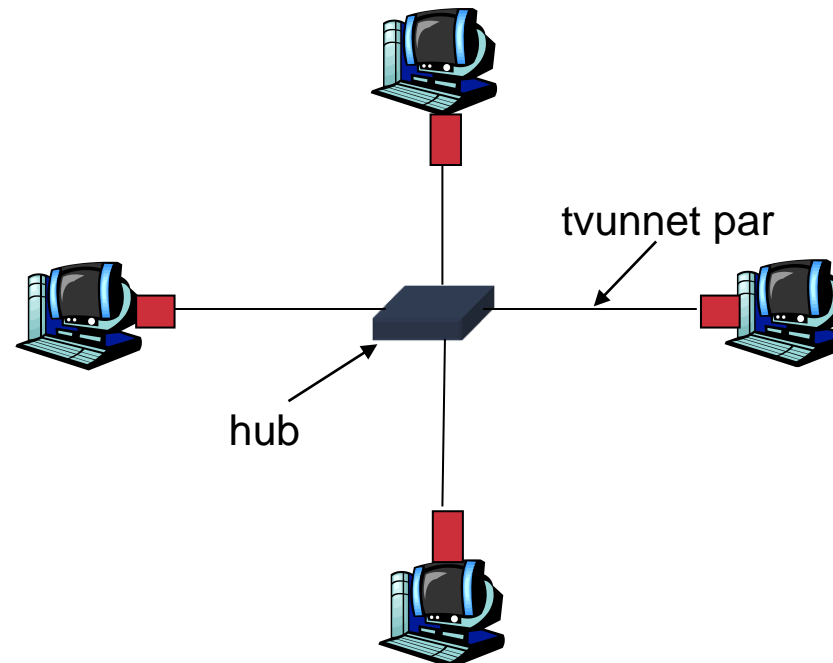
- **Forbindelsesløs:** Ingen håndhissing mellom sender og mottager
 - Derimot så fremforhandler nettverkskortene hvilken IEEE 802-versjon og bitrate de skal benytte første gang de er i forbindelse
- **Upålitelig:** mottager sender ikke ACK eller NAK tilbake til senderen
 - strømmen av datagrammer som 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**)

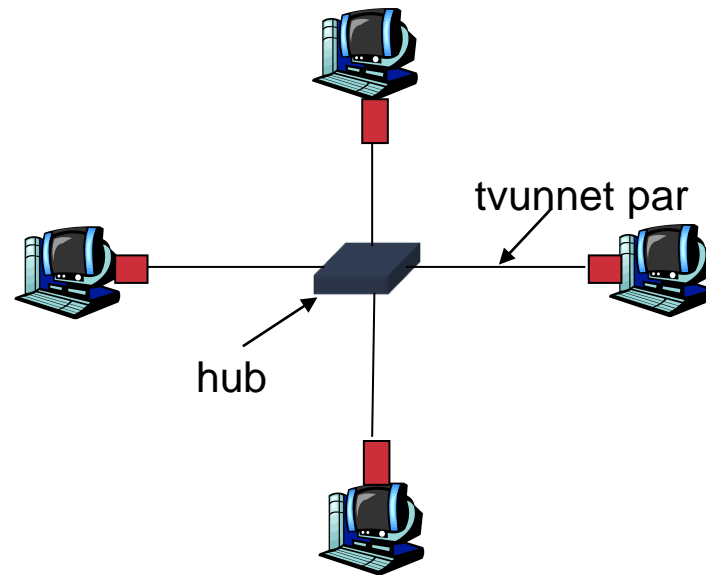
10BaseT og 100BaseT

- 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



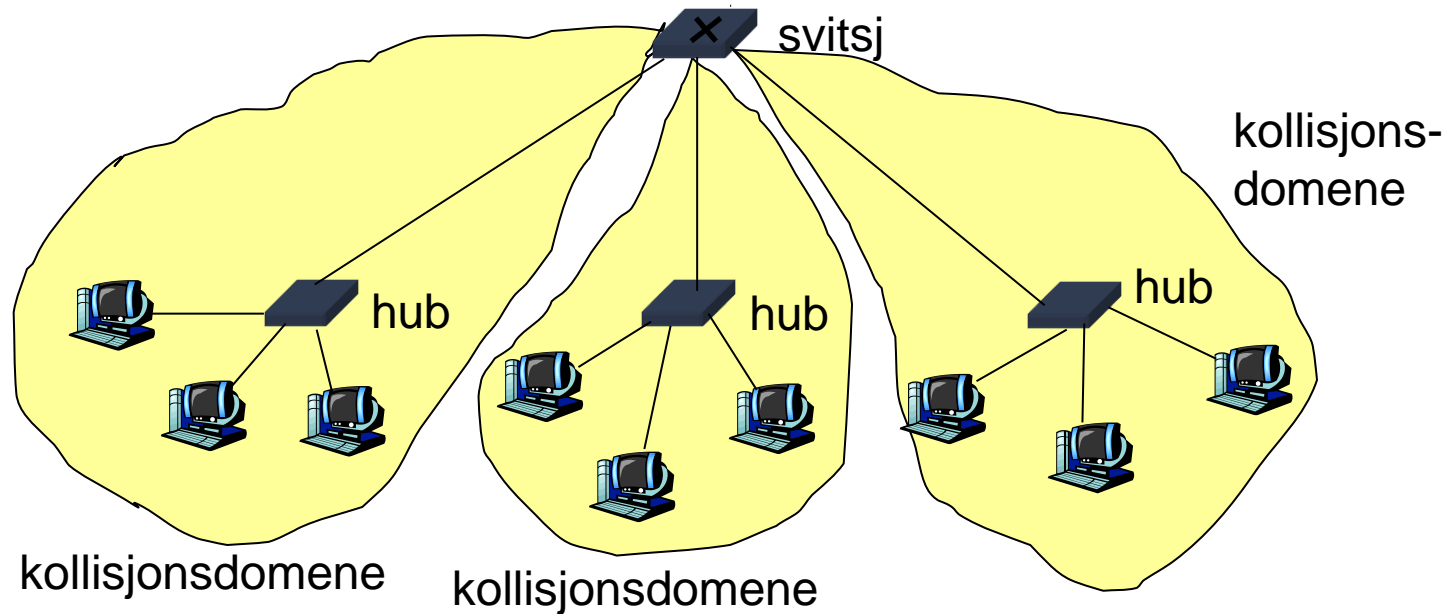
Huber er multiport repeatere (fysisk lag):

- 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



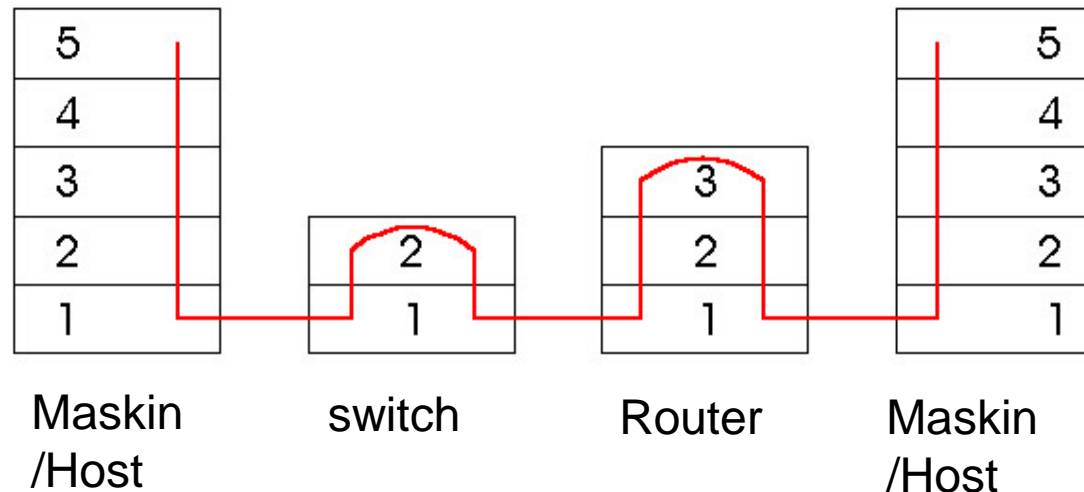
Svitsj: trafikkisolasjon

- 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 vs. routere

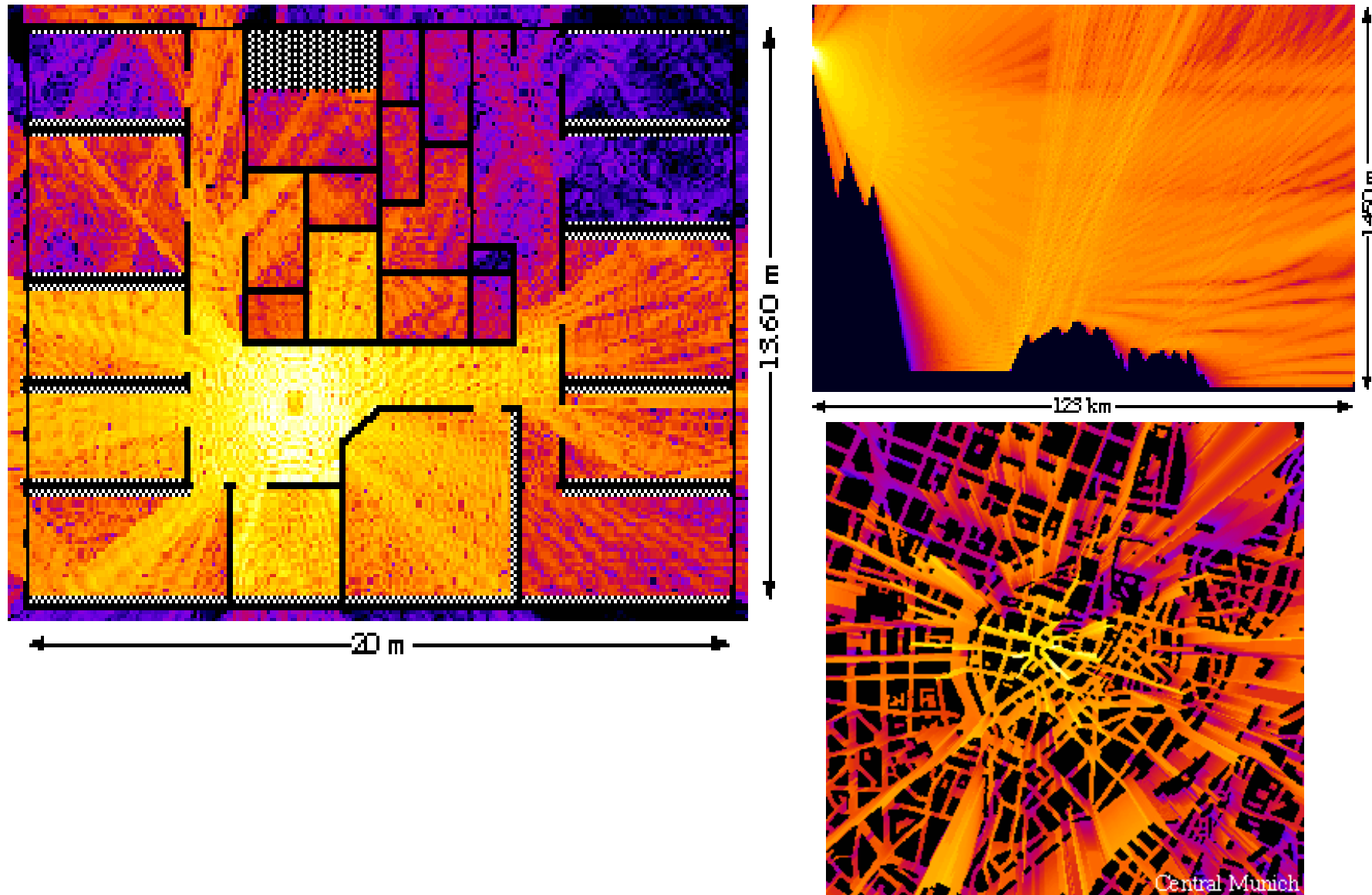
- begge er “store-and-forward” enheter
 - routere: nettlagsenheter (ser på nettlagsheadere)
 - svitsjer er linklagsenheter
- routere benytter routingtabeller og implementerer routingalgoritmer
- svitsjer benytter svitsjetabeller, gjør filtrering, og har selvlæring



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
 - OFDM
- **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*

Eksempel: radiostråling / intensitet



Repetisjon øvingsoppgaver

Hvorfor praktiske øvingsoppgaver?

- Et fag med kun teori blir veldig tørt og vanskeligere for de fleste
- Ved å se ting i praksis, om det så kun er i Wireshark, gjør teorien lettere å forstå
- Mange av verktøyene vi har vært gjennom vil gjøre resten av studiet enklere, og vil også være til stor hjelp når dere kommer ut i arbeidslivet
- Vi blir ikke eksperter i noen av verktøyene, vi har kun lært grunnleggende bruk av verktøy – så selv om noen synes verktøyene er vanskelige i seg selv, så trenger dere ikke kunne avansert bruk av disse verktøyene :-)

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

Flere studenter har allikevel ønsket en samlet oversikt over verktøy som kan være relevante for de praktiske oppgavene på eksamen, så jeg har derfor gått gjennom forelesningene og generert denne oversikten (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**

Putty/telnet for å sende epost

OSX brukere må installere telnet

telnet klienten er «default» i raw mode, start telnet for denne oppgaven som «telnet send.one.com 25» i terminal

Av sikkerhetshensyn er epost tjenesten vi brukte nå tatt ned i løpet av helgen, hvis man ikke gjorde oppgaven bør man se noen youtube videoer for å forstå temaet selv om du ikke får testet det i praksis

SMTP: https://www.youtube.com/watch?v=wOxj_HvnLmE

Putty/telnet for å laste ned HTTP

Hvor mange testet vg.no eller en annen webside som «avslutter tidlig»? Viktig å teste dette også, et tips er å skrive kommandoene inn i en editor (for eksempel notepad) og bruke Copy-Paste inn i PuTTY/telnet vinduet:

```
GET /index.htm HTTP/1.1
```

```
Host: www.eastwillsecurity.com
```

I PuTTY vinduet høyre-klikk for å kopiere inn, og trykk ENTER to ganger for å sende kommandoen.

HTTP: <https://www.youtube.com/watch?v=ptJYNY7UbQU>

Putty/telnet for å laste opp til HTTP

Hvor mange gikk utover oppgaven og testet andre HTTP kommandoer som HEAD og POST?

HEAD er enkelt erstatning for GET, POST gjør at du også kan laste opp filer/data (hvis serveren støtter det), eneste forskjellen er at du må sette Content-Length

```
POST /upload.php HTTP/1.1
```

```
Host: speedtest.tele2.net
```

```
Content-Length: 3
```

```
abc
```

That's all folks...

Lykke til på eksamen