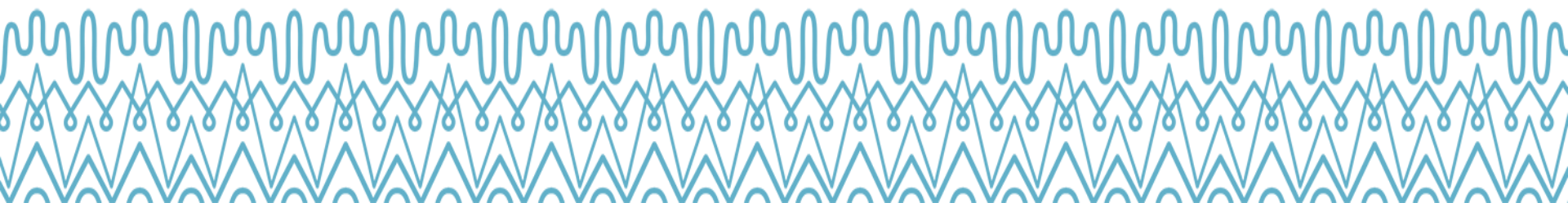


# TK1100

0xB forelesning:

## Repetisjon



# Eksamen

- 3 timer
- Ingen hjelpemidler
- Svar konsist, husk at man må rekke å svare på alle oppgaver!
  - Unntaket er oppgaver hvor det står “diskuter” eller “grei ut om”, der er det forventet å en mer utfyllende besvarelse
- Teller 75% av endelig karakter, så du kan forbedre karakteren du fikk på Del 1

# ASCII Tabellen (7 bit)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	<b>Space</b>	64	40	100	&#64;	<b>@</b>	96	60	140	&#96;	<b>`</b>
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	<b>!</b>	65	41	101	&#65;	<b>A</b>	97	61	141	&#97;	<b>a</b>
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	<b>"</b>	66	42	102	&#66;	<b>B</b>	98	62	142	&#98;	<b>b</b>
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	<b>#</b>	67	43	103	&#67;	<b>C</b>	99	63	143	&#99;	<b>c</b>
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	<b>\$</b>	68	44	104	&#68;	<b>D</b>	100	64	144	&#100;	<b>d</b>
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	<b>%</b>	69	45	105	&#69;	<b>E</b>	101	65	145	&#101;	<b>e</b>
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	<b>&amp;</b>	70	46	106	&#70;	<b>F</b>	102	66	146	&#102;	<b>f</b>
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	<b>'</b>	71	47	107	&#71;	<b>G</b>	103	67	147	&#103;	<b>g</b>
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	<b>(</b>	72	48	110	&#72;	<b>H</b>	104	68	150	&#104;	<b>h</b>
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	<b>)</b>	73	49	111	&#73;	<b>I</b>	105	69	151	&#105;	<b>i</b>
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	<b>*</b>	74	4A	112	&#74;	<b>J</b>	106	6A	152	&#106;	<b>j</b>
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	<b>+</b>	75	4B	113	&#75;	<b>K</b>	107	6B	153	&#107;	<b>k</b>
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	<b>,</b>	76	4C	114	&#76;	<b>L</b>	108	6C	154	&#108;	<b>l</b>
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	<b>-</b>	77	4D	115	&#77;	<b>M</b>	109	6D	155	&#109;	<b>m</b>
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	<b>.</b>	78	4E	116	&#78;	<b>N</b>	110	6E	156	&#110;	<b>n</b>
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	<b>/</b>	79	4F	117	&#79;	<b>O</b>	111	6F	157	&#111;	<b>o</b>
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	<b>0</b>	80	50	120	&#80;	<b>P</b>	112	70	160	&#112;	<b>p</b>
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	<b>1</b>	81	51	121	&#81;	<b>Q</b>	113	71	161	&#113;	<b>q</b>
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	<b>2</b>	82	52	122	&#82;	<b>R</b>	114	72	162	&#114;	<b>r</b>
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	<b>3</b>	83	53	123	&#83;	<b>S</b>	115	73	163	&#115;	<b>s</b>
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	<b>4</b>	84	54	124	&#84;	<b>T</b>	116	74	164	&#116;	<b>t</b>
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	<b>5</b>	85	55	125	&#85;	<b>U</b>	117	75	165	&#117;	<b>u</b>
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	<b>6</b>	86	56	126	&#86;	<b>V</b>	118	76	166	&#118;	<b>v</b>
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	<b>7</b>	87	57	127	&#87;	<b>W</b>	119	77	167	&#119;	<b>w</b>
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	<b>8</b>	88	58	130	&#88;	<b>X</b>	120	78	170	&#120;	<b>x</b>
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	<b>9</b>	89	59	131	&#89;	<b>Y</b>	121	79	171	&#121;	<b>y</b>
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	<b>:</b>	90	5A	132	&#90;	<b>Z</b>	122	7A	172	&#122;	<b>z</b>
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	<b>;</b>	91	5B	133	&#91;	<b>[</b>	123	7B	173	&#123;	<b>{</b>
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<b>&lt;</b>	92	5C	134	&#92;	<b>\</b>	124	7C	174	&#124;	<b> </b>
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	<b>=</b>	93	5D	135	&#93;	<b>]</b>	125	7D	175	&#125;	<b>}</b>
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	<b>&gt;</b>	94	5E	136	&#94;	<b>^</b>	126	7E	176	&#126;	<b>~</b>
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	<b>?</b>	95	5F	137	&#95;	<b>_</b>	127	7F	177	&#127;	<b>DEL</b>

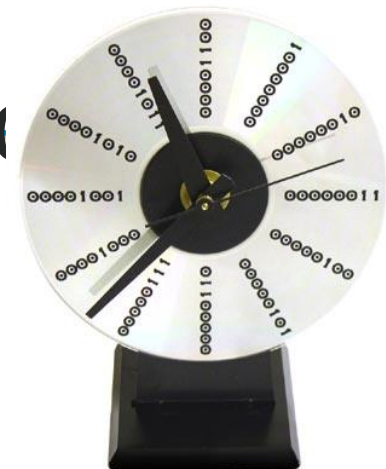
# Tolking av binære koder (noen få)

	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	𐀀		𐀁	
JVM bytekode	istore_3	Istore_2	frem	Istore
X86 opkode	DS	AND	JNO	AAA

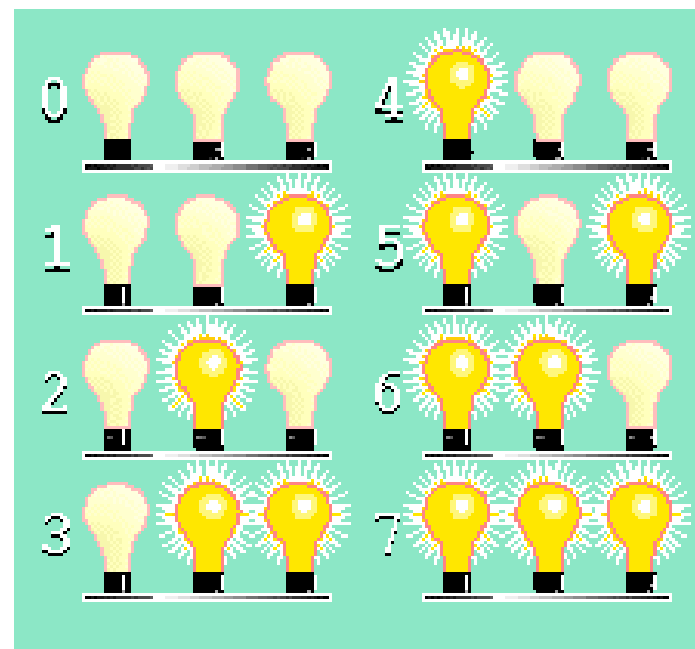
Hvordan vet man hvilken tolkning som er riktig?

- Hvordan vet man hvilken tolkning som er riktig?
- Det bestemmer (bruken i) programmet!

# Binær tall-representasjon

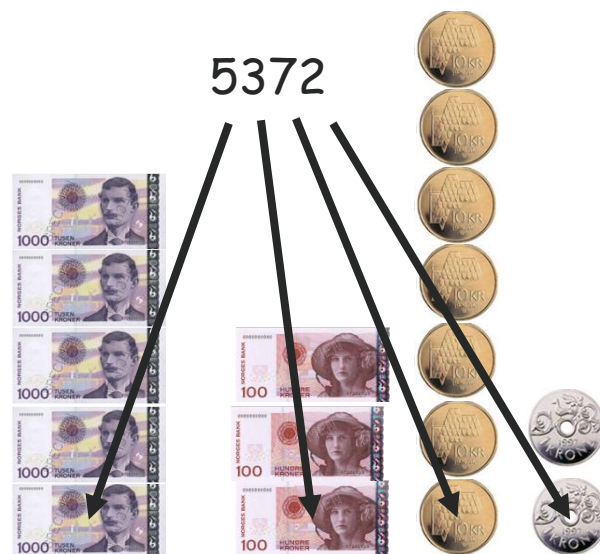


- Med 3 lamper kan vi representere  $2^3 = 8$  kombinasjoner av av/på
- Med Av=0 og På=1 får vi
  - 0 = 000    4 = 100
  - 1 = 001    5 = 101
  - 2 = 010    6 = 110
  - 3 = 011    7 = 111
- Dette kan utvides til å bruke flere lamper (transistorer), f. eks. 8, 16, 32, 64, ....



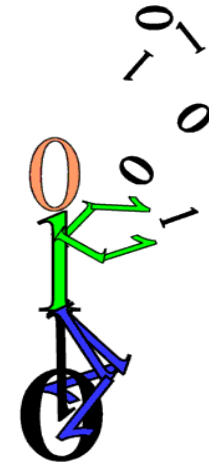
# Desimale tall - posisjontall

- 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

$$- 407_{23} = 4 \cdot 23^2 + 0 \cdot 23^1 + 7 \cdot 23^0 = 2116 + 0 + 7 = 2123_{10}$$

$$- \quad \quad = 6122_7$$

- Det fine med posisjonelle tallsystemer er at fravær av en potens-verdi («vekt») kan representeres med **0**

$$- 4\mathbf{0}7 \text{ er ikke det samme som } 47$$

# 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

$$\begin{aligned}
 1101010011 &= 1*2^9 + 1*2^8 + 0*2^7 + 1*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 \\
 &= 512 + 256 + 64 + 16 + 2 + 1 \\
 &= 851
 \end{aligned}$$



512	256	128	64	32	16	8	4	2	1
<b>1 1 0 1 0 1 0 0 1 1</b>									
512	256	64	16				2	1	

2 <sup>0</sup>	=	1
2 <sup>1</sup>	=	2
2 <sup>2</sup>	=	4
2 <sup>3</sup>	=	8
2 <sup>4</sup>	=	16
2 <sup>5</sup>	=	32
2 <sup>6</sup>	=	64
2 <sup>7</sup>	=	128
2 <sup>8</sup>	=	256
2 <sup>9</sup>	=	512
2 <sup>10</sup>	=	1024

**NB! Her mangler innledende nuller i 16 bit presisjon!**

0000 0011 0101 0011

# Regneoperasjoner - binærtall

## Addisjon

$$\begin{array}{r}
 \phantom{+} \overset{1}{0} \overset{1}{0} \overset{1}{0} 00 \phantom{00} 1011 \\
 + 0001 \phantom{00} 1010 \\
 \hline
 0010 \phantom{00} 0101
 \end{array}$$

Se  
også D1.0

## Multiplikasjon

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

Se  
også D1.0

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

1110 1101

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

2's komplement = 1's komplement + 1



# Subtraksjon

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

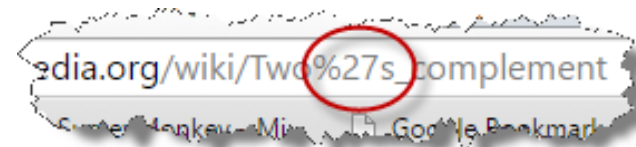
$$\begin{array}{rclcl}
 46 & = & 0010\ 1110 & = & 0010\ 1110 \\
 -37 & = & -0010\ 0101 & = & +1101\ 1011 \\
 \hline
 9 & & & & \hline
 \hline
 & & & & 1\ 0000\ 1001 \\
 & & & & \hline
 & & & & \hline
 \end{array}$$

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

# ASCII – struktur

PUGG!!

- **0x00-0x1F**: De 32 første kodene er **kontrolltegn**.
  - 0x07 er f.eks. beep i høyttaler
  - Linjeskift er
    - i **Windows** CarriageReturn (0x0D) **OG** LineFeed 0x0A),
    - i **Unix/Linux** og **OSX** bare LineFeed (0x0A)
  - Mellomrom-tegnet er **0x20**
    - Noensinne sett %20 e.l. i en URL?
- Tallene **kodes** med: **0x30-0x39**
  - Binærtallet **0000 0000** har ASCII-kode **0011 0000**
- "A" er **0x41** -> 01**00** 0001  
"a" er **0x61** -> 01**10** 0001
  - En enkelt bit forskjell mellom store og små bokstaver
  - Numerisk sorteres dermed alle de store foran de små bokstavene

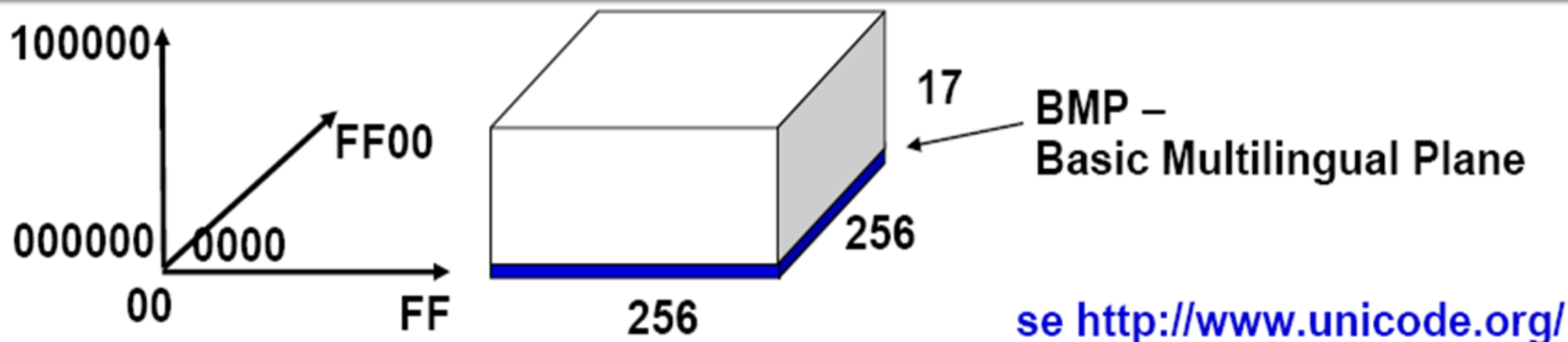


# Windows 1252 kodetabell

Bygger på ISO 8859-1

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
0	NUL	DLE	space	0	@	P	`	p	€	undef	no break space	°	À	Đ	à	ð
1	SOH	DC1	!	1	A	Q	a	q	undef	‘	¡	±	Á	Ñ	á	ñ
2	STX	DC2	”	2	B	R	b	r	,	’	¢	²	Â	Ò	â	ò
3	ETX	DC3	#	3	C	S	c	s	f	“	£	³	Ã	Ó	ã	ó
4	EOT	DC4	\$	4	D	T	d	t	„	”	¤	´	Ä	Ô	ä	ô
5	ENQ	NAK	%	5	E	U	e	u	...	•	¥	µ	Å	Õ	å	õ
6	ACK	SYN	&	6	F	V	f	v	†	–	¦	¶	Æ	Ö	æ	ö
7	BEL	ETB	'	7	G	W	g	w	‡	—	§	·	Ç	×	ç	÷
8	BS	CAN	(	8	H	X	h	x	^	~	¨	¸	È	Ø	è	ø
9	HT	EM	)	9	I	Y	i	y	‰	™	©	¹	É	Ù	é	ù
A	LF	SUB	*	:	J	Z	j	z	Š	š	ª	º	Ê	Ú	ê	ú
B	VT	ESC	+	;	K	[	k	{	‹	›	«	»	Ë	Û	ë	û
C	FF	FS	,	<	L	\	l		Œ	œ	¬	¼	Ì	Ü	ì	ü
D	CR	GS	-	=	M	]	m	}	undef	undef	-	½	Í	Ý	í	ý
E	SO	RS	.	>	N	^	n	~	Ž	ž	®	¾	Î	Þ	î	þ
F	SI	US	/	?	O	_	o	DEL	undef	ÿ	¯	¿	Ï	ß	ï	ÿ

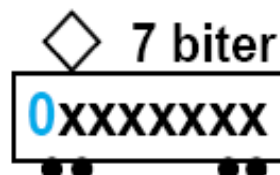
# UNICODE Struktur



- 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
    - Fimalogoer o.l. kan registereres her.

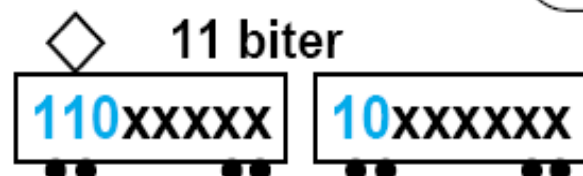
# Unicode UTF-8

- Enslig motorvogn = 0  
+ ASCII-kode

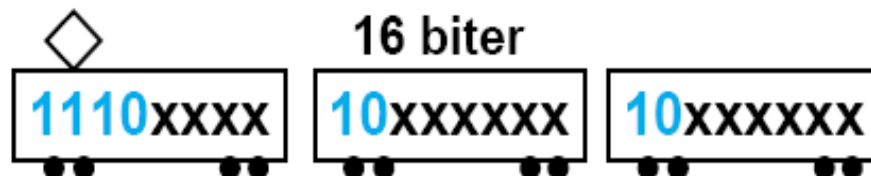


ASCII med en ekstra  
ledende 0  
er kompatibel med  
UTF-8

- 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

**110**x xxxx

**10**xx xxxx

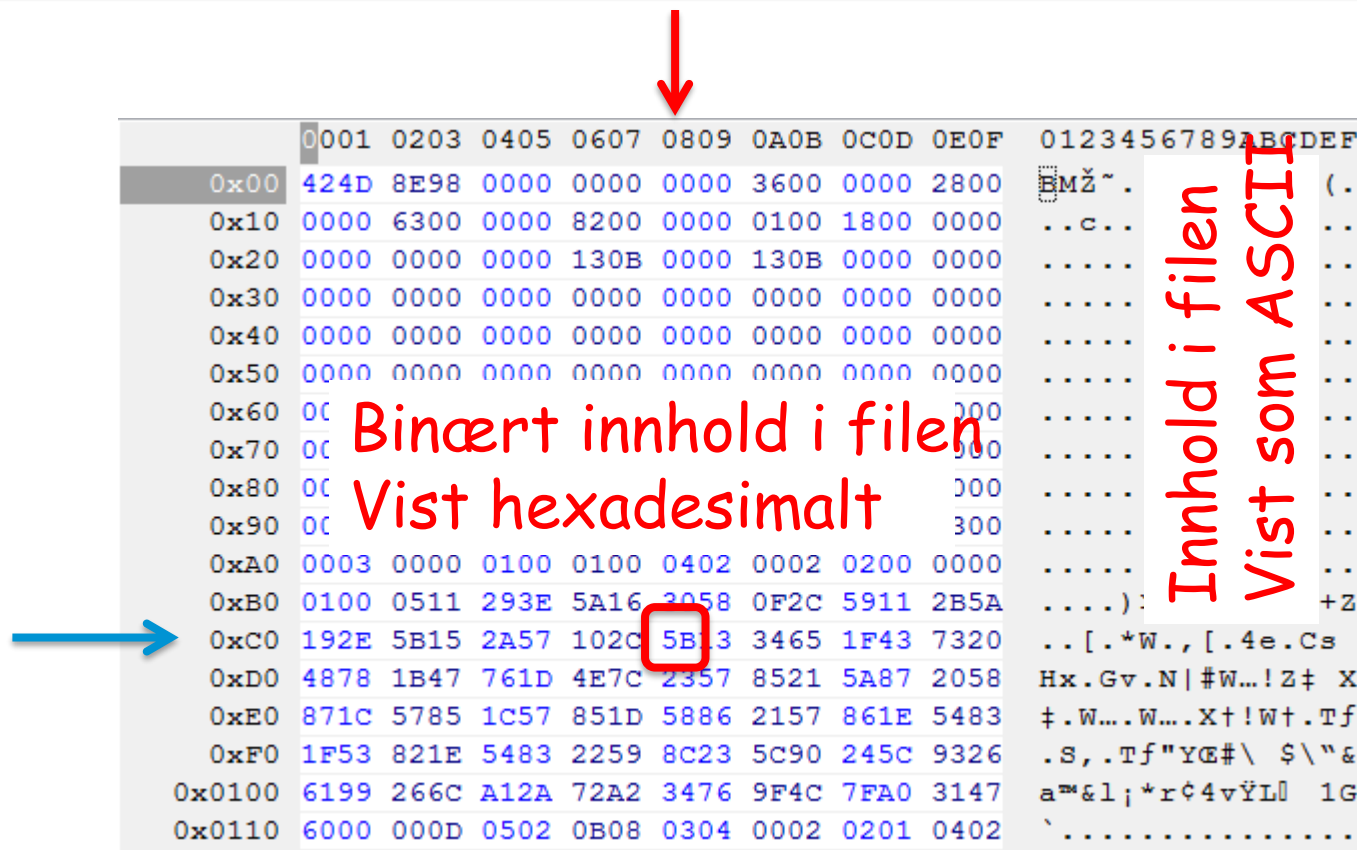
**110**0 00**11**

**10**00 **0101**

«Padding»

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

# HexEditor (2)

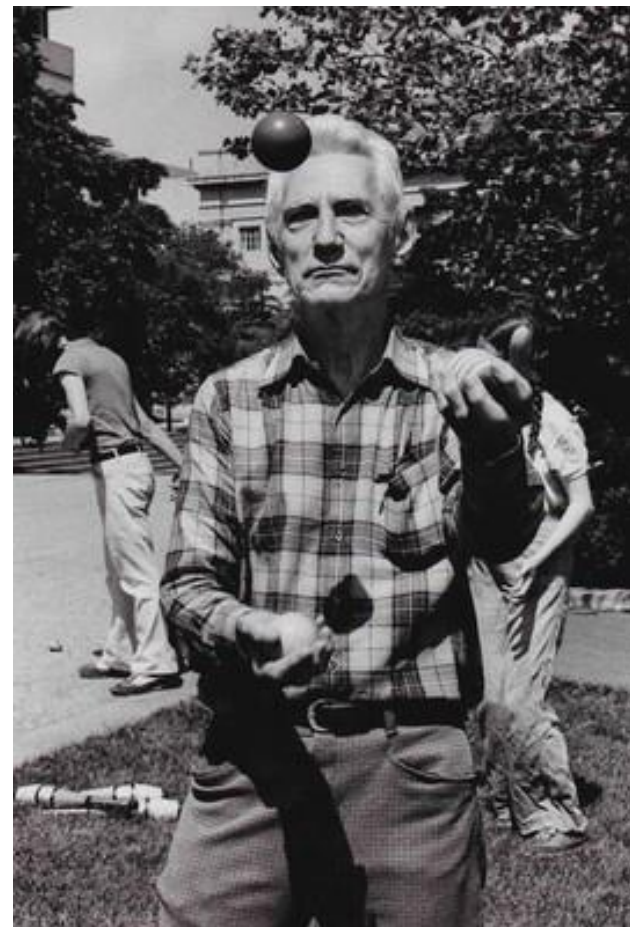


	0001	0203	0405	0607	0809	0A0B	0C0D	0E0F	0123456789ABCDEF
0x00	424D	8E98	0000	0000	0000	3600	0000	2800	BMŽ~.
0x10	0000	6300	0000	8200	0000	0100	1800	0000	..c..
0x20	0000	0000	0000	130B	0000	130B	0000	0000	.....
0x30	0000	0000	0000	0000	0000	0000	0000	0000	.....
0x40	0000	0000	0000	0000	0000	0000	0000	0000	.....
0x50	0000	0000	0000	0000	0000	0000	0000	0000	.....
0x60	00							000	.....
0x70	00							000	.....
0x80	00							000	.....
0x90	00							300	.....
0xA0	0003	0000	0100	0100	0402	0002	0200	0000	.....
0xB0	0100	0511	293E	5A16	2058	0F2C	5911	2B5A	.....):
0xC0	192E	5B15	2A57	102C	5B13	3465	1F43	7320	..[.*W.,[.4e.Cs
0xD0	4878	1B47	761D	4E7C	2357	8521	5A87	2058	Hx.Gv.N #W...!Z† X
0xE0	871C	5785	1C57	851D	5886	2157	861E	5483	†.W...W...X†!W†.Tf
0xF0	1F53	821E	5483	2259	8C23	5C90	245C	9326	.s,.Tf"Y&#\ \$\"&
0x0100	6199	266C	A12A	72A2	3476	9F4C	7FA0	3147	a™&l;*rç4vŸL 1G
0x0110	6000	000D	0502	0B08	0304	0002	0201	0402	`.....

På posisjon **0xC8** er det en byte med verdi **0x5B**  
 $= 0101\ 1011 = 91_{10}$ , som ASCII er det  
 glyfen [

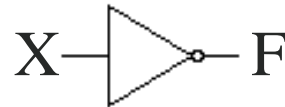
# Boolsk algebra

- 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



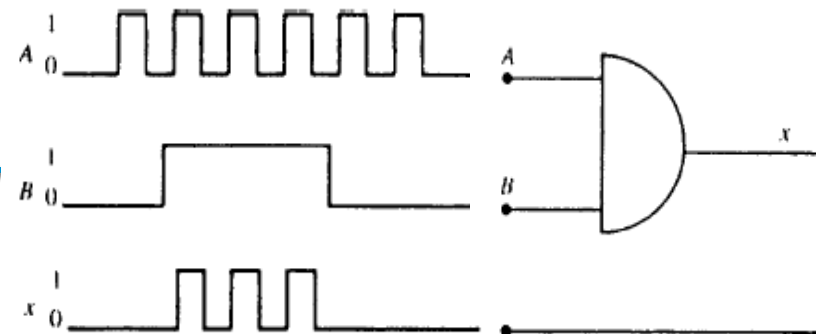
01101001

## NOT

Sannhetstabell

X	F
0	1
1	0

# AND og NAND



10010101

10101100



## AND

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

## NAND

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

# OR og NOR

10111001

10001110



OR

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

NOR

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

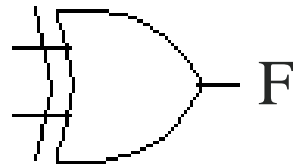
# XOR og XNOR

10110010

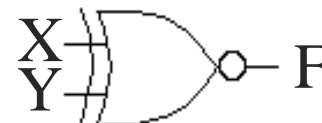
X

11001110

Y



01111100



## XOR

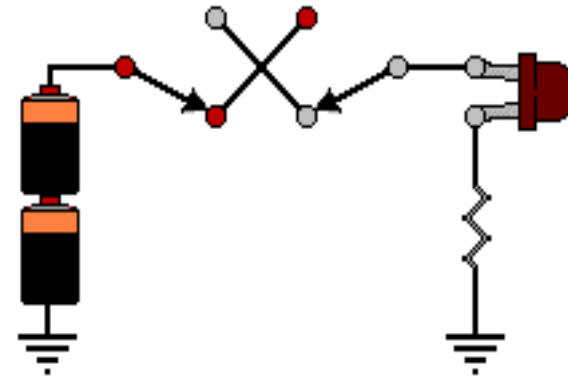
X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

## XNOR

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

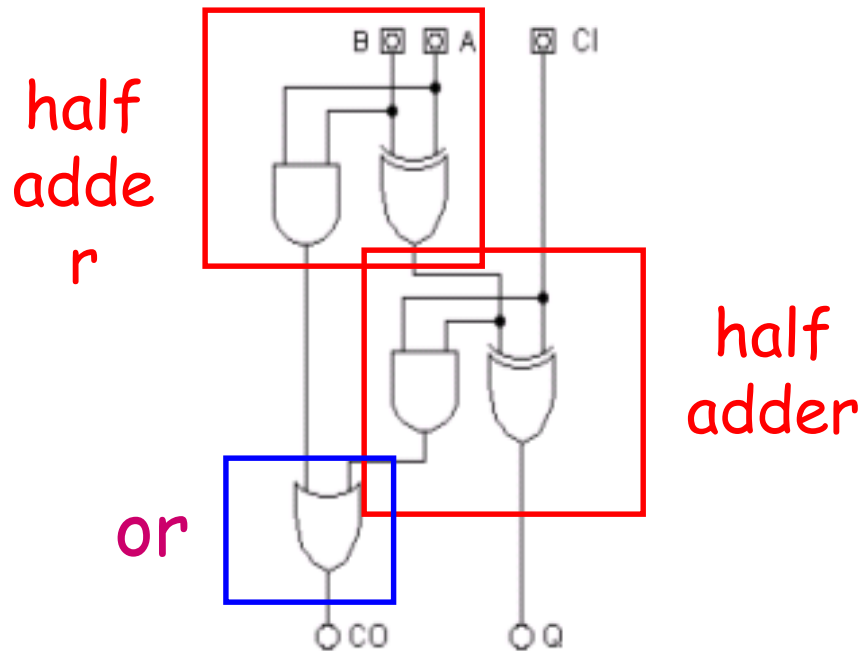
# Eksempel: XOR

A	B	C
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**

# CPU - Produsenter

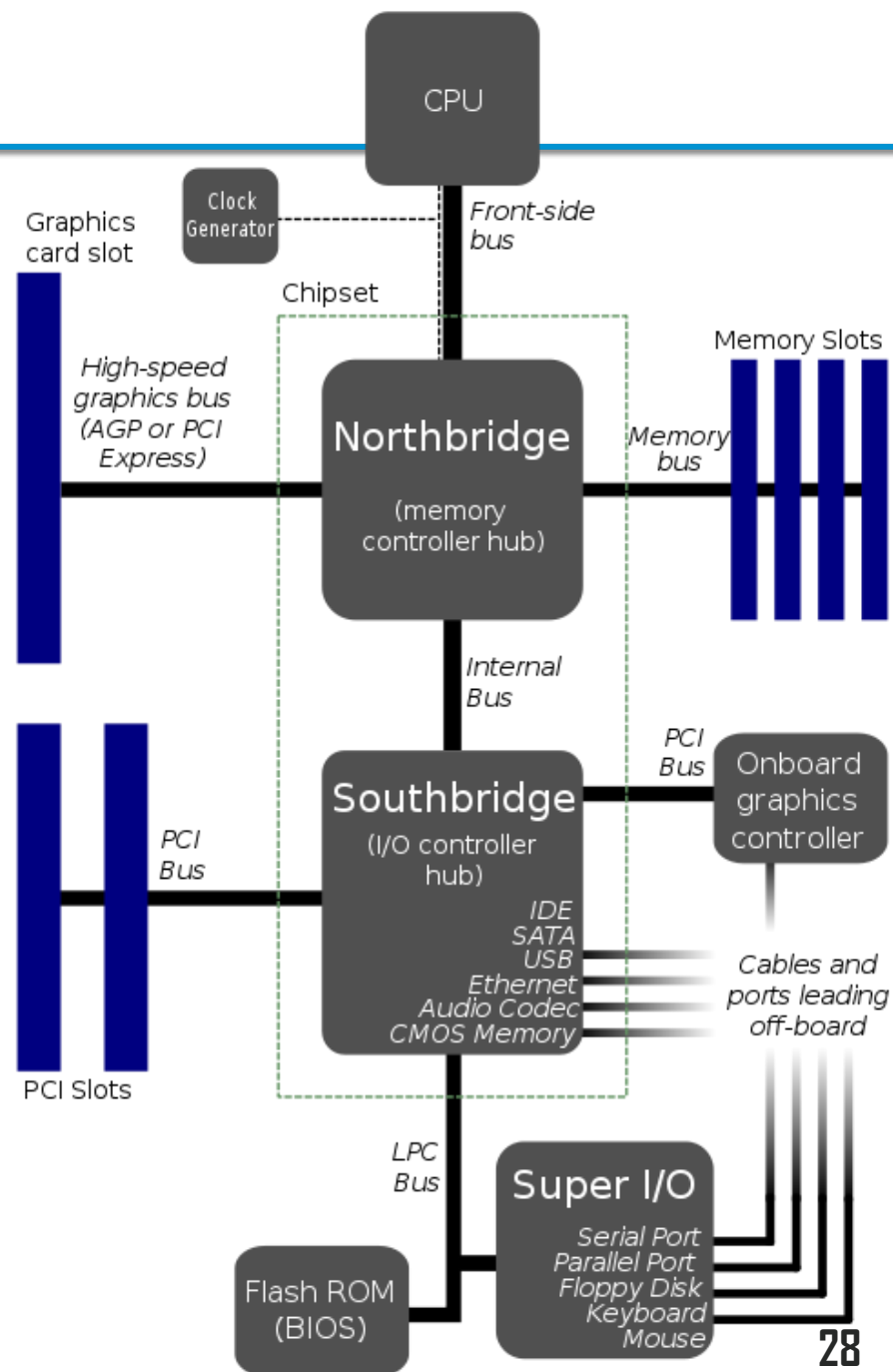
- Intel og AMD
  - Konstruerer og produserer CPUer, chipset m.m.
  - Fokus: **Desktop**, **server**, mobil, embedded
  - **C**omplex **I**nstruction **S**et **C**omputing
- MIPS og ARM
  - Konstruerer og lisenserer CPU-kjerner mm til ulike produsenter
  - Fokus: **Embedded**, **mobil**
  - **R**educed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

# x86 instruksjonsett (CISC)

- **Data flytting**
  - Flytte data mellom registre og til/fra RAM
  - mov, push, pop, ...
- **Kontrolloverføring:**
  - Utføre betingelsessetninger (if, while) og metodekall
  - je, jg, jmp, call, ret
- **Aritmetikk/Logisk:**
  - Utføre operasjoner på data i registre
  - cmp, add, sub, inc, mul, imul, not, and, or, xor
- **Input/Output:** in, out
  - Skrive/lese til porter (og minneadresser)
- **Debug og Interrupt håndtering:**
  - int, sti, hlt, nop
- **Flyttall** har egne instruksjoner
- SIMD vektor og matriseinstruksjoner
  - MMX, 3D Now!, SSE 1-4
- Egne instruksjoner for å endre prosessortilstand (system)

# Hovedkort

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

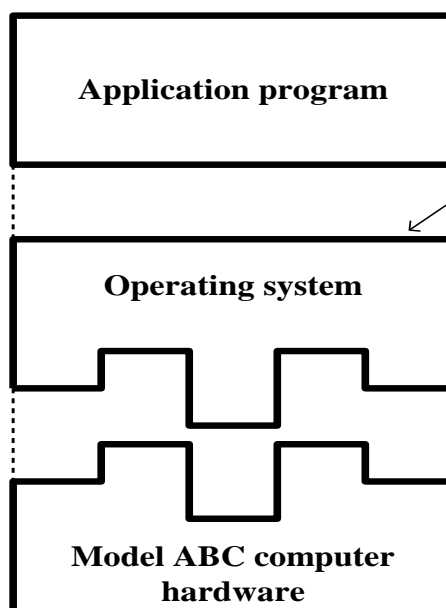
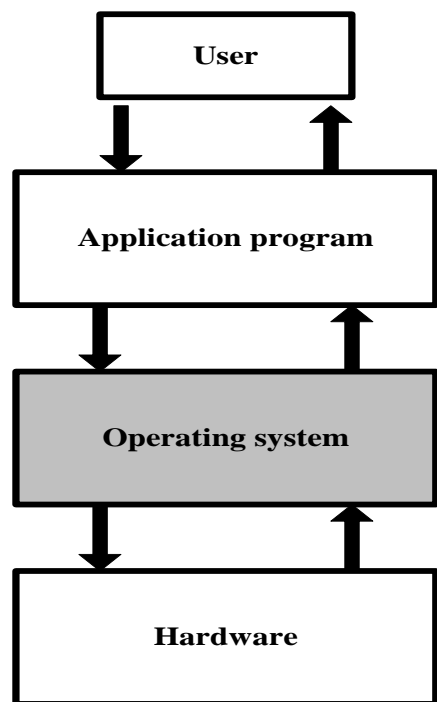
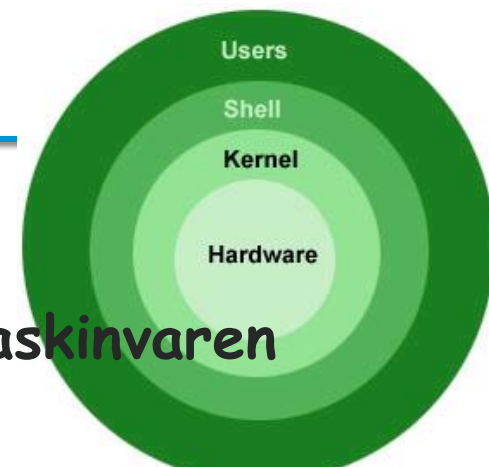


# Del eksamen 1

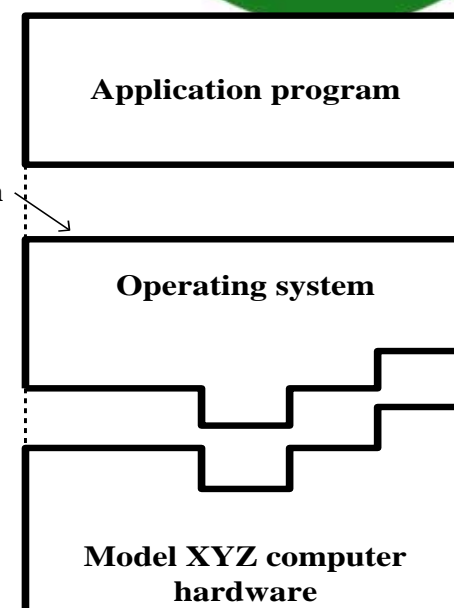
- Pensumet frem til her var med i del eksamen 1, på ordinær eksamen vil derfor ikke dette være med i del 2
- Merk at deler av videre pensum bygger på dette, for eksempel konvertering fra desimal til binær, og AND operatorer og addisjon er sentralt ifm TCP/UDP og IP
- Pensum for Del 2:

# Hva er et operativsystem?

Operativsystemet er programvare som ligger mellom brukeren/programmereren og maskinvaren



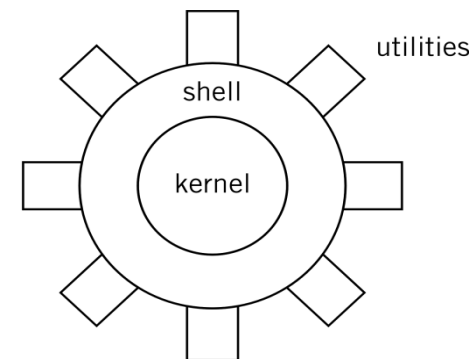
Platform



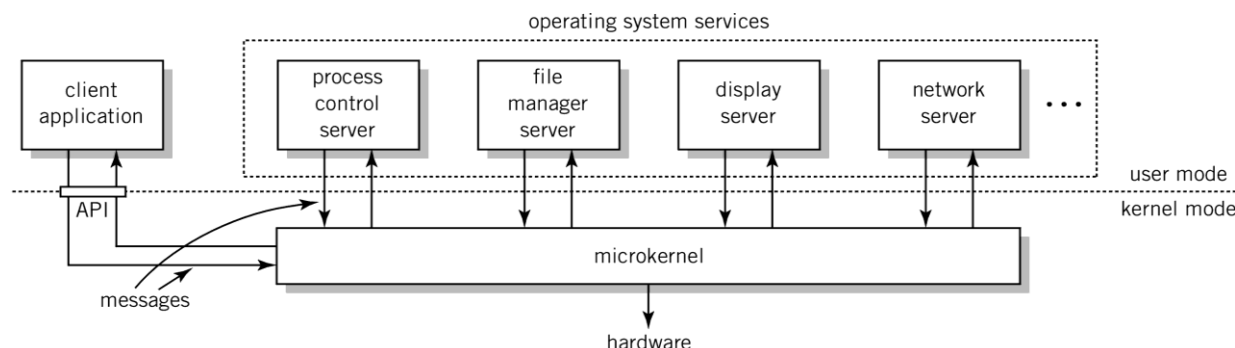
Samme applikasjon på ulike typer maskinvare (OS = "plattform")

# 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, ...



# 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



# 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 **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, utstyrskjører 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

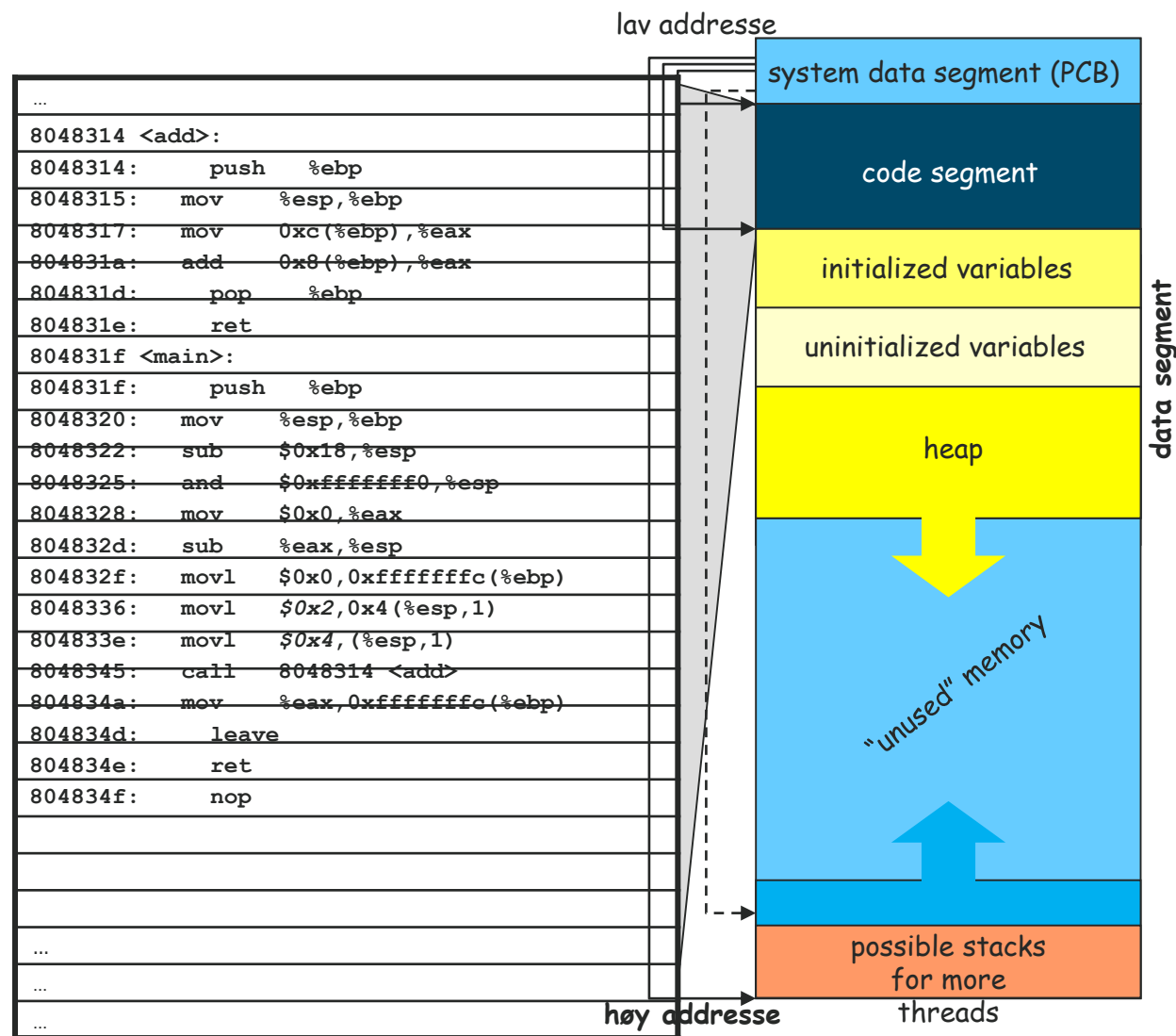
# Virtuelt minne

- = **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 kompiles 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).

# Prosessen (tråden) sitt Minne

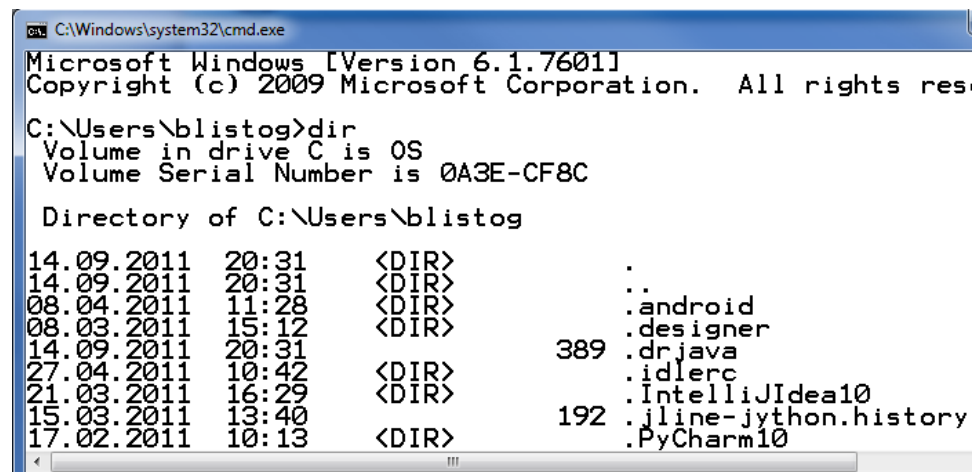
På Intel arkitekturen  
partisjonerer en oppgave  
(task) sitt tildelte minne

- et text (code) segment
  - Lest fra program fil  
for eksempel av `exec`
  - vanligvis read-only
  - Kan deles av flere tråder
- et data segment
  - initialserte globale variabler  
(0 / NULL)
  - uinitialiserte globale variabler
  - heap
    - dynamisk minne f.eks.,  
allokert 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



# "DOS" (cmd.exe) – UNIX/OSX (bash)

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

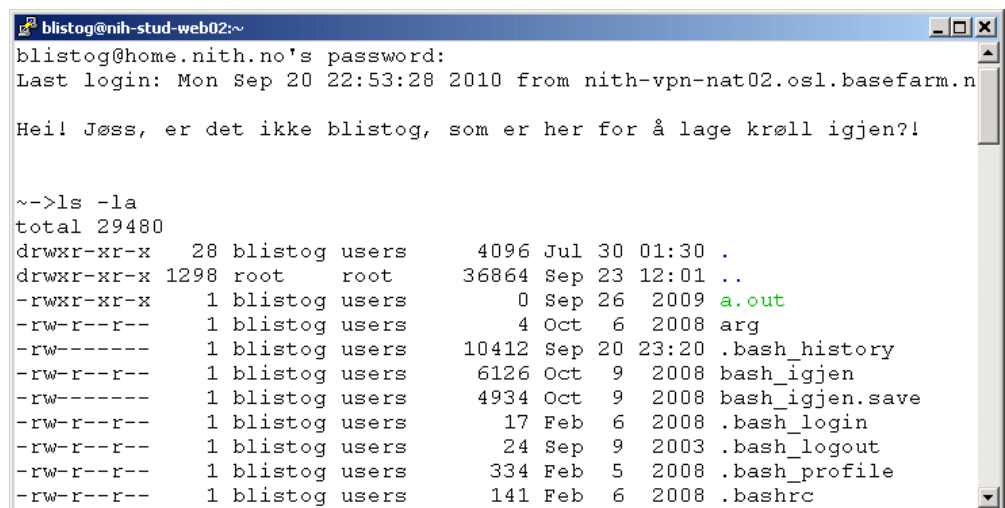


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

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    <DIR>          389.drjava
27.04.2011  10:42    <DIR>          .idlerc
21.03.2011  16:29    <DIR>          .IntelliJ IDEA10
15.03.2011  13:40    <DIR>          192.jline-jython.history
17.02.2011  10:13    <DIR>          .PyCharm10
```



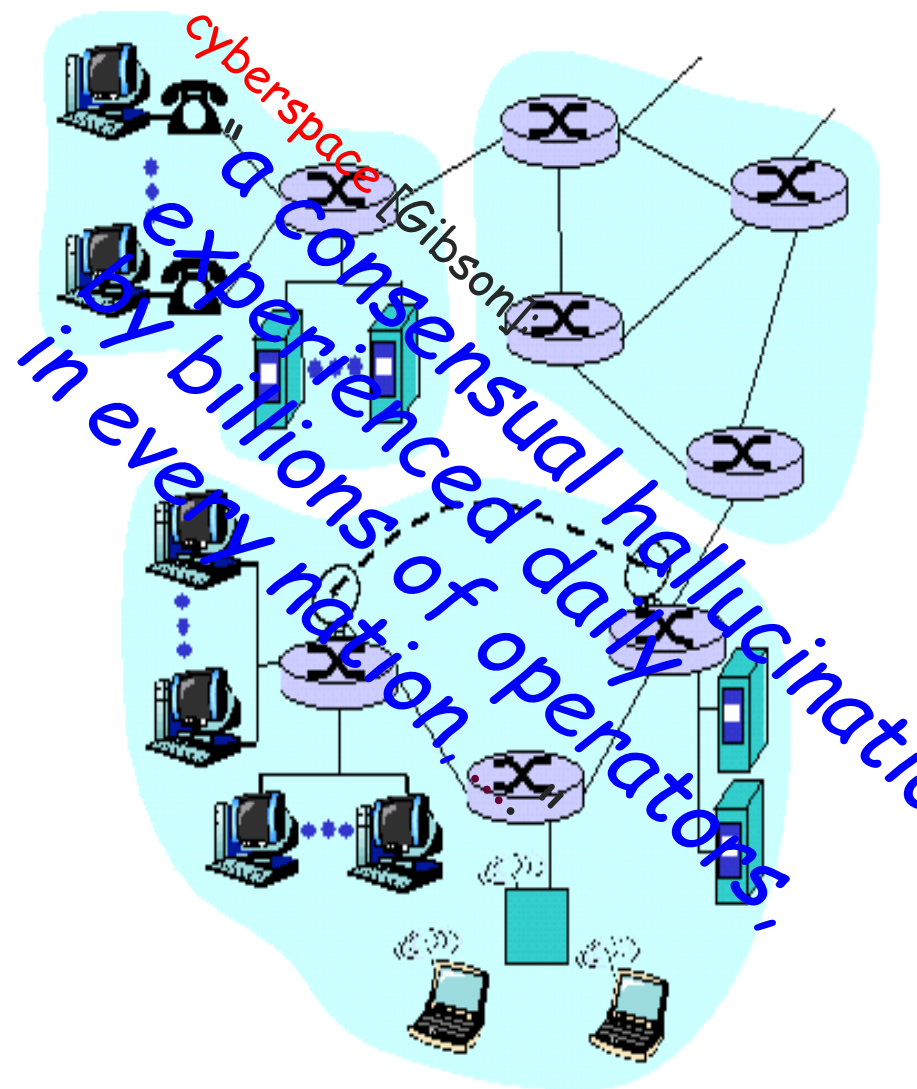
```
blistog@nih-stud-web02:~
blistog@home.nith.no's password:
Last login: Mon Sep 20 22:53:28 2010 from nith-vpn-nat02.osl.basefarm.no

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
-rw-----  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
```

# 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 sammenkople ulike LAN og WAN ("**TCP/IP-stacken**")

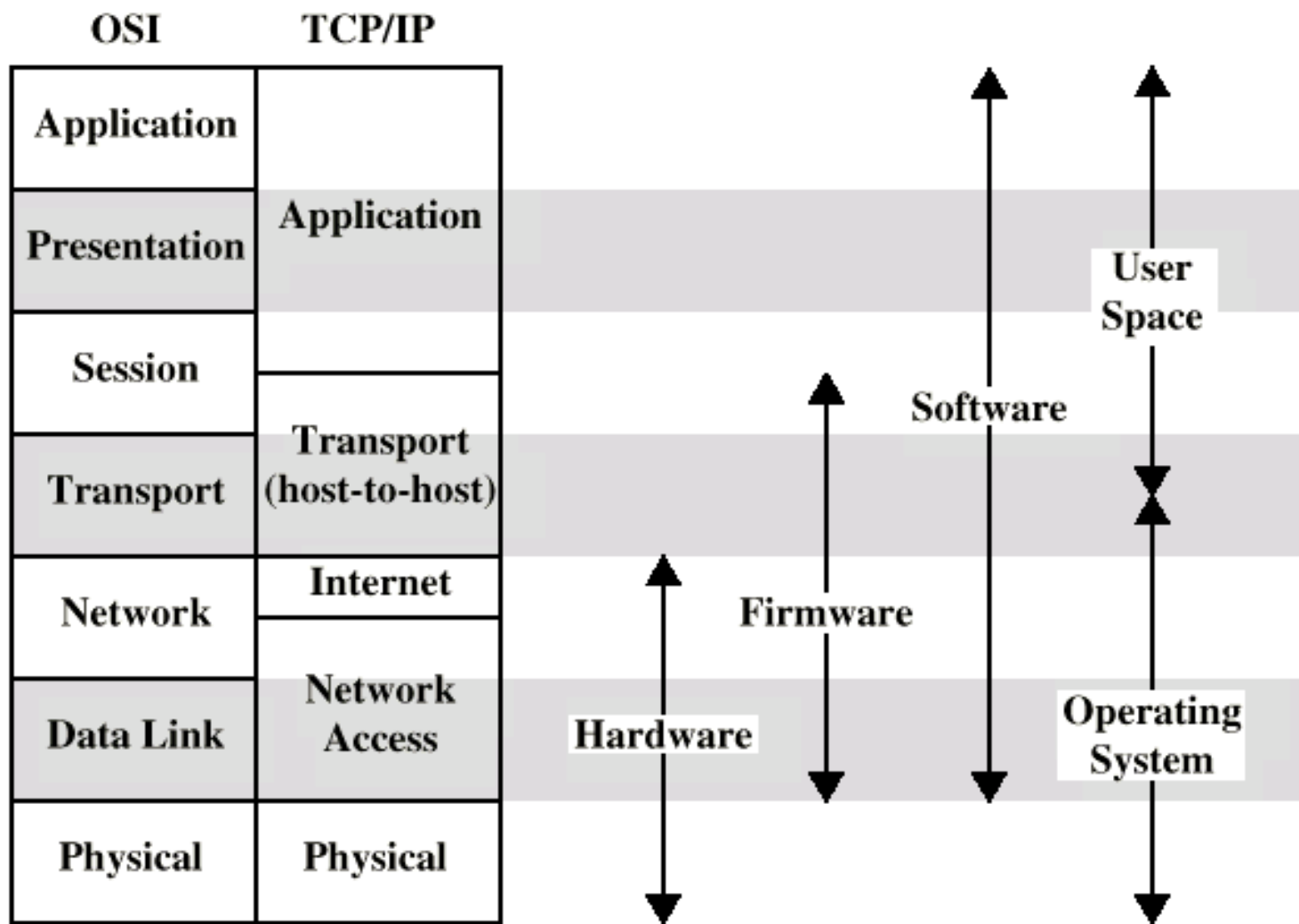


# Internett

- 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

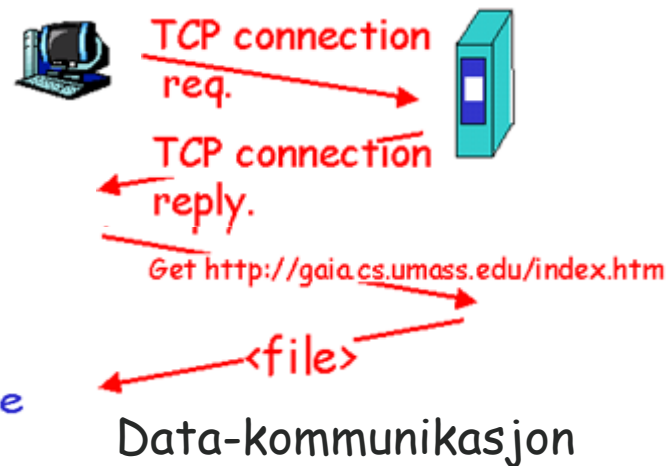
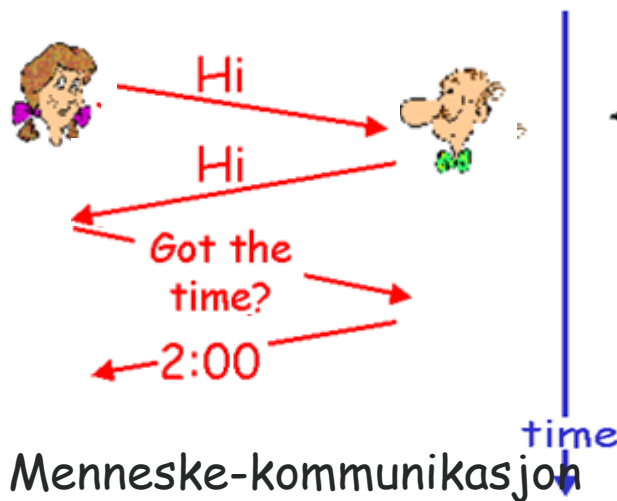
1972

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

# Protokoll

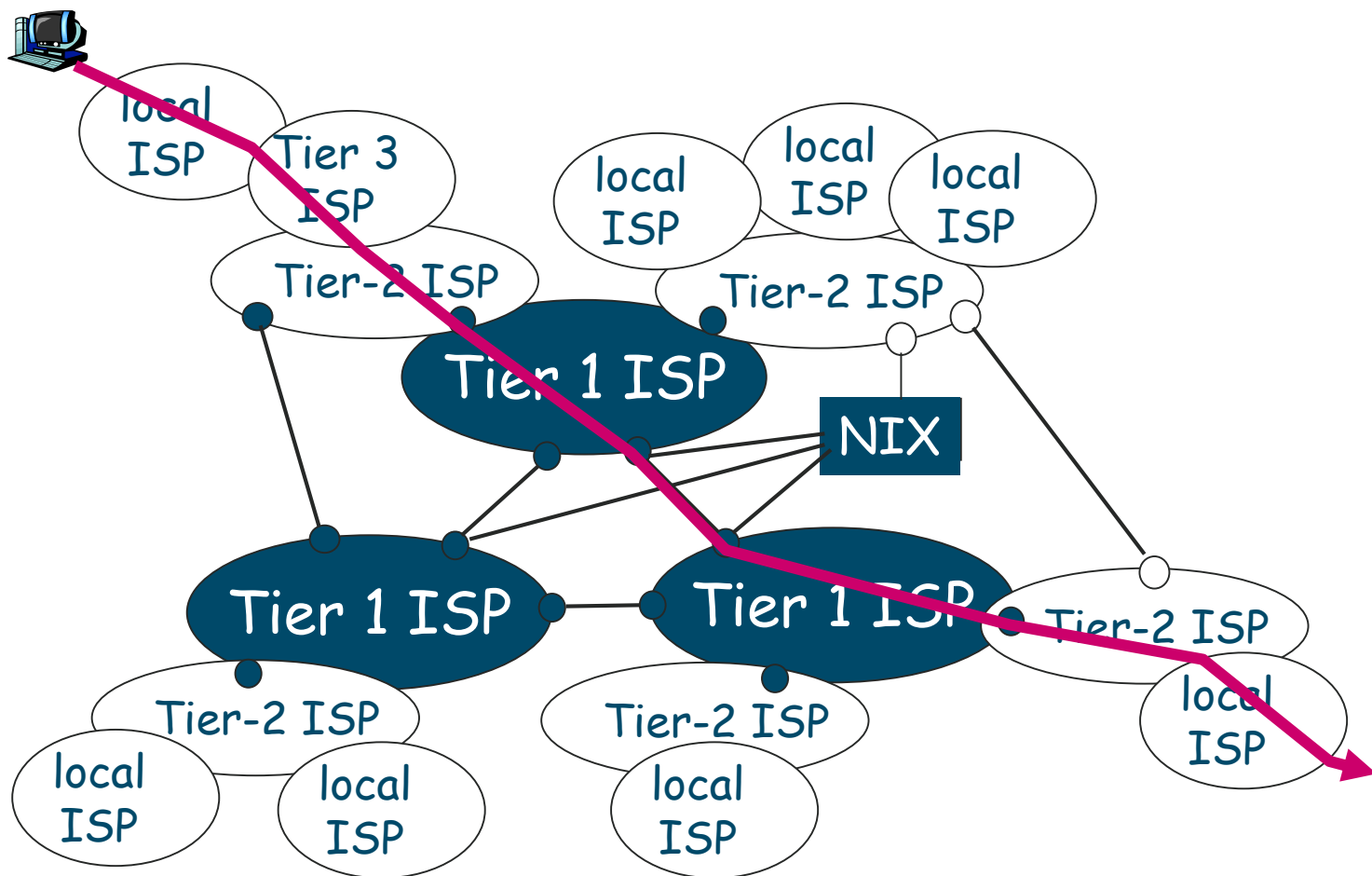
## Hva er en protokoll?

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



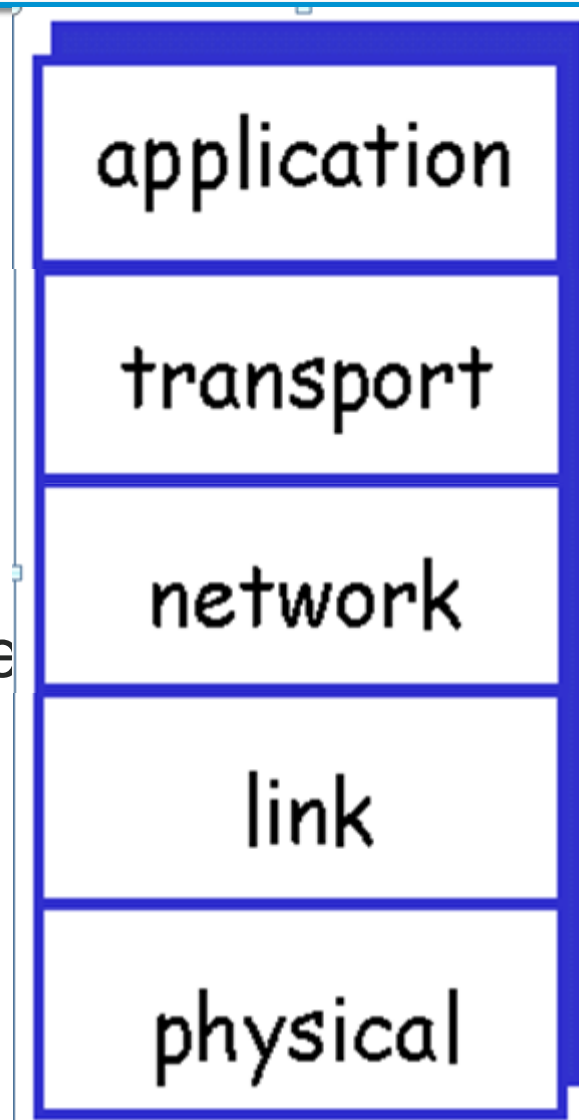
# Nettverk av nettverk!

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



## Internet protocol stacken

- Lag for nettverks**applikasjoner**
  - 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



# netstat -an

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

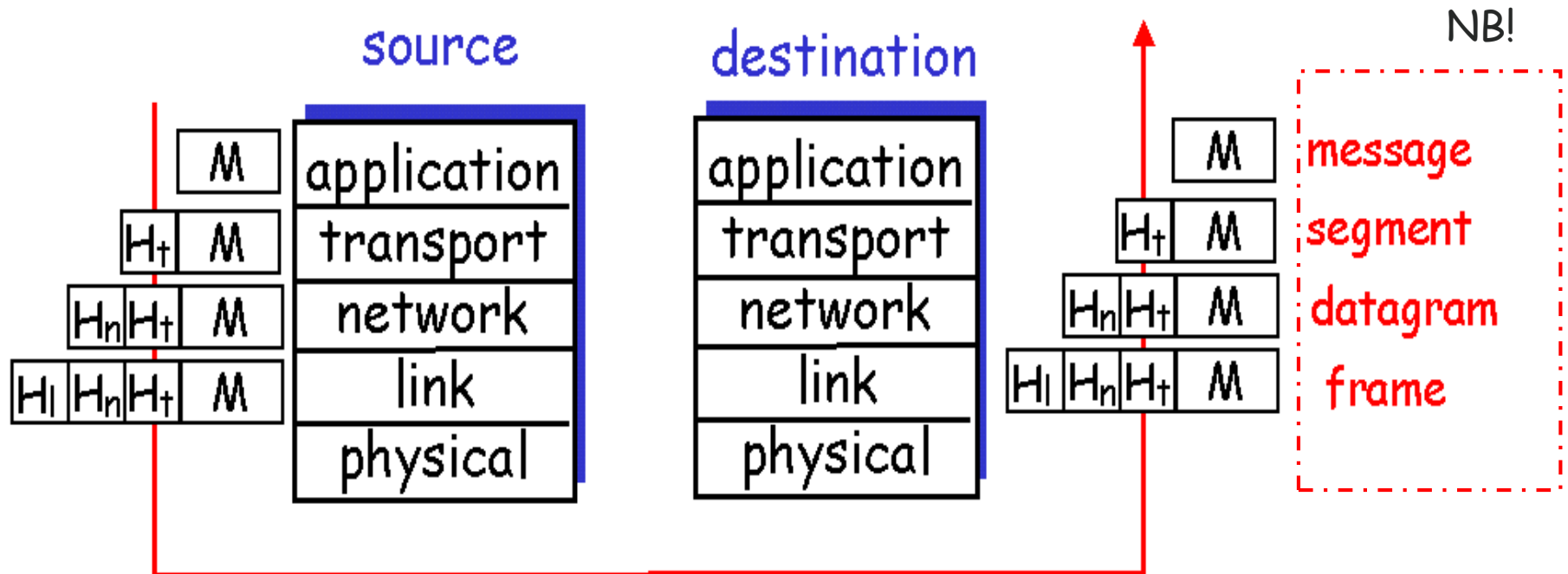
Aktive tilkoblinger

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")

# 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



# SIKKERHET?

- 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)

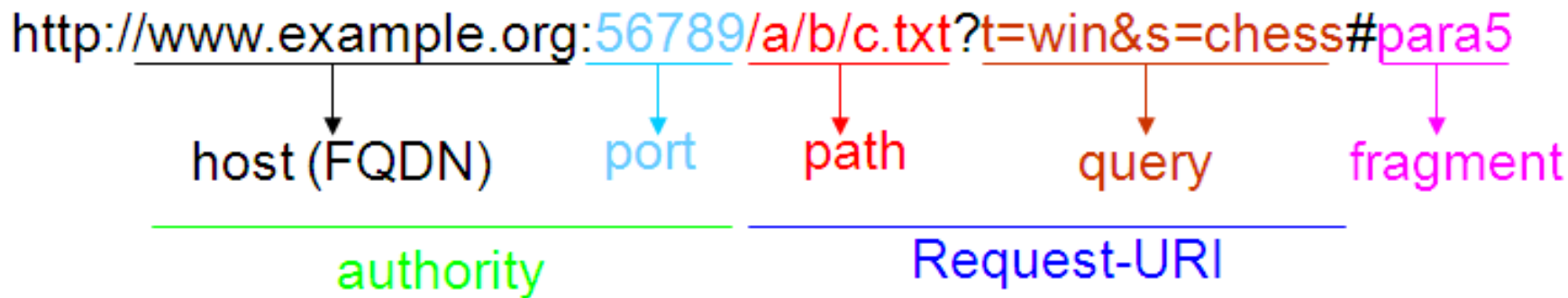
# Web-uttrykk

- 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 **anker**/posisjon innenfor ressurs ("dokument") (<**a** href= ....)

# Hoved (root) navne-tjenere

- Kontaktes av lokale tjenere ved behov
- 13 hoved navne-tjenere



# 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

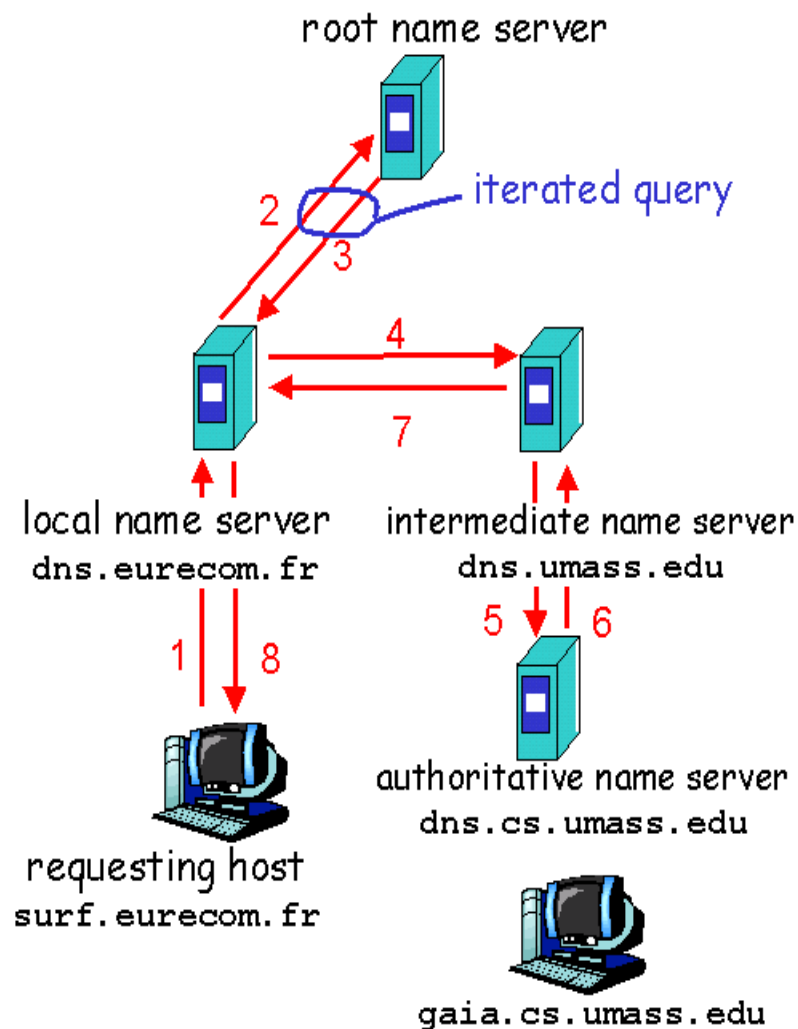
> set type=NS
> no
Server: UnKnown
Address: 2001:700:2e00::4

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
```

# Gjentatte spørringer

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



# DNS records

Distribuert database lagrer RR (resource records)

**RR format: navn, verdi, type, ttl**

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

# DNS-records: A, AAAA, PTR

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

`51.131.36.158.in-addr.arpa PTR blistog.nith.no.`

- **A**-records kopler navn med IPv4

`blistog.nith.no. A 158.36.131.51`

- **AAAA**-records kopler navn med IPv6

`blistog.nith.no. AAAA 2001:700:2e00::51`

- **CNAME**

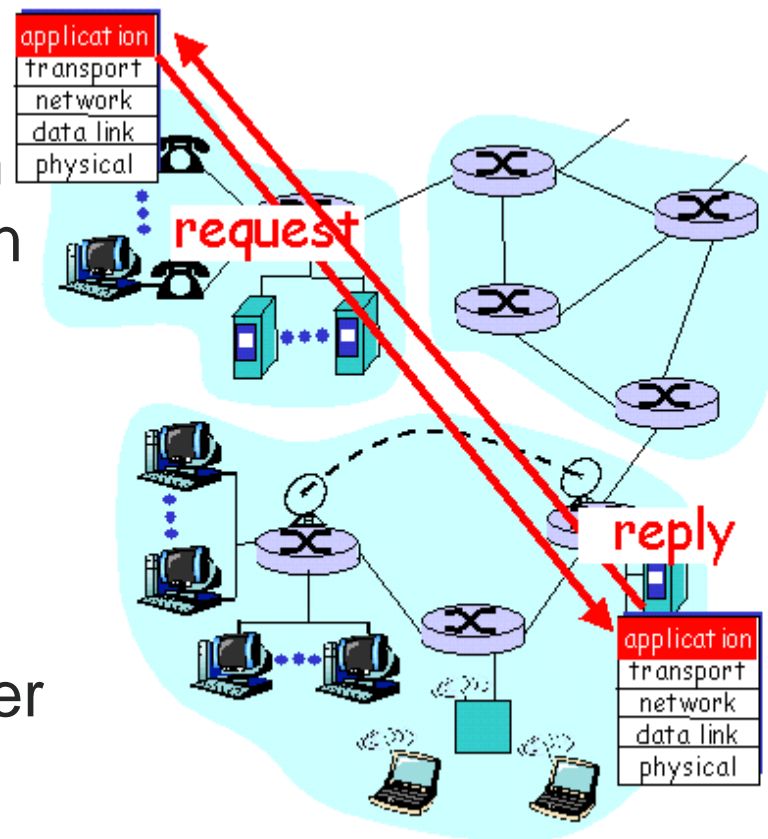
– Lar samme IP-adresse tilsvare flere ulike navn under domenet

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 nettverksskjernen 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, ...

# APPLIKASJONS- LAGET

# Klient/tjener

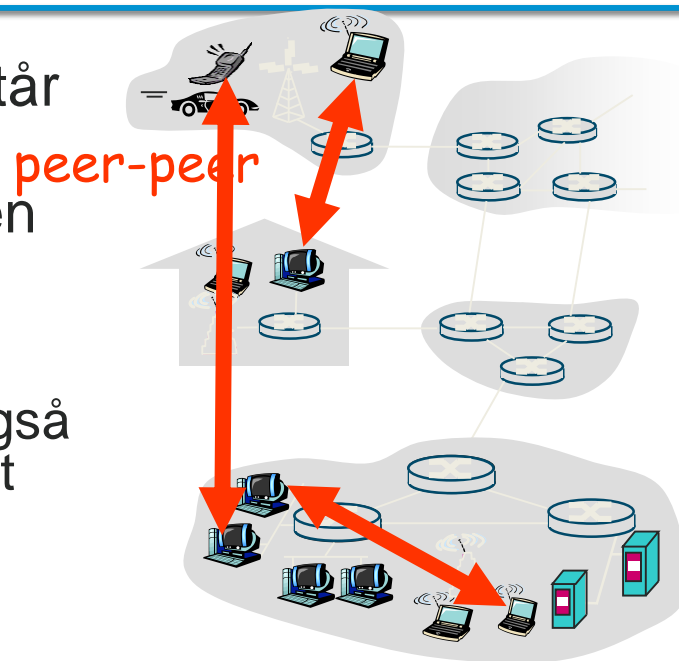
- Typisk oppsett i et nettverk
- **Klient**
  - Tar initiativet
  - Ber om en service fra tjeneren
  - På web er klienten i browseren
- **Tjener**
  - Leverer etterspurt service til klienten
  - Står «alltid på»
  - Har en fast, velkjent adresse
  - Er «flaskehals» fordi alle bruker den samme serveren/serverparken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



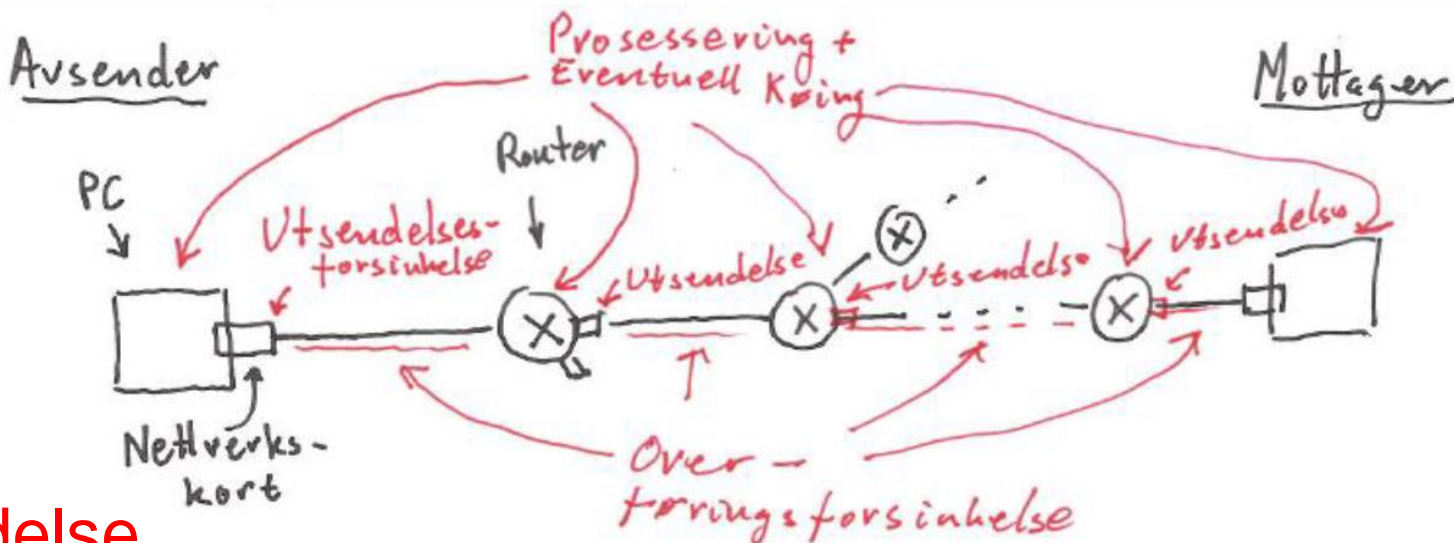


# Peer-to-Peer (P2P)

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



# 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

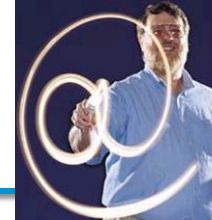
- **Prosessering**

- 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**

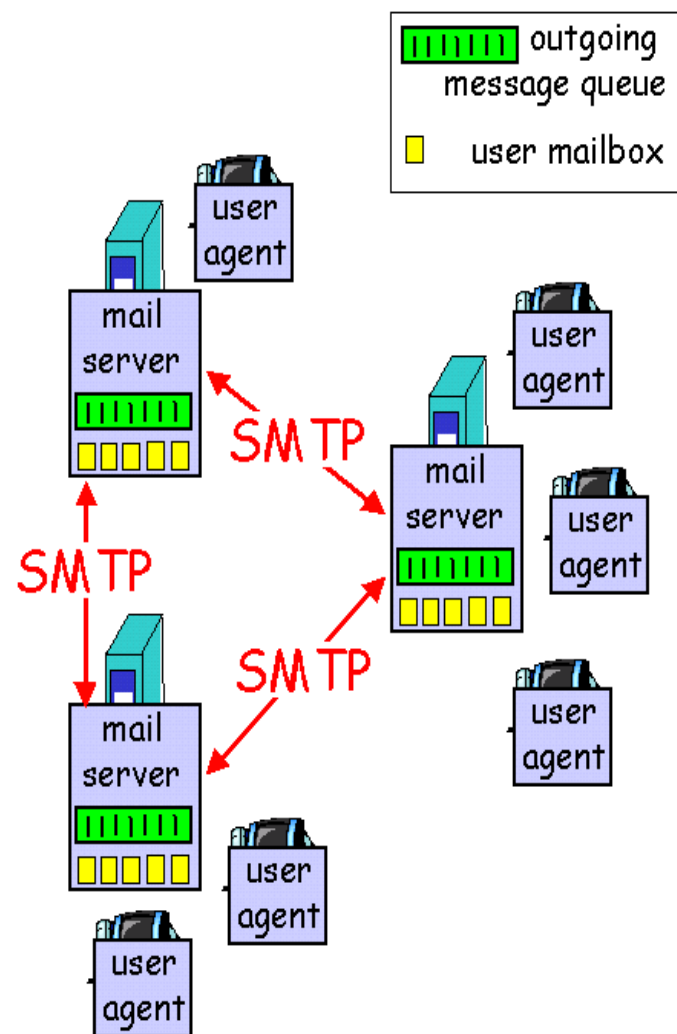
# Elektronisk post



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

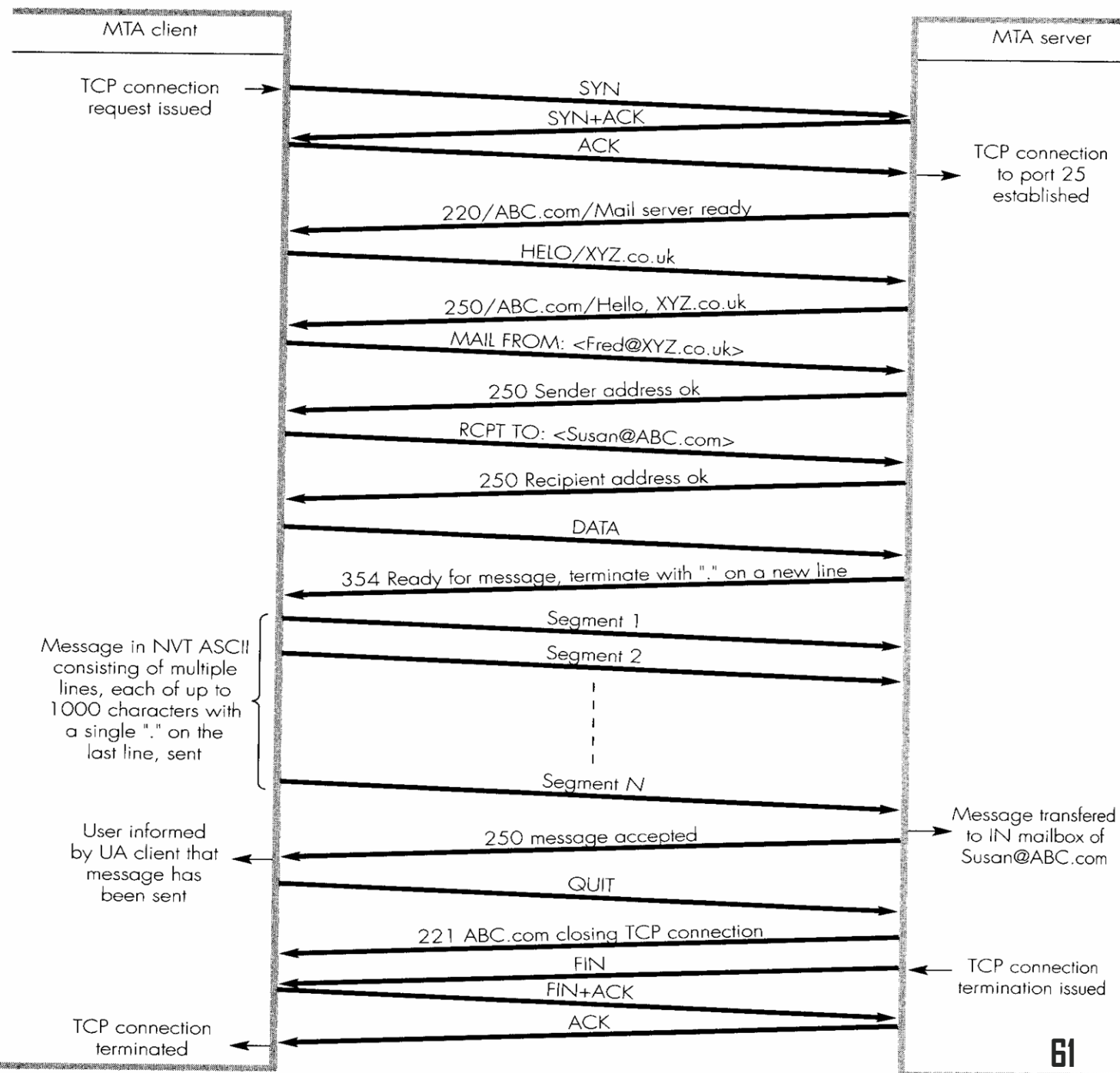
- Bruker agent

- 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

SMTP start

SMTP header

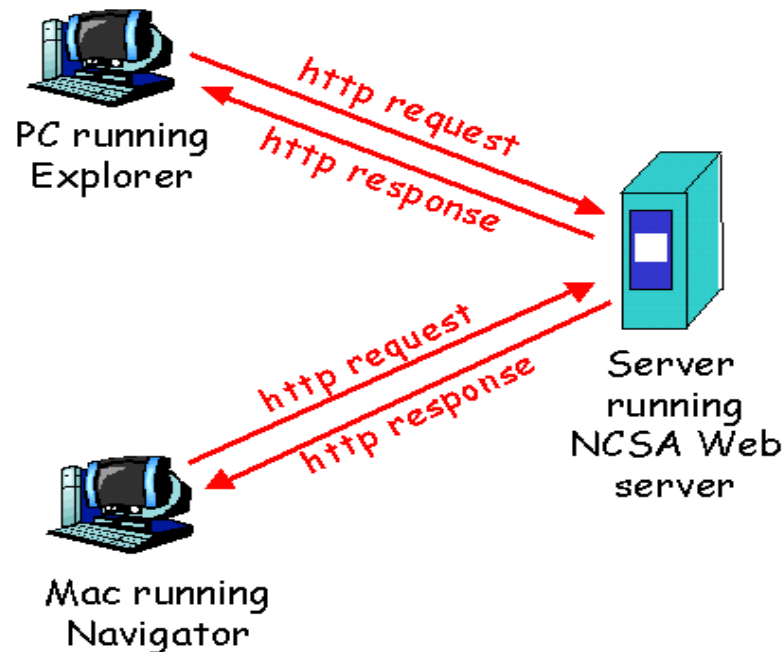
Mail header

Mail body

SMTP avslutt

# HTTP (HyperText Transfer Protocol)

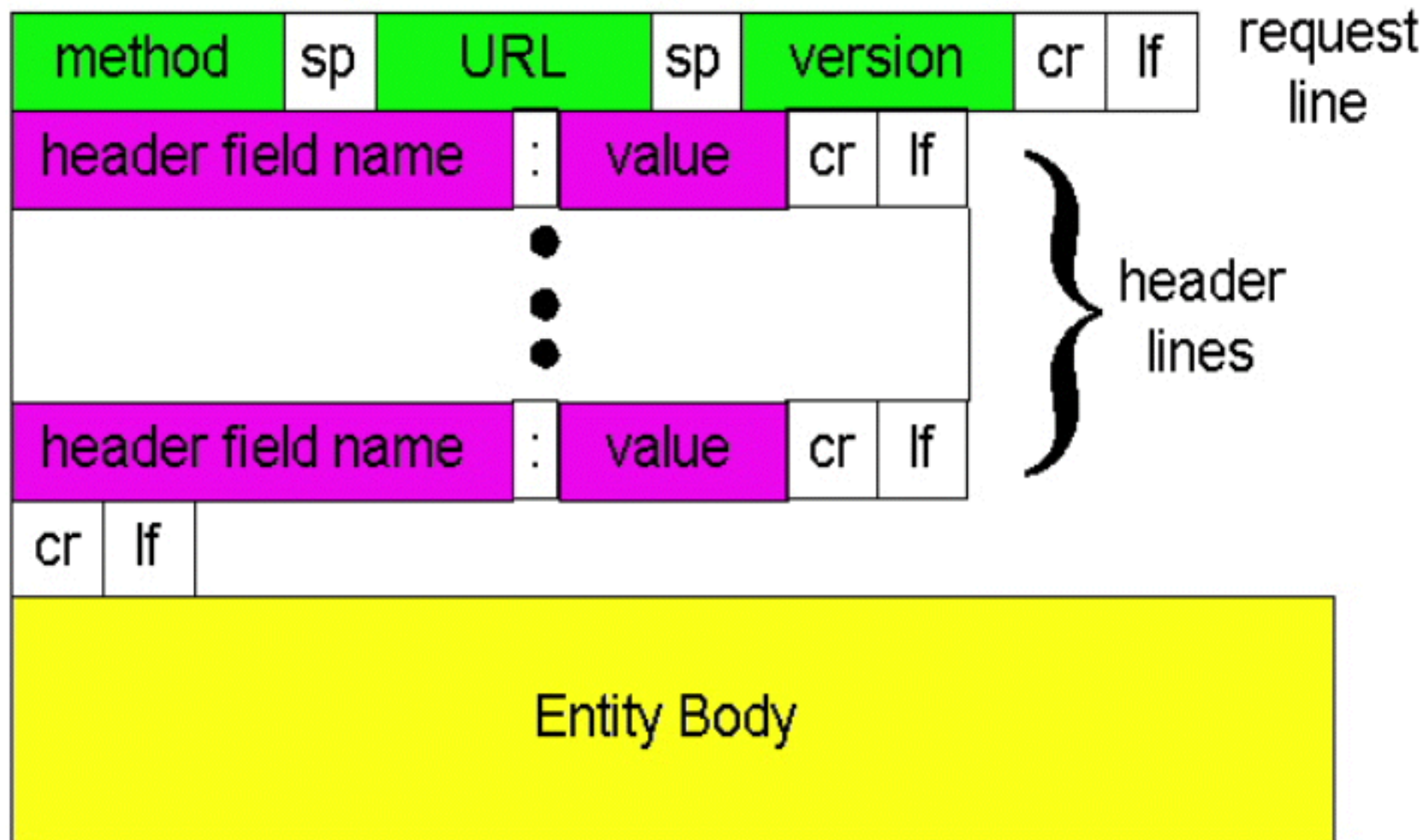
- Webens applikasjons-protokoll
  - En *enkel* filoverføringsprotokoll...
- Klient/tjener modell
  - Klienten spør etter, mottar og viser web "objekter"
  - Tjeneren sender objekter på etterspørsel





# 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

# HTTP 1.1 Meldingsformat

request line  
(GET, POST,  
HEAD commands)

GET /somedir/page.html HTTP/1.1

header  
lines

Host: www.someschool.edu

User-agent: Mozilla/4.0

Connection: close

Accept-language: fr

Carriage return  
line feed  
indicates end  
of message

(extra carriage return, line feed)

status line  
(protocol  
status code  
status phrase)

HTTP/1.1 200 OK

Connection close

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 ...

Obligatorisk

# HTTP svar statuskoder

- 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)

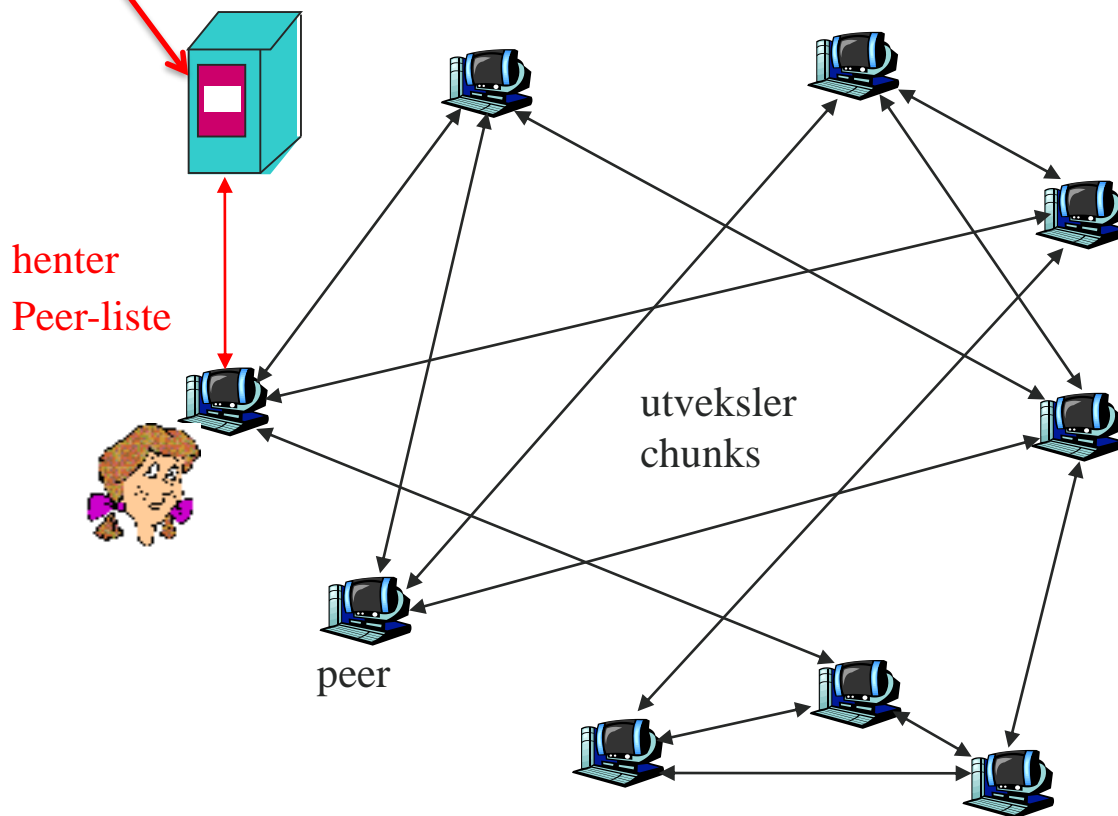
4xx = Client Error

5xx = Server Error

# Fildistribusjon: BitTorrent

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

torrent: gruppe av “peers” som utveksler “chunks” av en fil



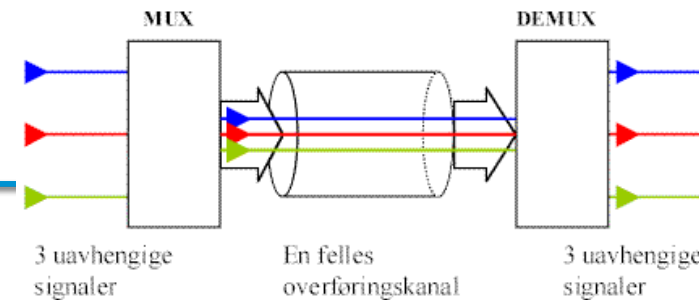
# Transportlaget: Agenda

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

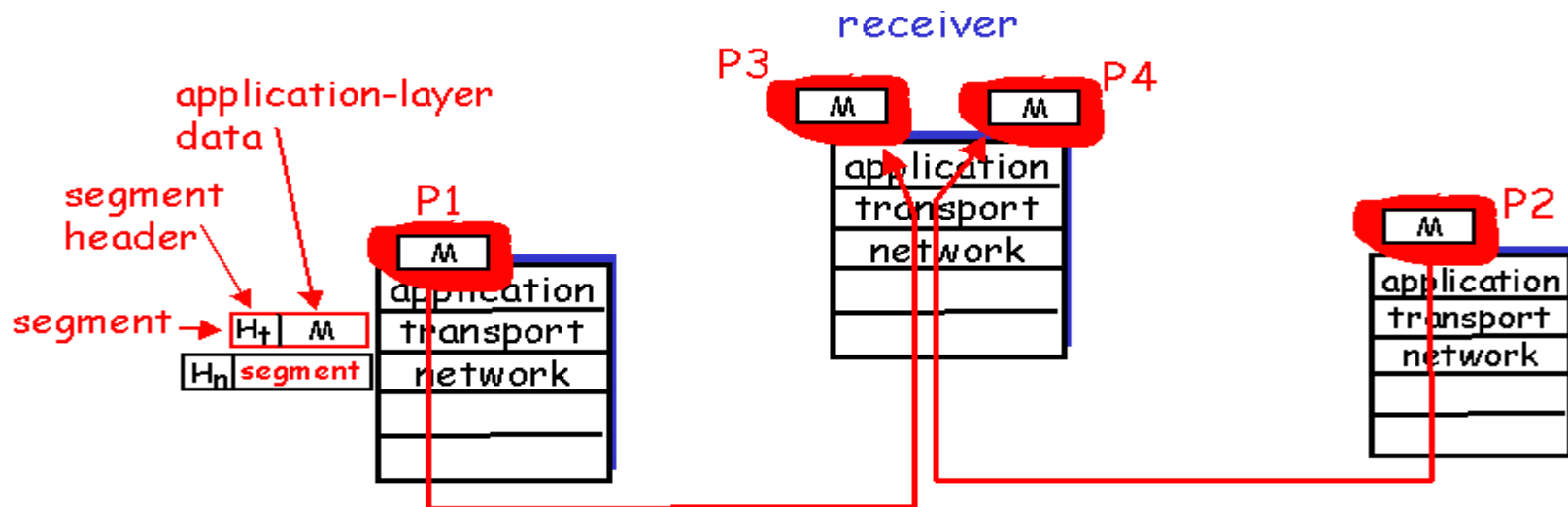
# Transport-lag protokoller

- 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



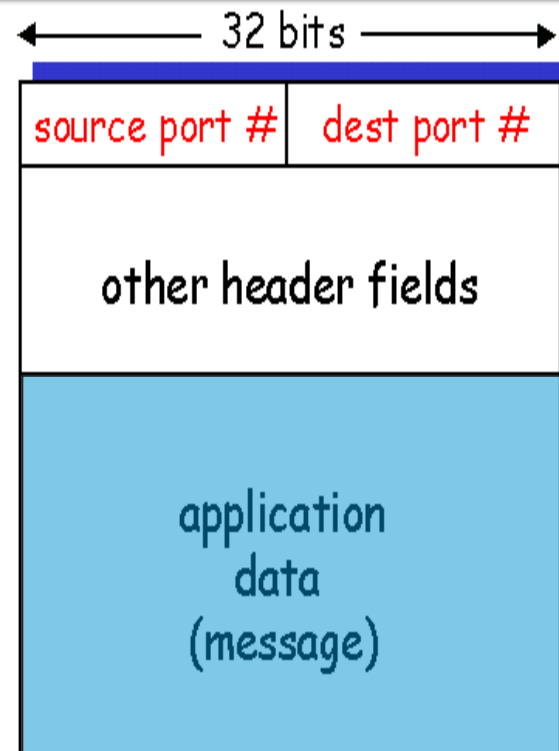
- **Segment**
  - Data-enhet som utveksles mellom transportlagene
  - TPDU (Transport Protocol Data Unit)
- **Demultipleksing**
  - Levere motatte segmenter til riktig **prosess**





# Multipleksing <- portnummer

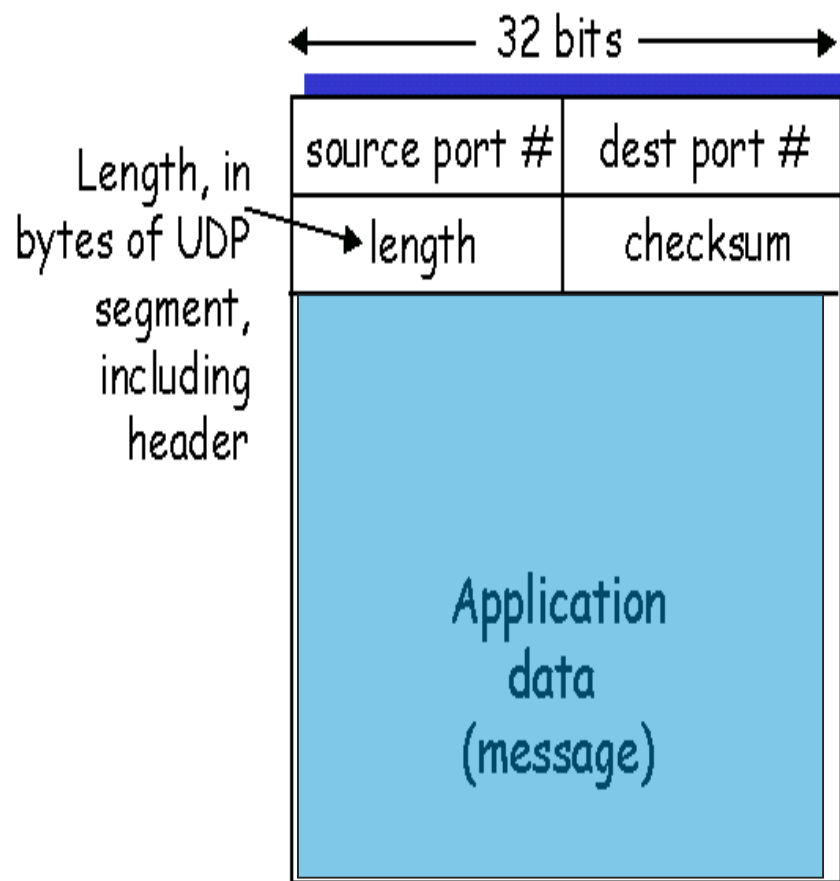
- Samler data fra applikasjons-prosesser og pakker disse med et hode (header)
- Hodet inneholder **senders** og **mottakers portnummer**
- Portnummer = **16 bit** unsigned heltall
- Portene **0-1023** er «**well known**» (RFC 1700)
  - Secure Shell: port **22**
  - SMTP: port **25**
  - DNS: port **53**
  - 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

# UDP

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



UDP segment format

# UDP sjekksum

- 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  $\text{sum} = 1111\ 1111\ 1111\ 1111 \Rightarrow$  alt OK
- I beste fall gir dette bare en **indikasjon** på om feil er oppstått under overføringen

# Ex: Internet sjekksummen

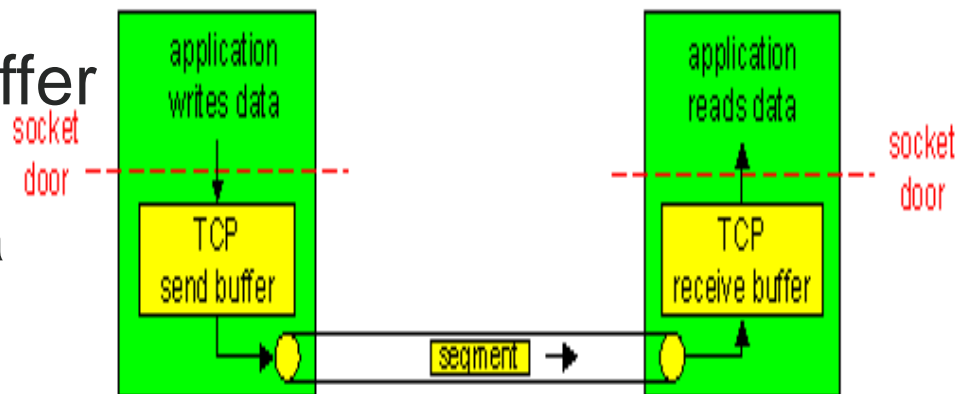
Merk: Mente i mest signifikante posisjon legges til LSb (Minst signifikante bit) !

Ex: To 16 bit deler av samlet pakke legges sammen

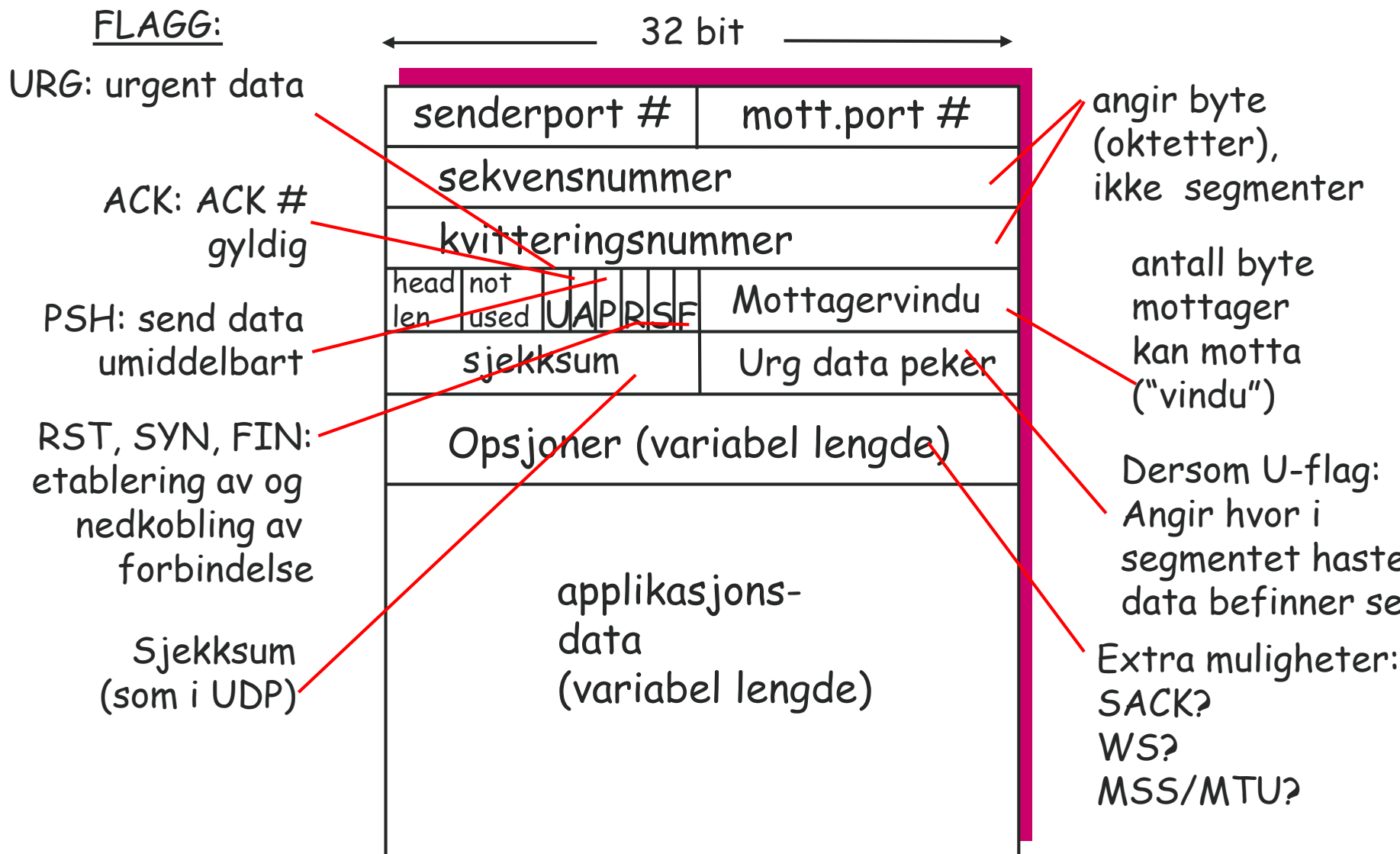
	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
<hr/>																
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1
	<hr/>															
sum	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

# TCP (Transmission Control Protocol)

- Punkt til punkt
  - En avsender, en mottaker
- Pålitelig, ordnet byte-strøm
- Pipeline
  - Flyt- og metnings-kontroll bestemmer vindu-størrelse
- Avsender og mottaker-buffer
- Full duplex data
  - Begge kan sende og motta samtidig
- Forbindelses-orientert
  - Handshake før dataoverføring
- Flytkontroll
  - Avsender drukner ikke mottaker



# Oppbygging av TCP-header

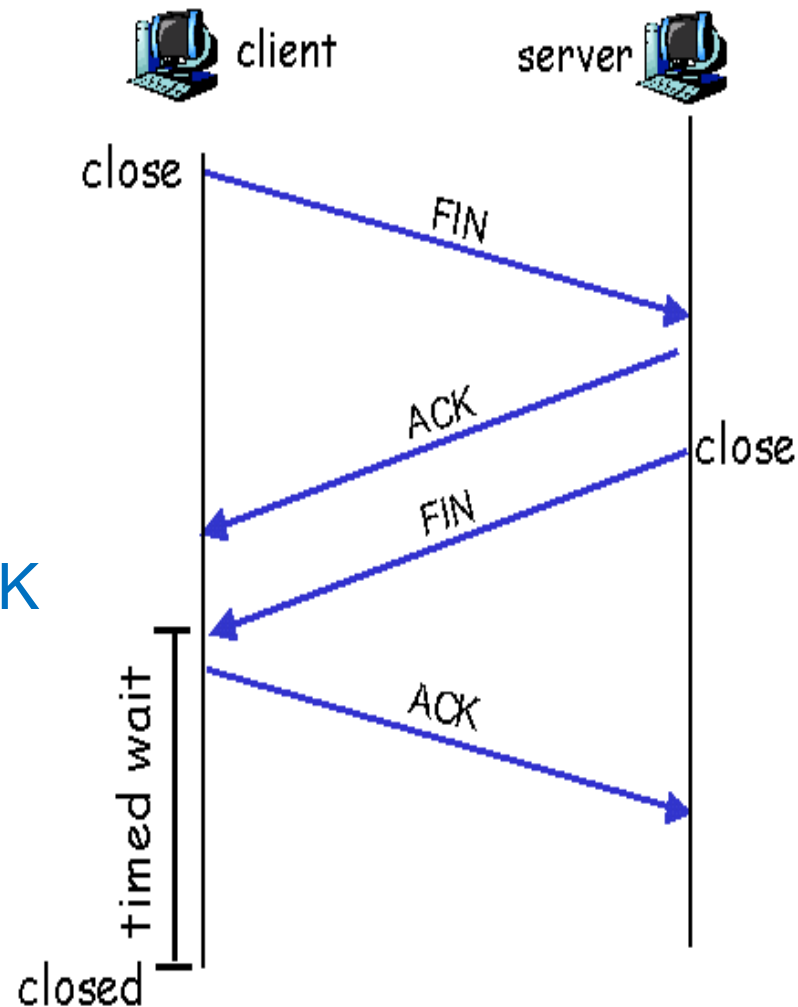


# TCP: Oppstart av forbindelsen

- 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



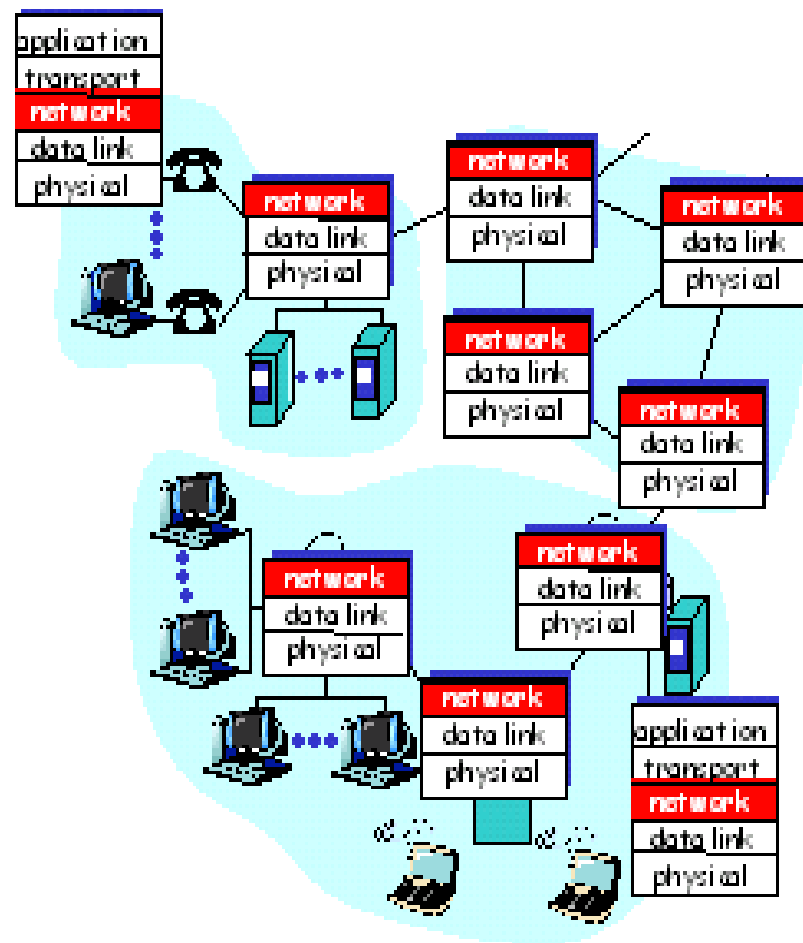


# Oversikt over nettverkslaget

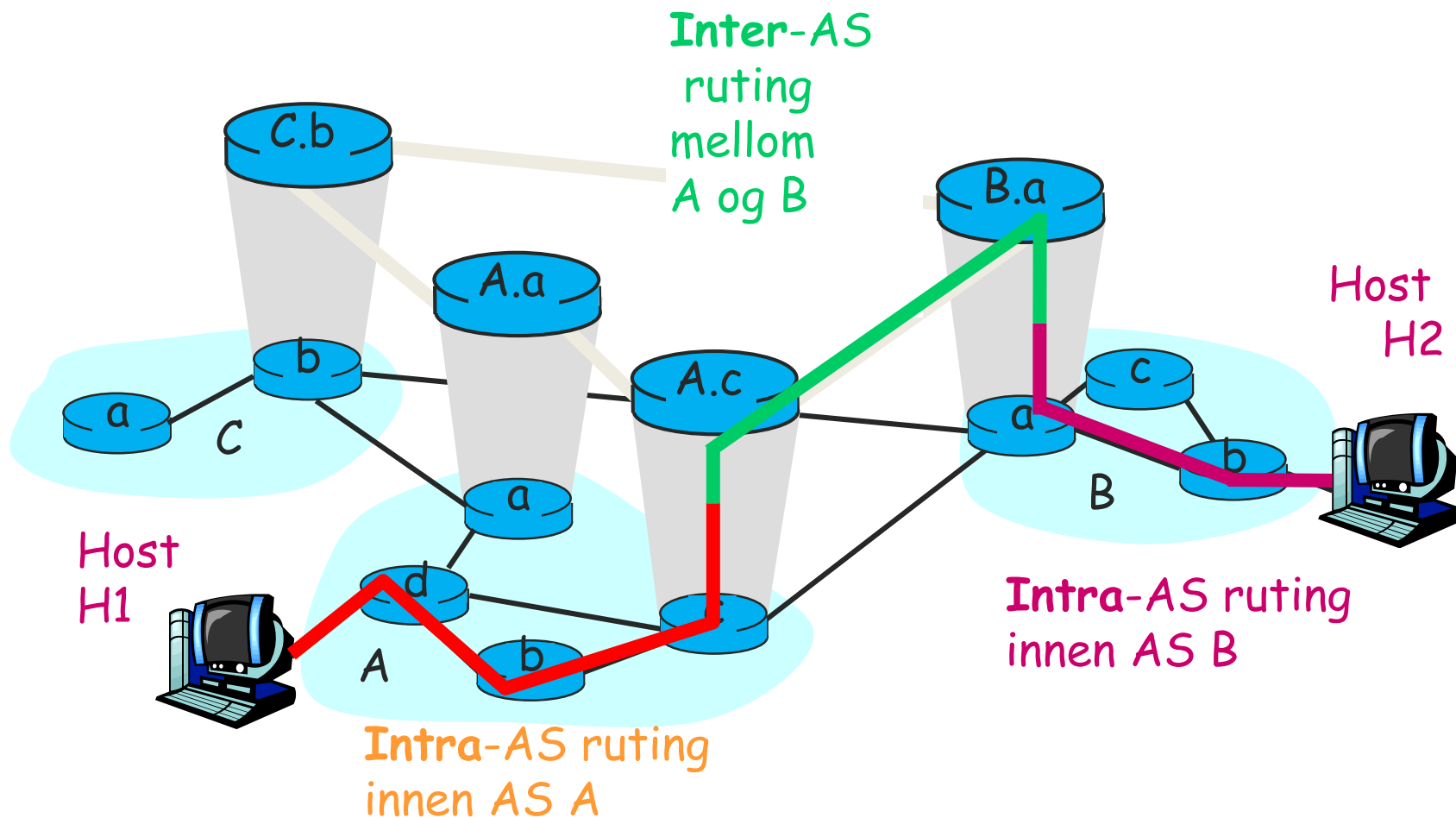
- Nettlagets oppgaver
  - Prefix-/datagram-svitsjing på nettlagsnivå
- IPv4
  - headeren
  - IP-adresser og prefix-routing
    - IP fragmentering
  - Litt om DHCP
- ICMP
- NAT
- IPv6
- Litt om AS og routing i LAN, WAN og stamnett

# 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



# Intra-AS og Inter-AS routing

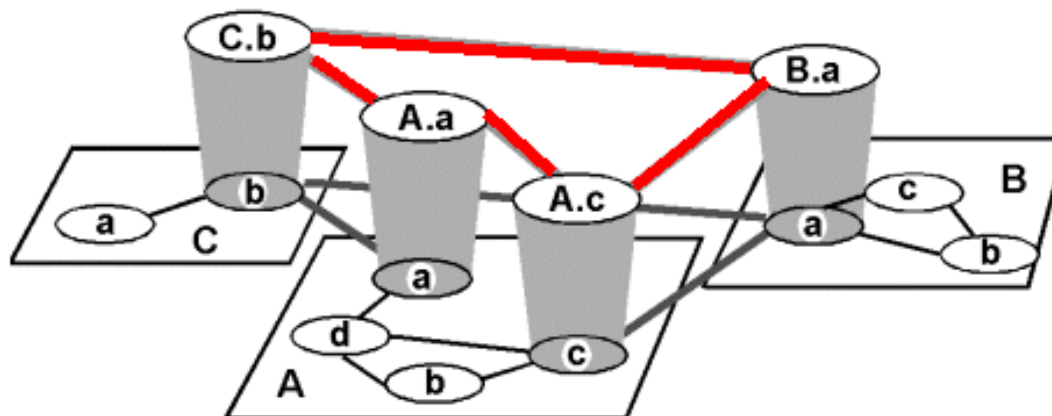


# Intra-AS rutning

- 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

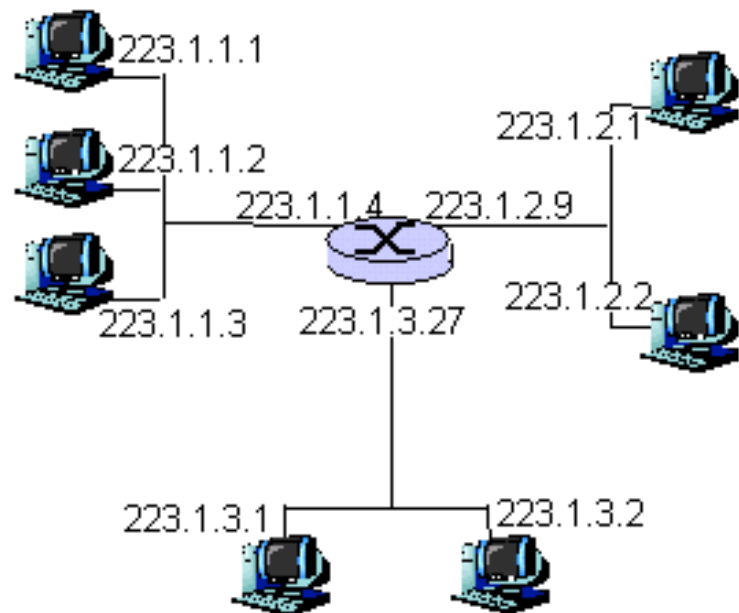
# Internett **inter**-AS ruting

- 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



# 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



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

# 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}:
```

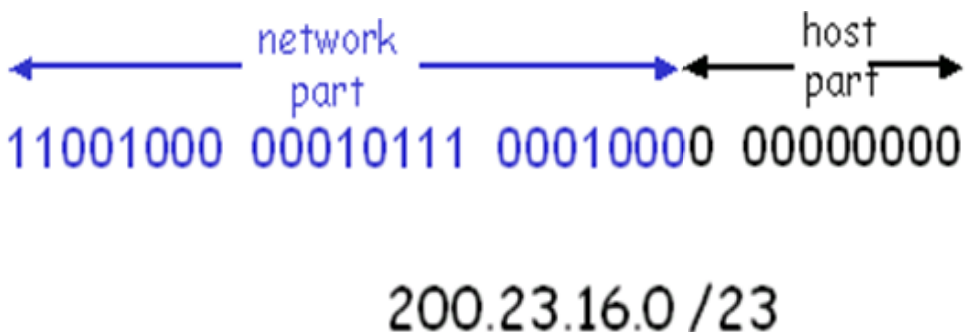
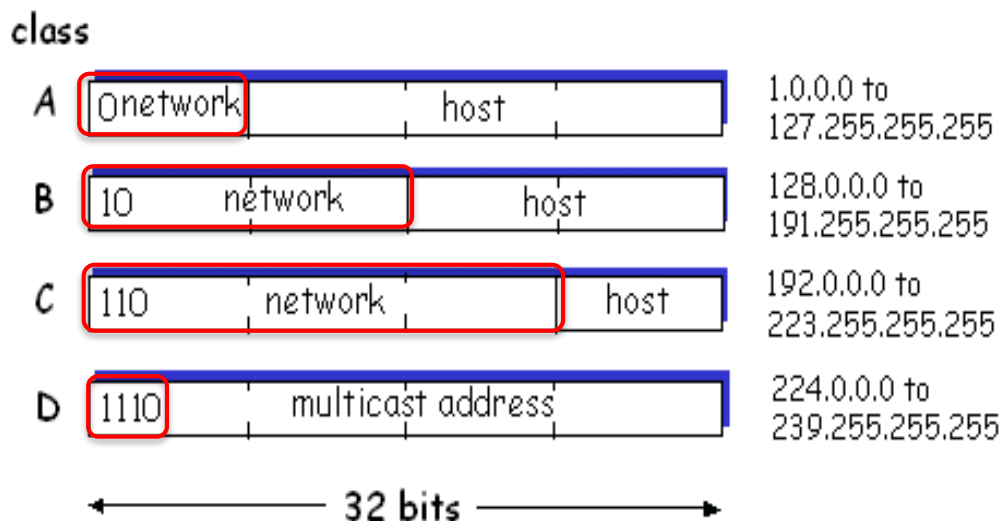
```
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix  . : 
```

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

Teknikker for  
å sende IPv6  
gjennom IPv4  
nettverk

# 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





# IPv4 adresser: tildeling

- 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 nettverks-paramerte (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

# DHCP

- 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

IP protokollversjon

header lengde  
(byte)

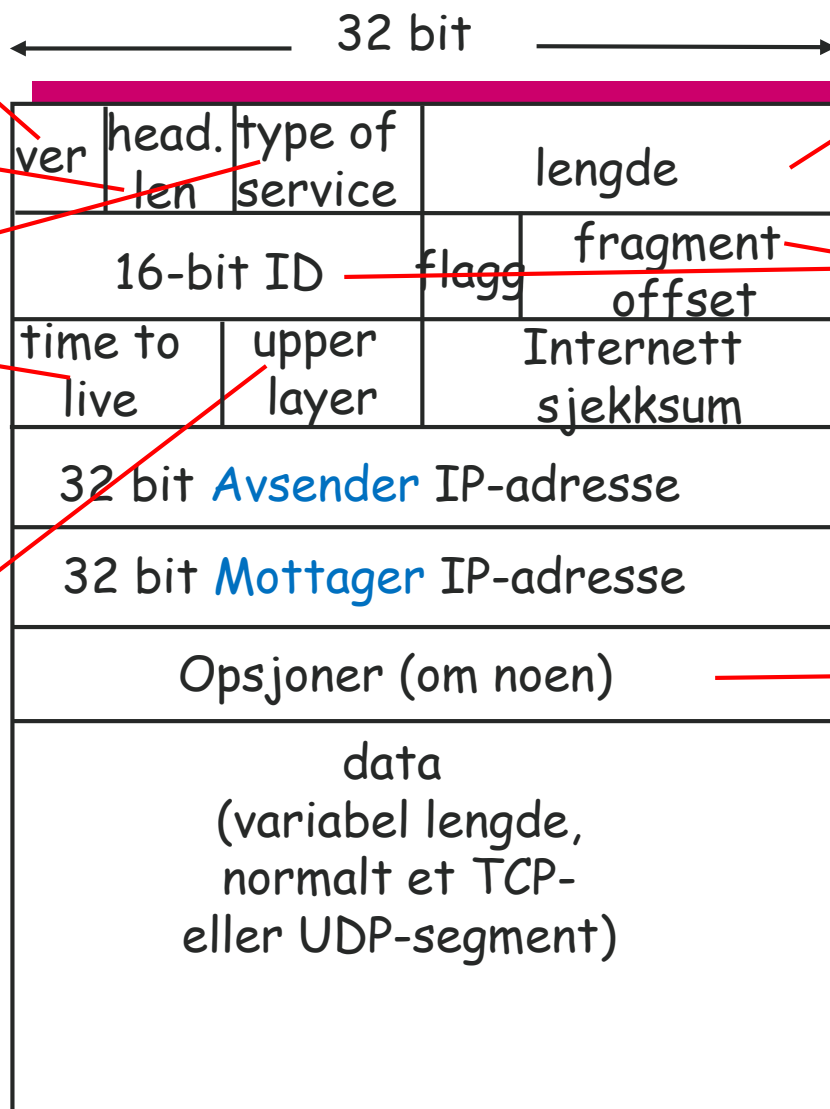
"type" data

maks antall  
gjenværende hopp  
(dekrementeres i  
hver router)

protokollen som skal ha  
nyttelasten (TCP, UDP)

hvor mye overhead med  
TCP?

- ❑ Min 20 byte for TCP
- ❑ Min 20 byte for IP
- ❑ = 40 byte + app. lags overhead

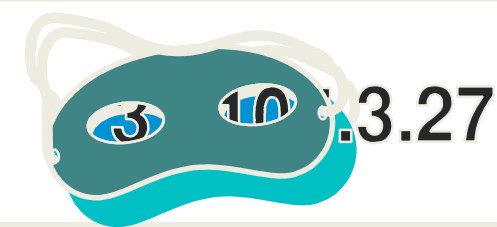


total datagram-  
lengde (byte)

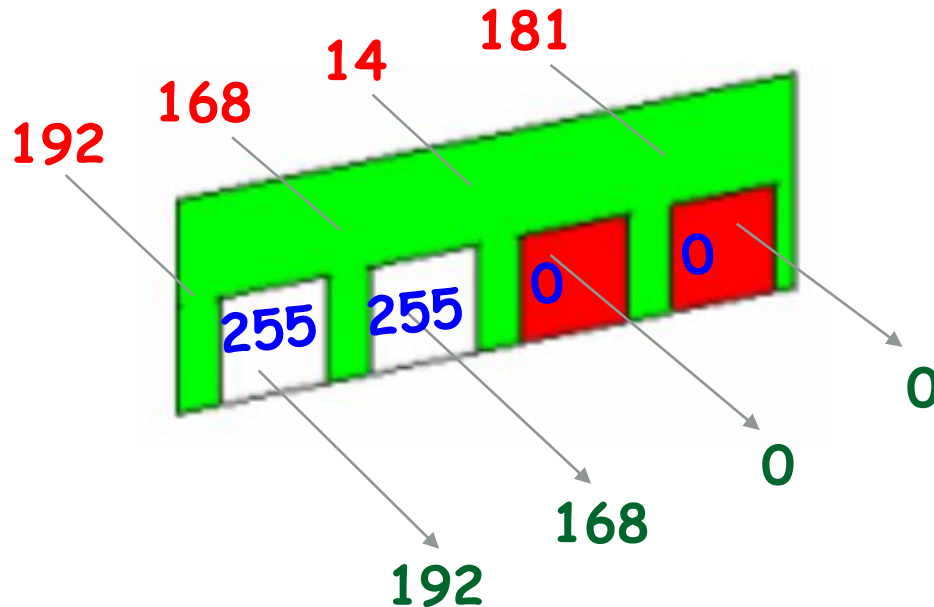
for  
fragmentering/  
sammensetting

F. eks.  
tidsstempel,  
record route,  
spesifisere  
liste av rutere  
man skal innom

# 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



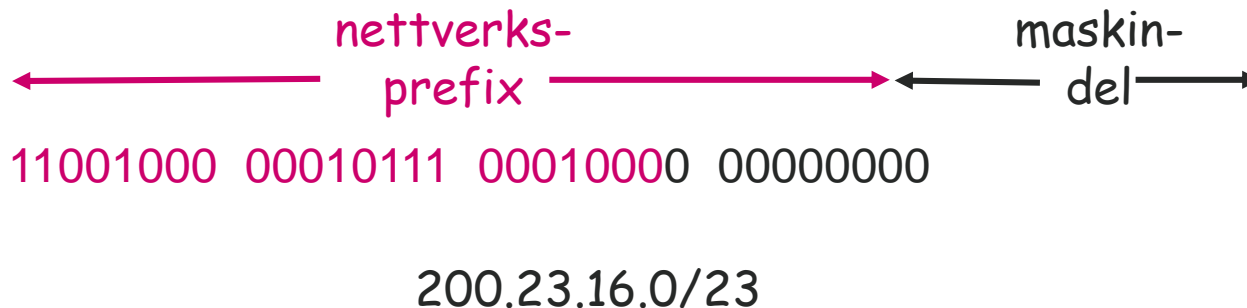
$$\begin{array}{r}
 192.168.14.181 \\
 \text{AND } 255.255.0.0 \\
 \hline
 = 192.168.0.0
 \end{array}$$

# IP-adresse & Nettmaske = IP-Nettverk

- Maskiner/adaptere 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.

# IP-adressering: CIDR

- “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
<hr/>										
	10	.	21	.	0001	10	00	.	0000	0000
22 bit til <b>prefix</b> , 10 bit til <b>host</b>										

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) – 127.0.0.1
- 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
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

[RFC 1918](#)

# 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

# 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

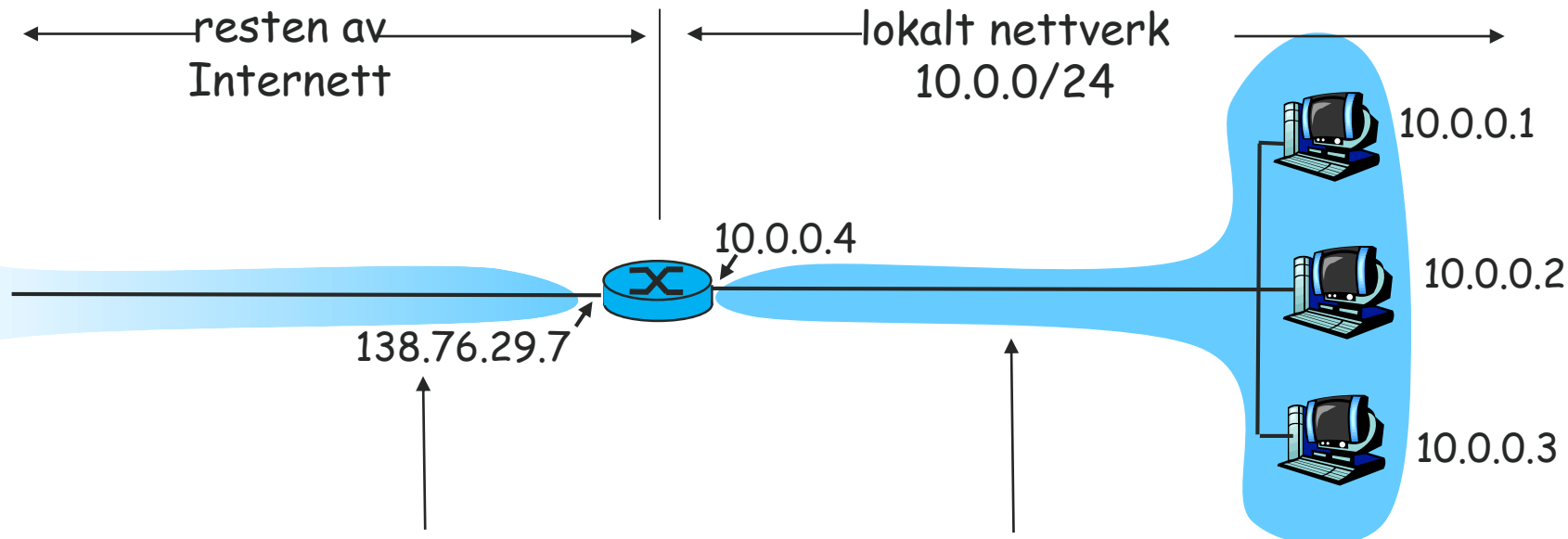
Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
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

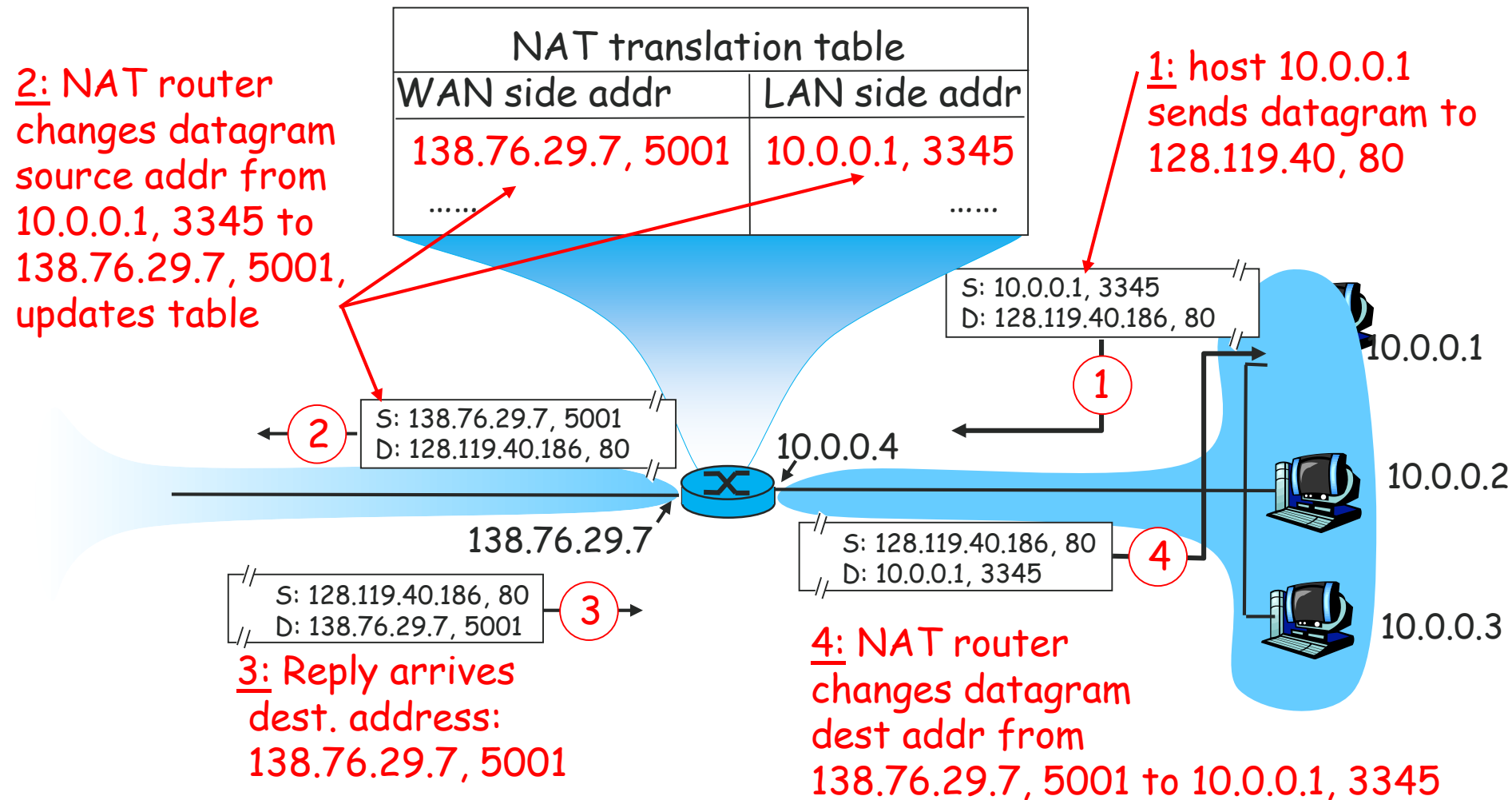
Gateway med  
NAT-server



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

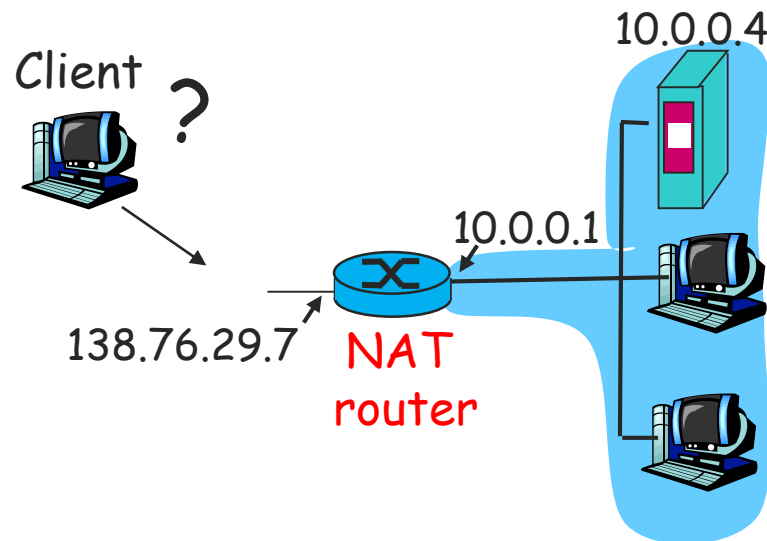
Datagram avsender eller  
mottager innenfor dette nettverket  
har 10.0.0/24 adresse for  
kilde, mål (som vanlig)  
Bruker (typisk) PRIVATE ADRESSER  
(10.x.x.x, 192.168.x.x,...)

# NAT: Network Address Translation



# NAT traversering problemet

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

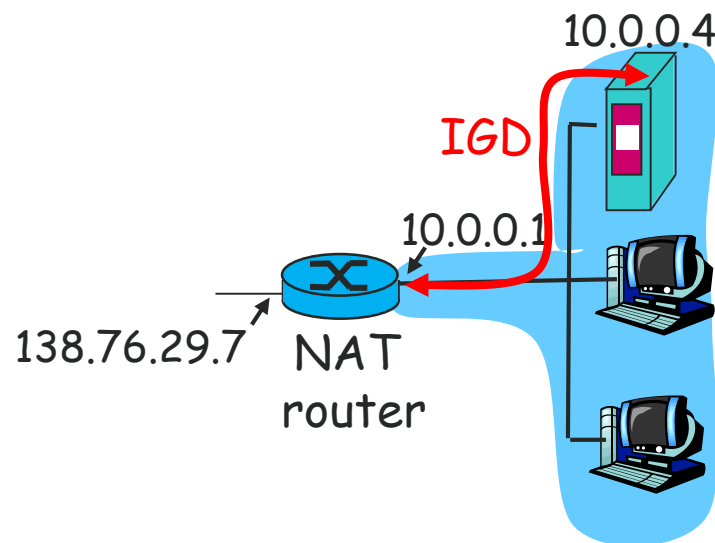




# NAT traversal problem

- 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"



# IPv6 header

- Version: 0110
- Traffic Class
  - Prioritering innad i en datastrøm
- Flow Label
  - QoS
  - Sikre differensiering 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
  - Tilsvarende TTL slik det ble praktisert IPv4
- DATA

ver	pri	flow label	
payload len		next hdr	hop limit
source address (128 bits)			
destination address (128 bits)			
data			

# IPv6 notasjon

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

Ethernet-kort eth0:

```
Tilkoblingsspesifikt DNS-suffiks : oslo.nith.no
Beskrivelse . . . . . : Intel(R) PRO/1000 PL Network Connecti
on
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-adapternr. (ikke egentlig del av standarden)
- fe80:: = fe80:0000:0000:0000 = nettprefix
- :: er minimum fire nuller, her 12 ut fra resten av adressen
- 020e:7bff:fe98:f8a1 er basert på MAC-adressen

- 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

# 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

# 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

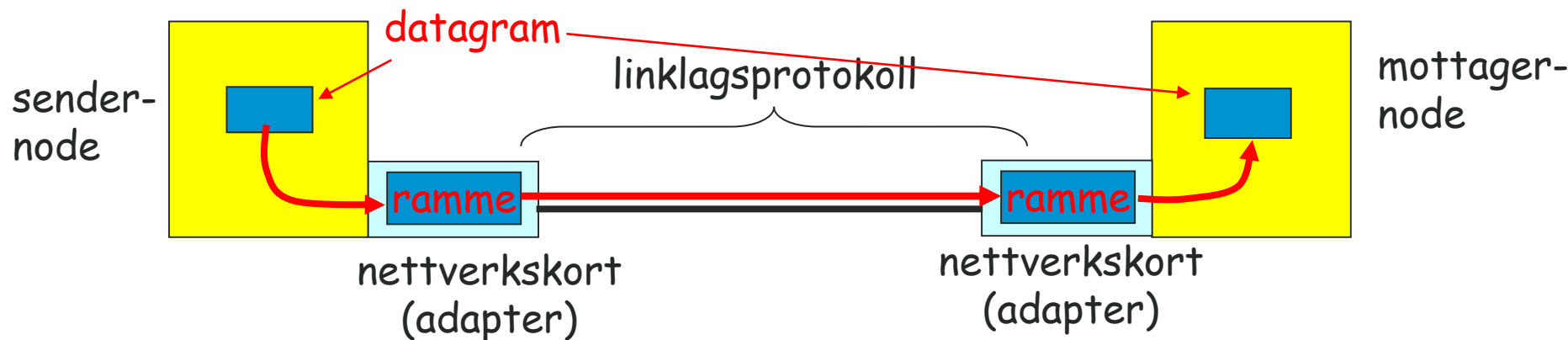
## transportanalogi

- 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 naboroder**
  - 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



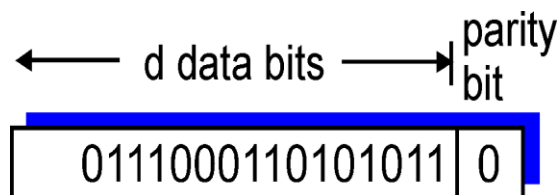
- linklaget implementert i nettverkskort (NIC)
  - Ethernet-kort, 802.11-kort e.l.
- senderside:
  - innkapsling av datagram i en ramme
  - adderer bit for deteksjon av bitfeil, (sekvensnummer, flytkontroll etc.)
- mottagerside
  - ser etter bitfeil, re-transmisjon, flytkontroll etc.
  - ekstraherer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



# Paritetssjekk

## Ett-bits paritet:

Kan oppdage dersom ett bit er feil



## Like paritet:

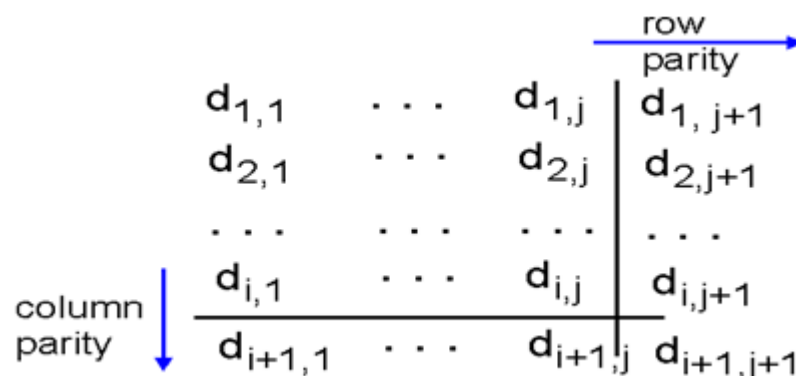
Det totale antall enere (inkl paritetsbit) skal være et **partall**

## Odde paritet:

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

## Todimensjonale paritetsbit:

Kan oppdage og rette dersom ett bit er feil



1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

*no errors*

1	0	1	0	1	1
1	0	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

parity error

*correctable  
single bit error*

# 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.

# 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

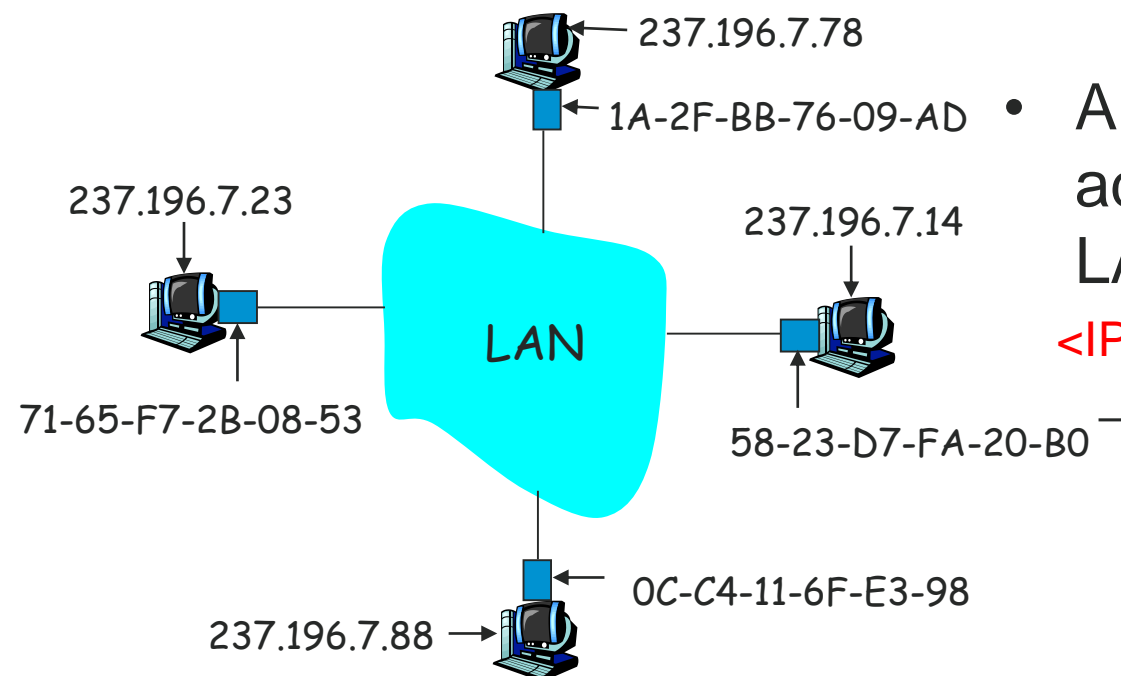
Hvordan finne MAC-adressen til en node man kjenner IP-adressen til?

- Hver IP-node (maskin og ruter) på et LAN har en **ARP**-tabell/cache

- ARP-tabell: IP/MAC adressemapping for noen LAN-noder

<IP-adresse; MAC-adresse; TTL>

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

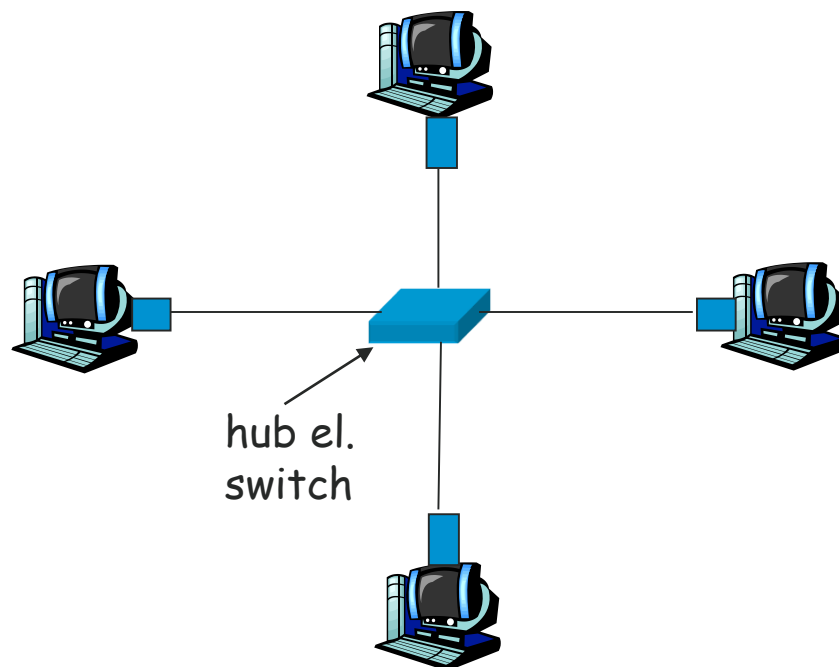


# ARP (Address Resolution Protocol)

- 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

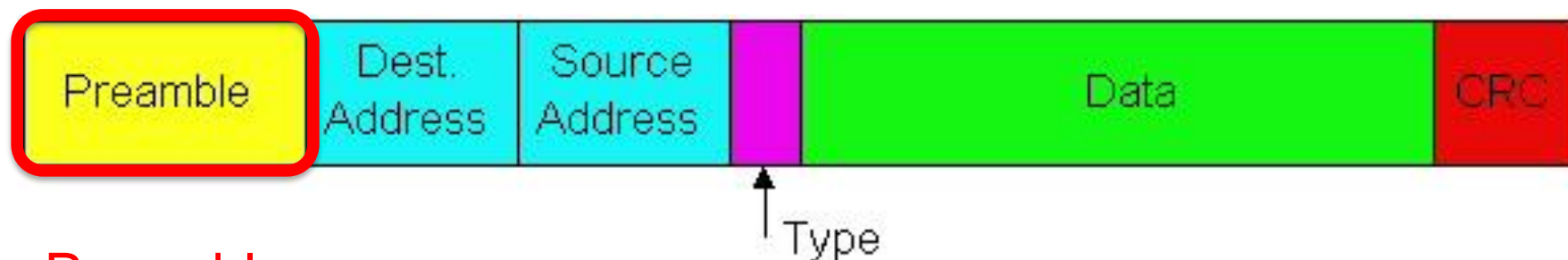
# Stjernetopologi

- 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)



# Ethernets rammestruktur

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

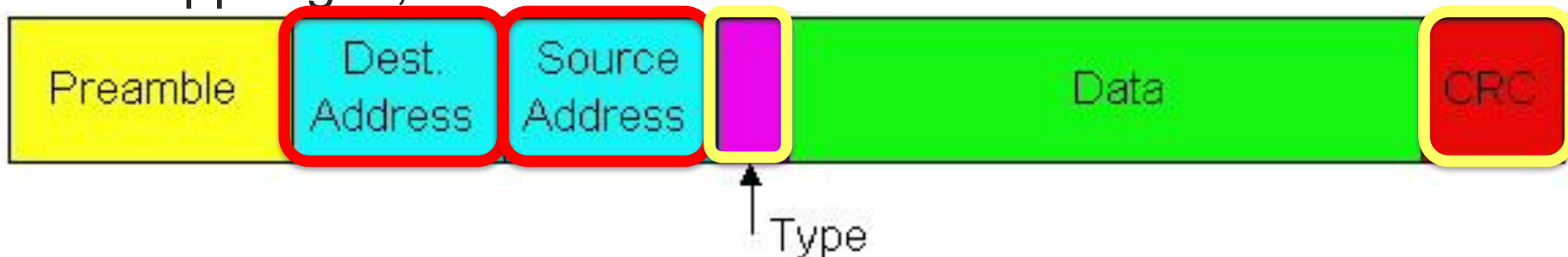


## Preamble:

- 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





# 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**)

# 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

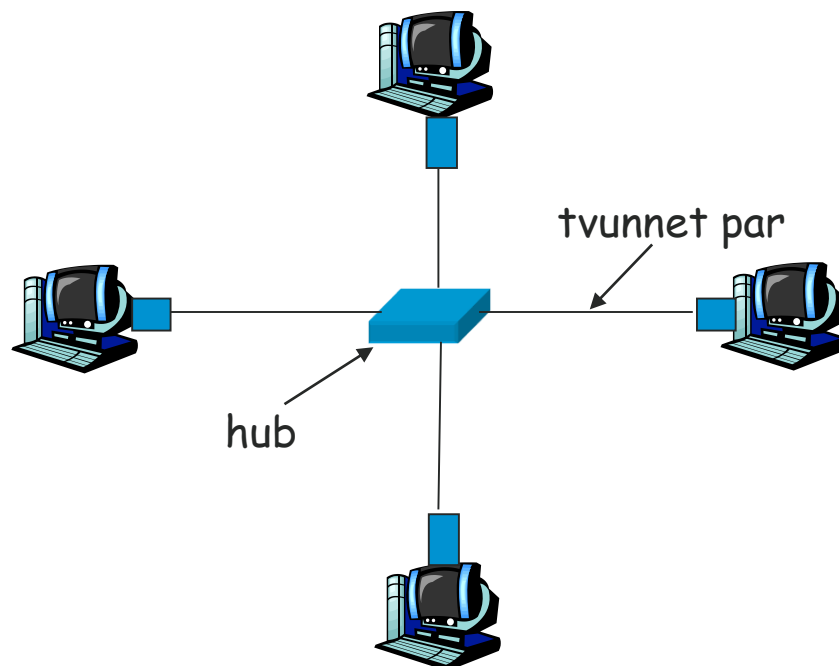
**Bit-tid:** 10 ns for 100 Mb/s Ethernet;  
for  $K = 1023$  vil følgelig ventetiden være omkring 5 ms

## **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..

# 10BaseT og 100BaseT

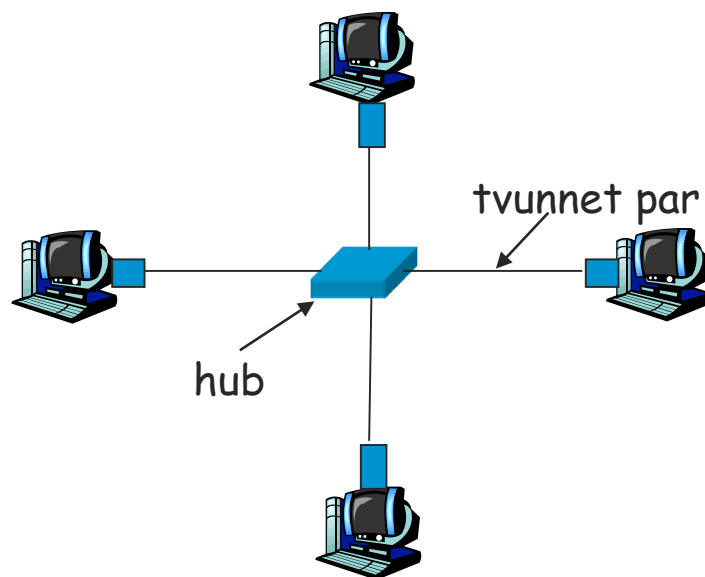
- 10/100 Mbps rater; sistnevnte kalles “fast ethernet”
- 1 GbE (1000BASE-T)
  - Bruker alle trådparrene, og komplisert koding
- **T** står for “twisted pair” (tvunnet par)
- Noder forbundet med en “hub”(eller switch): stjernetopologi; maks avstand fra node til hub er ca 100 m



# Huber

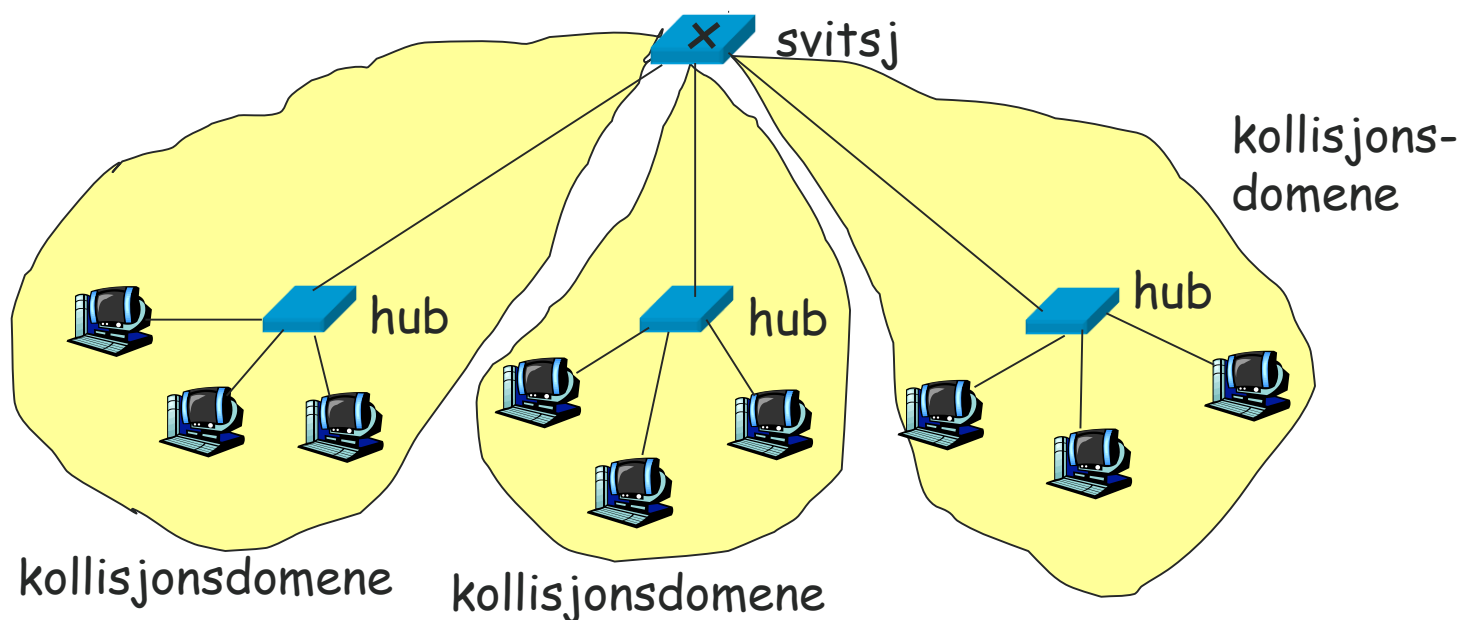
Huber er multiport repeatere (fysisk lag):

- 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



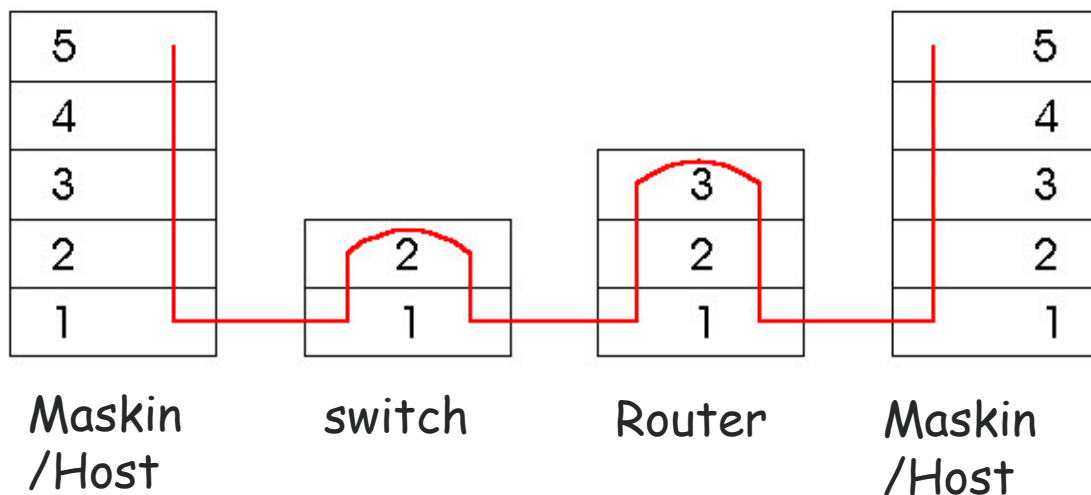
# Svitsj: trafikkisolasjon

- 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

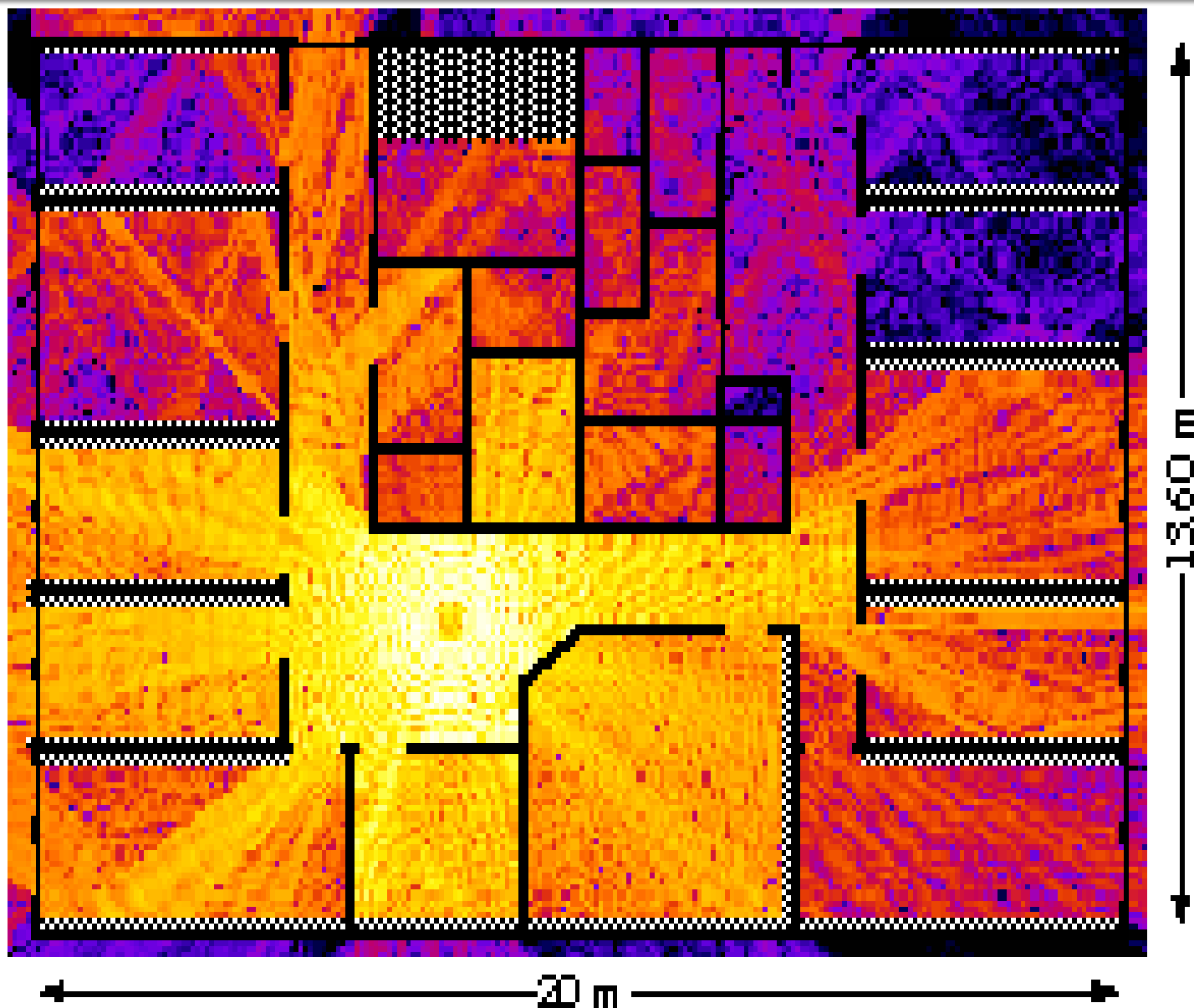




# IEEE 802.11 Wireless LAN (Wi-Fi)

- 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



That's all folks...

# Dagens øving

---

- Løs tidligere eksamensoppgave (2008, 2009 og 2017)
- Bruk veilederne og foreleser til å få hjelp til alt du er usikker på!
- Lykke til på eksamen
- Lykke til neste semester (jeg ser dere igjen i TK2100 – og datasikkerhet er GØY :-)