

Nettlaget

Mål:

- Forstå prinsippene bak nettlagets oppgaver:
 - Ruting
 - Skalerbarhet
 - Hvordan en ruter virker
- Hvordan dette er løst i Internett

Temaer:

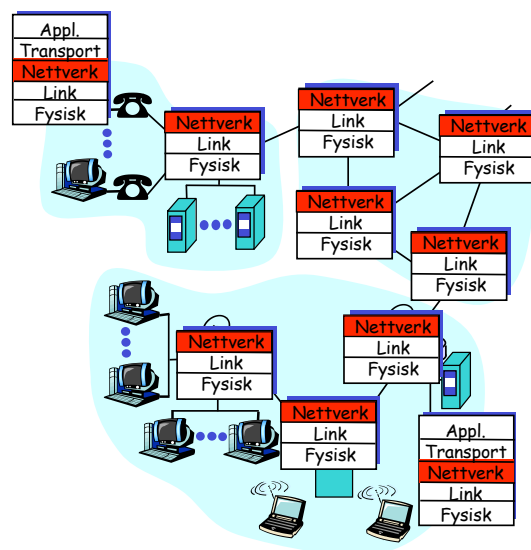
- Nettlagets oppgaver
- Rutingprinsipp: veivalg
- Hierarkisk ruting
- IP
- Internettets ruting protokoller
 - intra-domain
 - inter-domain

Nettlagets funksjoner

- Transportere pakker fra sender til mottaker.
- Nettlagsprotokoller i alle endemaskiner og rutere.

Tre viktige funksjoner:

- *Finne en sti*: veien fra kilde til destinasjon
- *switching*: flytte pakker fra input porten til output porten
- *call setup*: noen nettverk krever en initialiseringsfase før data kan sendes.



Nettverkets tjenestemodell

Hvilken modell skal man velge for den kanalen som transporterer pakker fra sender til mottaker?

Tjeneste abstraksjoner

- Garantert båndbredde?
- Ingen variasjon i forsinkelser (jitter)?
- Ingen pakketap?
- Levering i rekkefølge?
- Metningsfeedback til sender?

Den viktigste Abstraksjonen man må forholde seg til:

virtual circuit
eller
datagram?

Virtual circuits

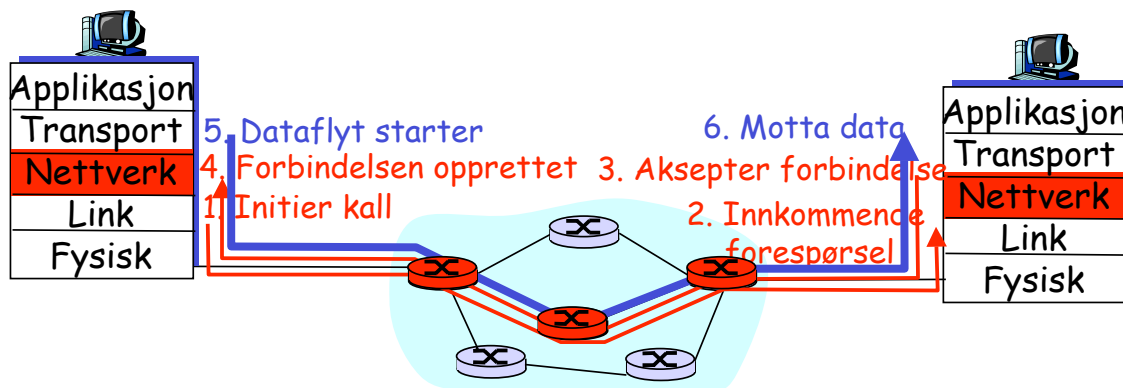
"Ligner på en telefonlinje hva gjelder..."

- ytelse
- Hva nettverket må gjøre langs stien fra kilde til destinasjon

- Forbindelsen må settes opp før data kan sendes
- Hver pakke inneholder ID av forbindelsen (ikke av destinasjonen)
- Hver ruter må vite om hvilke forbindelser som går igjennom den.
- link, ruterressurser (båndbredde, buffere) kan allokeres til en forbindelse
 - Oppførsel som om det ikke var noen annen trafikk på nettet.

Virtual circuits: signaleringsprotokoler

- ❑ Brukes til å sette opp, vedlikeholde og ta ned en VC.
- ❑ Benyttes i ATM, frame-relay, X.25
- ❑ Ikke benyttet i dagens Internett

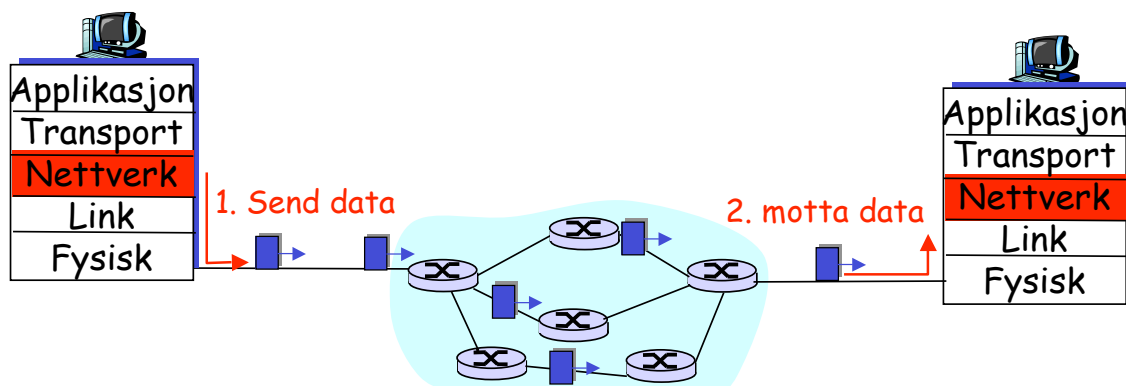


[[simula](#) . research laboratory]

Nettlaget -5

Datagram nettverk: Internett modellen

- ❑ Ingen fase for oppretting av forbindelse
- ❑ rutere: ingen informasjon om ende til endeforbindelse
 - Ikke noe "forbindelsesbegrep" på nettverkslaget
- ❑ Pakker blir videresendt på bakgrunn av destinasjons ID.
 - Pakker mellom samme kilde og destinasjon kan ta forskjellige stier



[[simula](#) . research laboratory]

Nettlaget -6

Datagram OG VC nettverk: hvorfor?

Internett

- Utsveksling av data mellom datamaskiner
 - "elastisk" tjeneste, ingen strikte timing krav.
- "smarte" endesystemer (datamaskiner)
 - Kan tilpasse seg, utføre kontroll, feilhåndtering
 - Enkelt nettverk, kompleksiteten er samlet i endesystemene
- Mange basalteknologier
 - Forskjellige karakteristika
 - Uniform tjeneste er vanskelig

ATM

- Utviklet fra telefoni
- Menneskelig konversasjon:
 - Strikt timing, pålitelighetskrav
 - Trenger tjenestegarantier
- "dumme" endesystemer
 - telefoner
 - Kompleksiteten i nettverkene

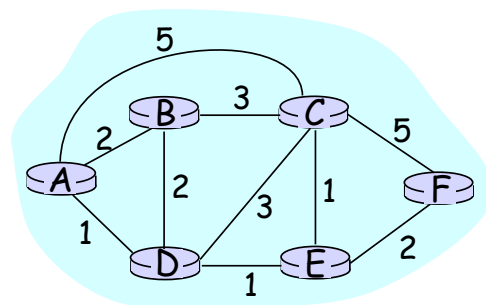
Ruting

Ruting protokoll

Mål: finne en "god" sti (sekvens av rutere) gjennom nettverket fra kilde til destinasjon

Graf-abstraksjon for rutingalgoritmer:

- Nodene i grafen er rutere
- Kantene er fysiske linker
 - Link kostnad: forsinkelse, \$ kost, eller metningsnivå



- "god" sti:
 - Typisk minimum kost (for ett eller annet kost-begrep)
 - Andre definisjoner er mulige

Klassifikasjon av rutingalgoritmer

Global eller desentralisert informasjon?

Global:

- ❑ Alle rutere har all informasjon om topologi og link-
- ❑ "link state" algoritmer

Desentralisert:

- ❑ Ruter kjenner til sine nærmeste naboer, og deres link-kostnad.
- ❑ Iterativ prosess som utveksler informasjon med naboen
- ❑ "distansevektor" algoritmer

[[simula](#) . research laboratory]

Statisk eller dynamisk?

Statisk:

- ❑ Stier (ruter) endrer seg langsomt over tid.

Dynamisk:

- ❑ Stier endrer seg hurtig
 - Periodisk oppdatering
 - Som reaksjon på endring i link-kostnad

Nettlaget -9

En "Link-State" algoritme

Dijkstra's algoritme

- ❑ Topologi og linkkostnader er kjent for alle noder
 - Dette oppnås ved at alle noder gjør en "link state broadcast"
 - Alle noder har samme informasjon
- ❑ Hver node regner ut den stien som gir lavest kostnad fra seg selv til hver tenkelig destinasjon:
 - gir **ruting tabell** for den noden
- ❑ iterativ: etter k iterasjoner kjenner noden den billigste veien til k destinasjoner.

Notasjon:

- ❑ $c(i,j)$: link kostnad fra node i til node j. Settes initielt til uendelig dersom i og j ikke er naboer.
- ❑ $D(v)$: nåværende beste kostnad til destinasjon v.
- ❑ A : noden selv
- ❑ N : settet av noder som vi nå har funnet billigste veien til.
- ❑ $P(v)$: siste noden før v i den beste stien funnet til nå.

[[simula](#) . research laboratory]

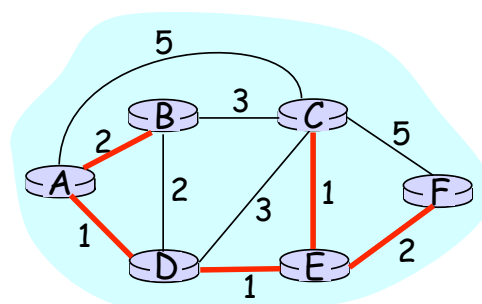
Nettlaget -10

Dijkstra's Algorithm

- 1 **Initialization:**
- 2 $N = \{A\}$
- 3 for all nodes v
- 4 if v adjacent to A
- 5 then $D(v) = c(A,v)$
- 6 else $D(v) = \text{infty}$
- 7
- 8 **Loop**
- 9 find w not in N such that $D(w)$ is a minimum
- 10 add w to N
- 11 update $D(v)$ for all v adjacent to w and not in N :
- 12 $D(v) = \min(D(v), D(w) + c(w,v))$
- 13 /* new cost to v is either old cost to v or known
- 14 shortest path cost to w plus cost from w to v */
- 15 **until all nodes in N**

Dijkstra's algoritme: eksempel

Step	start N	$D(B),p(B)$	$D(C),p(C)$	$D(D),p(D)$	$D(E),p(E)$	$D(F),p(F)$
→ 0	A	2,A	5,A	1,A	uendelig	uendelig
→ 1	AD	2,A	4,D		2,D	uendelig
→ 2	ADE	2,A	3,E			4,E
→ 3	ADEB		3,E			4,E
→ 4	ADEBC					4,E
5	ADEBCF					



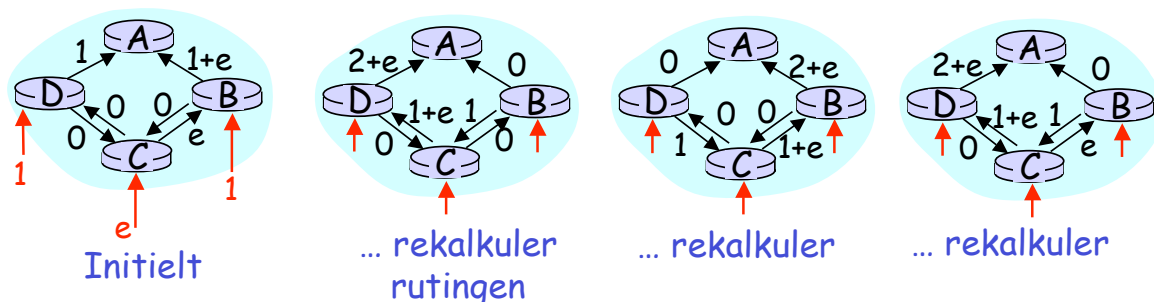
Dijkstra's algoritme, diskusjon

Kompleksitet: n noder

- Hver iterasjon: må sjekke alle noder w som ikke er i N
- $n*(n+1)/2$ sammenligninger: $O(n^2)$
- Mulig å implementere mer effektivt: $O(n \log n)$

Oscillasjoner er mulige:

- e.g., linkkostnad = hvor mye trafikk som går på linken



[**simula** . research laboratory]

Nettlaget -13

Distansevektor algoritmen

iterativ:

- Fortsetter inntil ingen noder utveksler informasjon.
- *Selv-terminerende:* ingen sentral avgjørelse om at alg. er ferdig

asynkron:

- Noder trenger *ikke* utveksle informasjon så lenge intet endrer seg

distribuert:

- Hver node kommuniserer bare med sine naboer

Distansetabellen

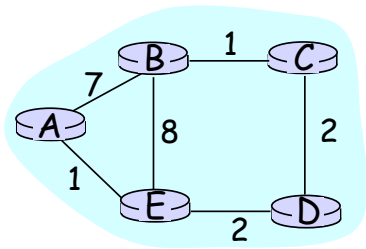
- Hver node har sin egen
- En rad for hver mulig destinasjon
- En kolonne for hver nabo (mulige utlinker)
- eksempel: i node X, for dest. Y via nabo Z:

$$D^X(Y,Z) = \text{distanse fra X til Y, via Z som neste hopp} \\ = c(X,Z) + \min_w \{D^Z(Y,w)\}$$

[**simula** . research laboratory]

Nettlaget -14

Distansetabell: eksempel



$$D^E(C,D) = c(E,D) + \min_w \{D^D(C,w)\} \\ = 2+2 = 4$$

$$D^E(A,D) = c(E,D) + \min_w \{D^D(A,w)\} \\ = 2+3 = 5$$

$$D^E(A,B) = c(E,B) + \min_w \{D^B(A,w)\} \\ = 8+6 = 14$$

Passerer E to ganger! (E-B-C-D-E-A)!

Kostnad til destinasjon via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destinasjon

Distansetabell gir rutingtabell

Kostnad til destinasjon via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destinasjon

	Utgående link, kostnad
A	A,1
B	D,5
C	D,4
D	D,2

destinasjon

Distansetabell → Rutingtabell

Distansevektor ruting: overblikk

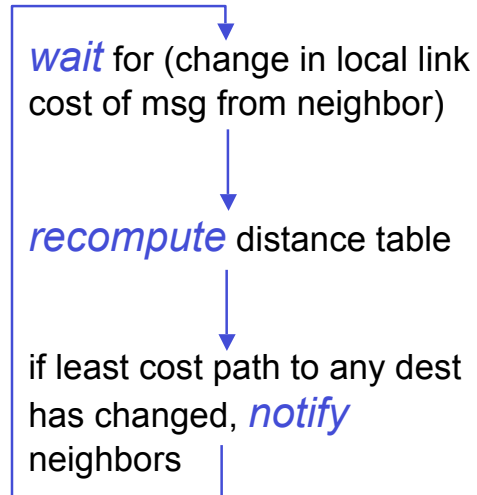
Iterativ, asynkron: Hver lokale iterasjon trigges av:

- endringer i lokal link-kost
- melding fra nabo om at dens minste konstans stitabell har endret seg

Distribuert:

- hver node sender oppdateringer til naboen *bare* når minste kostnads sti til en destinasjon har endret seg.
 - Naboe sender meldinger til *sine* naboer igjen dersom det er nødvendig.

Hver node:



Distansevektoralgoritmen:

I alle noder, X:

- 1 Initialization:
- 2 for all adjacent nodes v:
- 3 $D^X(*,v) = \text{infty}$ /* the * operator means "for all rows" */
- 4 $D^X(v,v) = c(X,v)$
- 5 for all destinations, y
- 6 send $\min_w D^X(y,w)$ to each neighbor /* w over all X's neighbors */

Distansevektoralgoritmen forts.

```

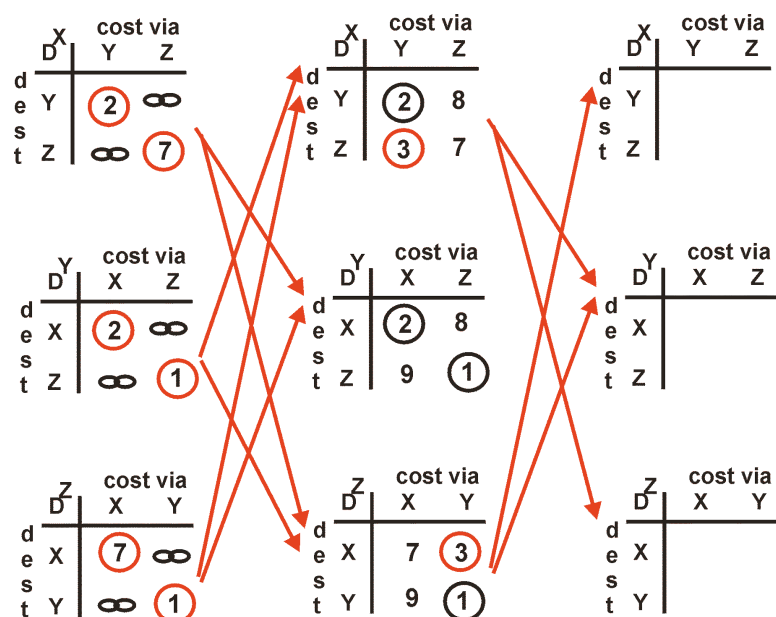
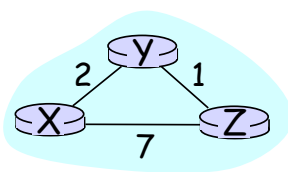
8 loop
9 wait (until I see a link cost change to neighbor V
10      or until I receive update from neighbor V)
11
12 if (c(X,V) changes by d)
13   /* change cost to all dest's via neighbor v by d */
14   /* note: d could be positive or negative */
15   for all destinations y:  $D^X(y,V) = D^X(y,V) + d$ 
16
17 else if (update received from V wrt destination Y)
18   /* shortest path from V to some Y has changed */
19   /* V has sent a new value for its  $\min_w D^V(Y,w)$  */
20   /* call this received new value is "newval" */
21   for the single destination y:  $D^X(Y,V) = c(X,V) + \text{newval}$ 
22
23 if we have a new  $\min_w D^X(Y,w)$  for any destination Y
24   send new value of  $\min_w D^X(Y,w)$  to all neighbors
25
26 forever

```

[[simula](#) . research laboratory]

Nettlaget -19

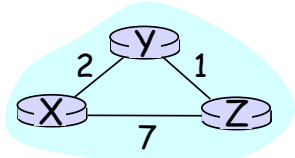
Distansevektoralgoritmen: eksempel



[[simula](#) . research laboratory]

Nettlaget -20

Distansevektoralgoritmen: eksempel



D	cost via	
	Y	Z
de	2	∞
st	∞	7

D	cost via	
	X	Z
de	2	∞
st	∞	1

D	cost via	
	X	Y
de	7	∞
st	∞	1

D	cost via	
	Y	Z
de	2	8
st	3	7

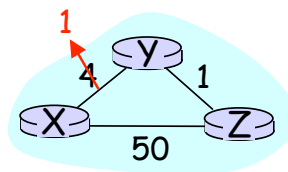
$$D^X(Y,Z) = c(X,Z) + \min_w \{D^Z(Y,w)\} = 7+1 = 8$$

$$D^X(Z,Y) = c(X,Y) + \min_w \{D^Y(Z,w)\} = 2+1 = 3$$

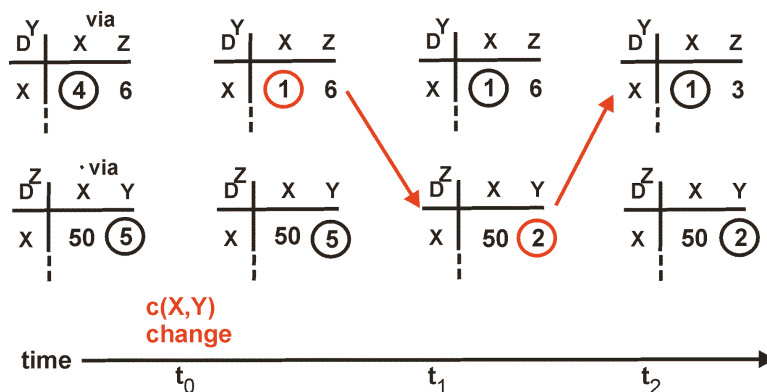
Distansevektor: endring i link-kostnad

Endring i link-kostnad:

- node oppdager endring i kostnad på lokal link.
- Oppdaterer distansetabellen (linje 15)
- dersom korteste sti endrer seg, si fra til naboene (lines 23,24)



"gode nyheter spres fort"

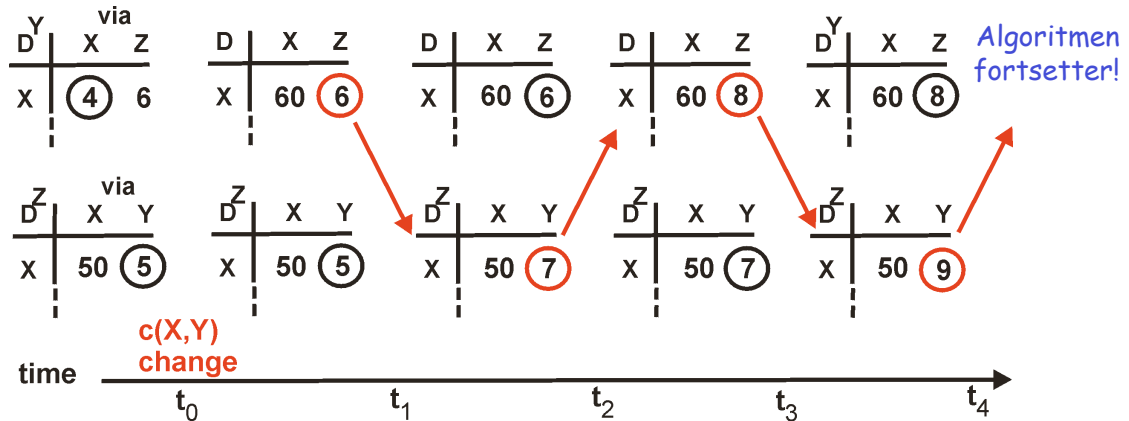
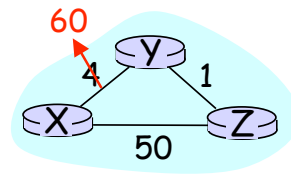


Algoritmen terminerer

Distansevektor: endring i link-kostnad

Endring i link kostnad:

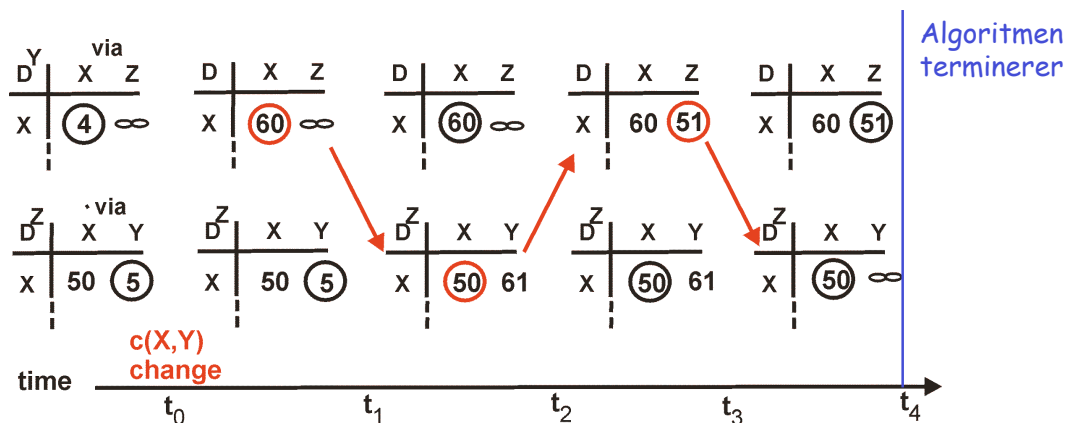
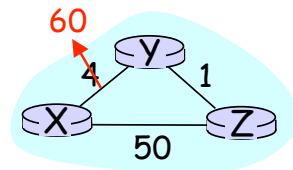
- gode nyheter spres fort
- dårlige nyheter spres sakte - "telle til uendelig" problem!



Distansevektor: "poisoned reverse"

Om Z ruter gjennom Y for å nå X :

- Z forteller Y at sin (Z's) distanse til X er uendelig (så Y ikke ruter til X via Z)
- løser dette "telle til uendelig" problemet?



Sammenligning av LS og DV algoritmene

Meldingskompleksitet

- **LS:** med n noder, E linker, $O(nE)$ msgs sent hver
- **DV:** meldingsutveksling bare mellom naboer
 - konvergenstiden varierer

Konvergenshastighet

- **LS:** $O(n^2)$ algoritme krever $O(nE)$ meldinger
 - kan oscillere
- **DV:** konvergenstiden varierer
 - kan gi ruting-løkker
 - "telle til uendelig" problem

Robusthet: hva skjer dersom en ruter feiler?

LS:

- node kan melde om feil linkkostnad
- hver node regner bare ut sin egen tabell

DV:

- DV node kan melde om feil sti-kostnad
- hver node sin tabell brukes av andre
 - feil propagerer gjennom nettverket

Hierarkisk ruting

Til nå har vi forholdt oss til et idealisert bilde.

- Alle rutere identiske
 - "flat" nettstruktur
- ... *ikke* sant i praksis

skala: med 50 millioner destinasjoner:

- kan ikke lagre alle destinasjoner i rutingtabeller!
- Utveksling av rutingtabeller ville okkupere linkkapasiteten!

administrativ autonomi

- internet = nettverk av nettverk
- hver enkelt netverksadministrator ønsker å kontrollere ruting i sitt eget nettverk.

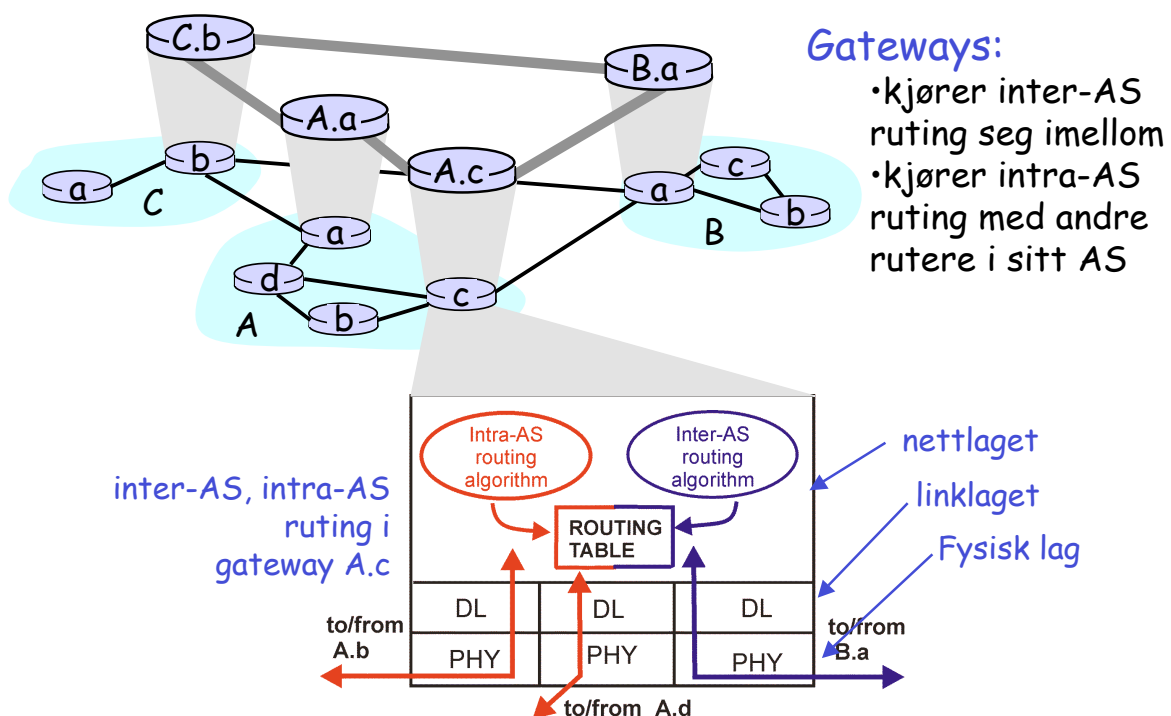
Hierarkisk ruting

- Samle rutere i regioner, "autonomous systems" (AS)
- rutere i samme AS kjører samme rutingprotokoll
 - "inter-AS" ruting protokoll
 - rutere i forskjellige AS kan kjøre forskjellige intra-AS ruting protokoller

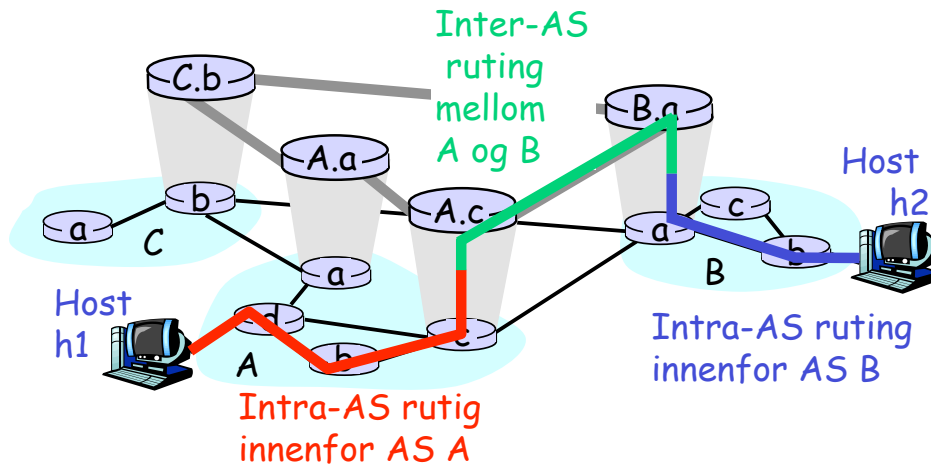
gateway rutere

- Spesielle rutere i et AS
- kjører intra-AS ruting protokoll med alle andre rutere i AS
- også ansvarlig for ruting til destinasjoner utenfor AS.
 - kjører *inter-AS ruting* protokoll med andre gateway rutere

Intra-AS og Inter-AS ruting



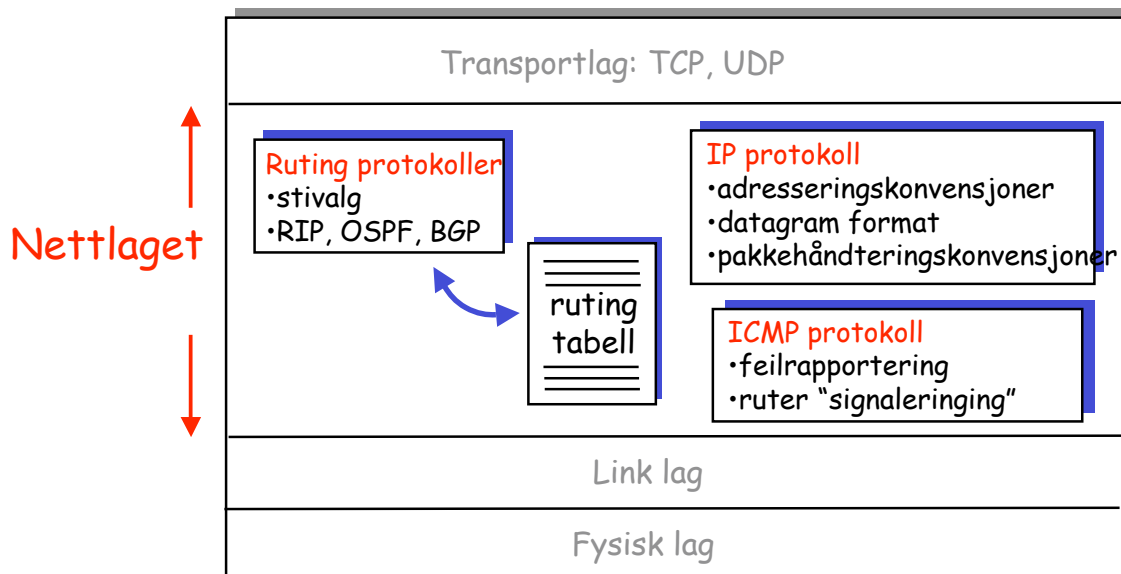
Intra-AS og Inter-AS ruting



- Vi skal studere inter-AS og intra-AS Internett ruting protokoller om et øyeblikk

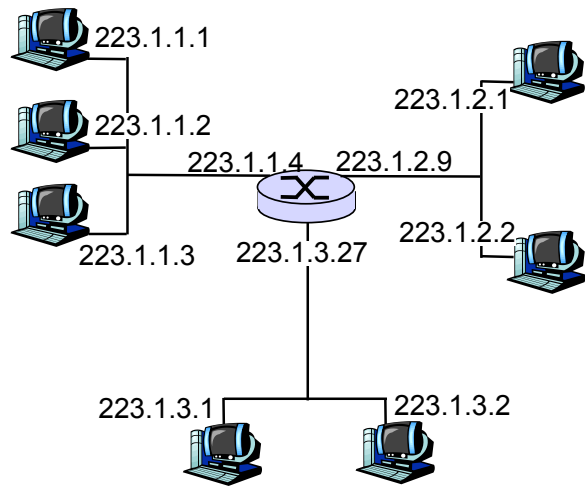
Nettlaget i Internett

Nettverkslagets funksjoner:



IP Adressering

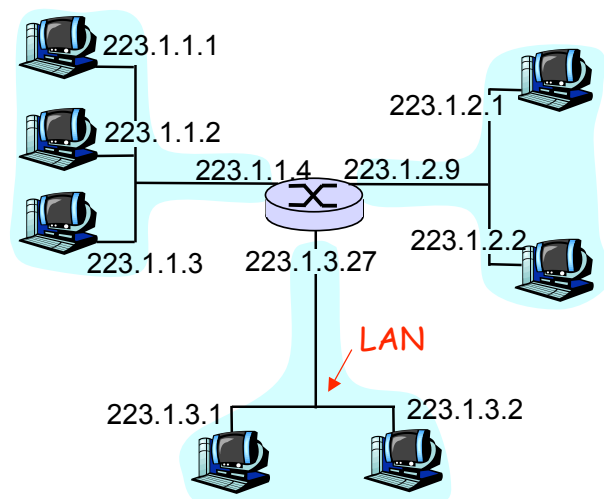
- **IP adresse:** 32-bit identifikator for maskin, ruter *interface*
- **interface:** koblingen mellom maskin/ruter og fysisk link
 - en ruter har typisk mange interfaces
 - en maskin kan ha flere interfacer (men vanligvis bare ett)
 - IP adresser assosiert med interface, ikke maskin.



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

IP Adressering

- **IP adresse:**
 - nettverksdelen (høyeste bits)
 - "vertsmaskin" delen (laveste bits)
- **Hva er et nettverk ?** (fra IP-adresse synspunkt)
 - interfacer som har felles nettverksdel i adressen
 - kan nå hverandre fysisk uten å gå gjennom en IP-ruter



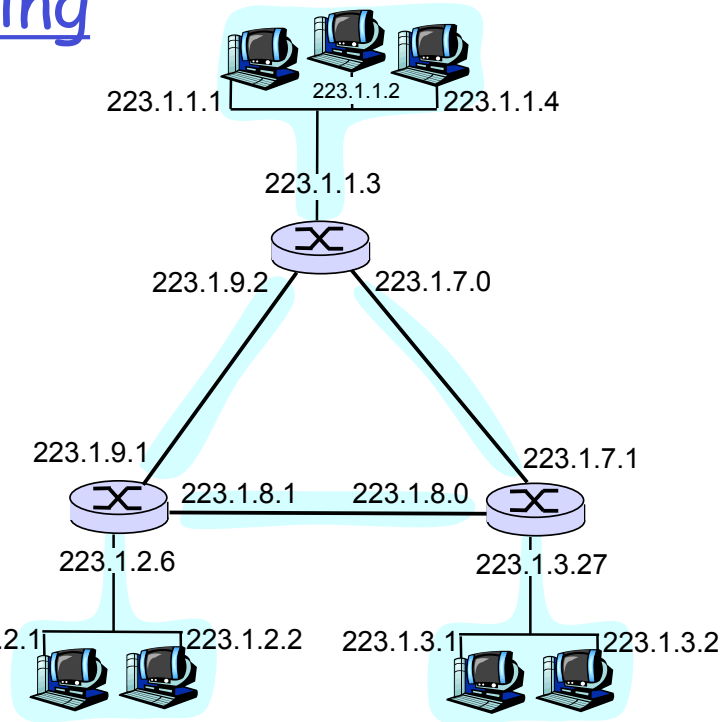
Nettverk som består av 3 IP nettverk (for IP adresser som starter med 223, er de første 24 bits nettverksadresse)

IP Adressering

Hvordan finne nettverket?

- Frikoble interface begrepet fra ruterbegrepet
- skape "øyer" av isolerte nettverk

Sammenkoblet system bestående av seks nettverk



IP Adresser

class

A	0	nettverk	vertsmaskin	10.0.0 til 127.255.255.255
B	10	nettverk	vertsmaskin	128.0.0.0 til 191.255.255.255
C	110	nettverk	vertsmaskin	192.0.0.0 til 239.255.255.255
D	1110	multicast adresse		240.0.0.0 til 247.255.255.255

← 32 bits →