Yhteystila-algoritmi

Dijkstran algoritmi

- verkon topologia ja linkkien hinta kaikkien solmujen tiedossa
  - Saavutetaan “linkin tila yleislähetyksillä”
  - Kaikilla solmuilla on sama informaatio
- Laskee pienimmän hinnan polut yhdestä solmusta (“lähde”) kaikkiin muihin solmuihin
  - Saadaan ko. solmun forwardointitaulukko
- iteratiivinen: *k*:n iteraation jälkeen tiedetään *k*:n kohteen pienimmän hinnan polut

Notaatio:

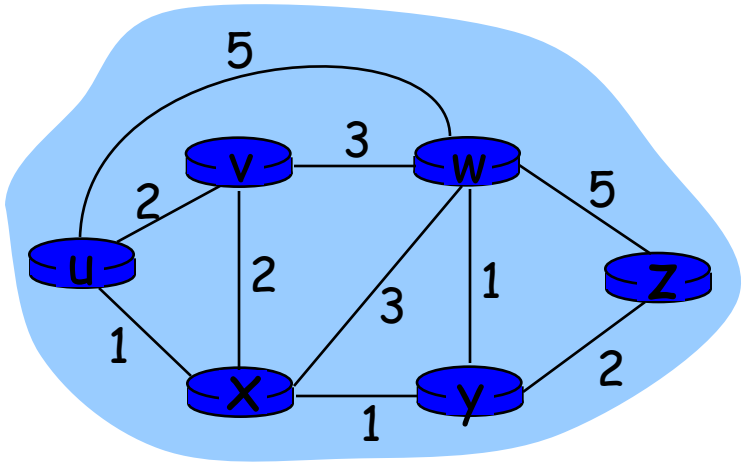
- c(x,y): linkin hinta solmusta x solmuun y; = ∞ jos eivät ole naapureita
- D(v): polun nykyinen hinta lähteestä kohteeseen v
- p(v): edeltävä solmu polulla lähteestä kohteeseen v
- N': joukko solmuja joiden pienimmän hinnan polku tiedetään varmasti

Dijkstran Algoritmi

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
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'
```

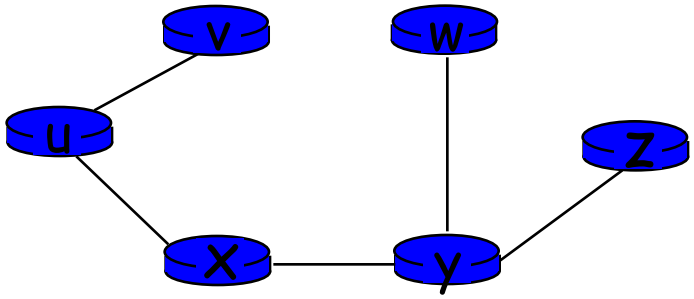
Dijkstran algoritmi: esimerkki

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Dijkstran algoritmi: esimerkki(2)

Resulting shortest-path tree from u:



Resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Etäisyysvektorialgoritmi (1)

Bellman-Ford yhtälö

määritellään

$d_x(y)$:= pienimmän polun hinta solmusta x solmuun y

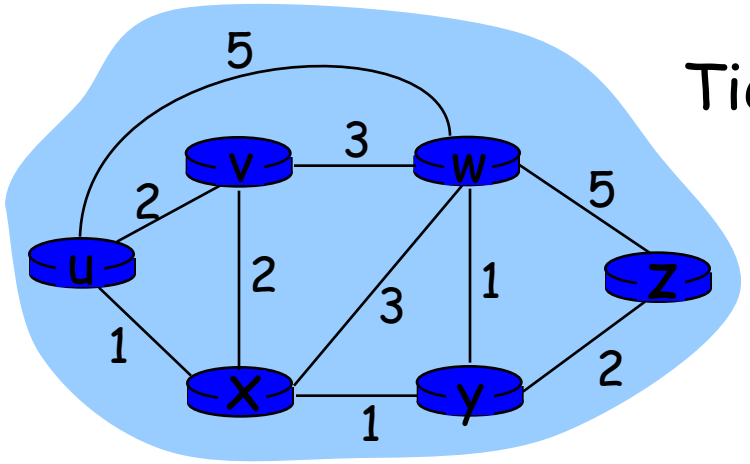
c(x,v) oli linkin hinta solmusta x solmuun v

Silloin

$$d_x(y) = \min \{c(x,v) + d_v(y) \}$$

missä min otetaan kaikkien x:n naapureiden yli

Bellman-Ford esimerkki (2)



Tiedetään, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

Bellman-Ford yhtälö sanoo:

$$d_u(z) = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

Etäisyysvektorialgoritmi (3)

- $D_x(y)$ = estimaatti pienimmälle hinnalle solmusta x solmuun y
- Solmu x tietää hinnan jokaiseen naapuriin v: $c(x,v)$
- Solmu x ylläpitää etäisyysvektoria $\mathbf{D}_x = [D_x(y): y \in N]$
- Solmu x säilyttää myös naapureidensa etäisyysvektoreita
  - Jokaiselle naapurille v, x säilyttää tiedon $\mathbf{D}_v = [D_v(y): y \in N]$

Etäisyysvektorialgoritmi (4)

Perusidea:

- Jokainen solmu lähettää säännöllisesti oman arvionsa etäisyysvektorista (DV) naapureilleen
- Kun solmu x vastaanottaa uuden etäisyysvektori-estimaatin naapuriltaan, päivittää se oman etäisyys-vektorinsa käyttäen Bellman-Ford yhtälöä:

$$D_x(y) \leftarrow min_v\{c(x,v) + D_v(y)\} \quad for\ each\ node\ y \in N$$

- Normaalissa olosuhteissa, estimaatti $D_x(y)$ *suppenee todelliseen pienimpään hintaan* $d_x(y)$

Etäisyysvektorialgoritmi (5)

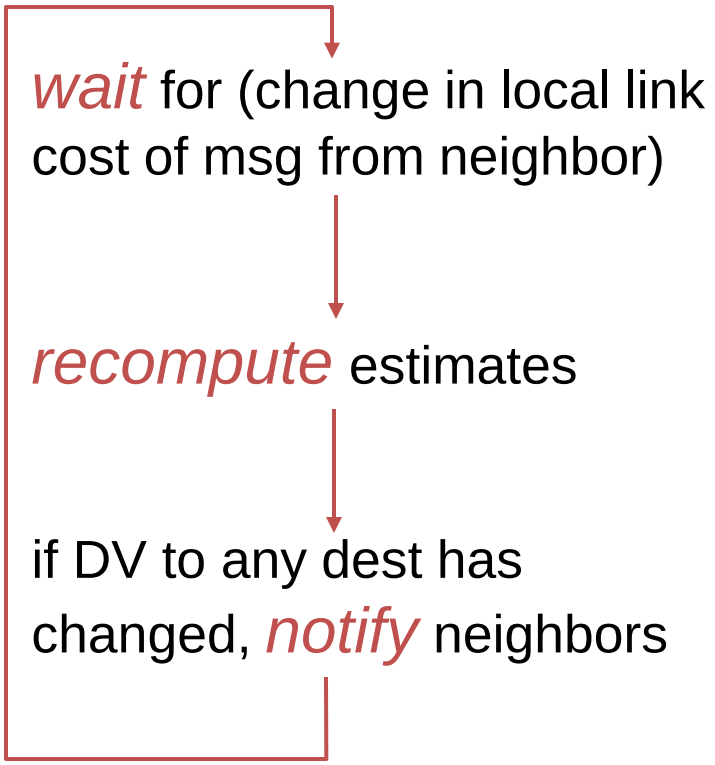
Iteratiivinen, asynkroninen:
jokainen paikallisen iteraation syy:

- Paikallisen linkin hinnan muutos
- Naapurin etäisyysvektorin päivitysviesti

Hajautettu:

- Jokainen solmu ilmoittaa *vain* naapureilleen kun sen oma etäisyysvektori muuttuu
  - Naapurit ilmoittavat sitten omille naapureilleen jos tarpeellista

Jokainen solmu:



$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$
$= \min\{2+1\ ,\ 7+0\} = 3$

node x table

from	cost to x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from	cost to x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from	cost to x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from	cost to x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

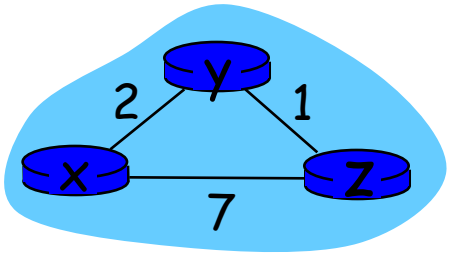
node z table

from	cost to x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from	cost to x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

3



time

3