

Etäisyysvektoriprotokollat

Etäisyysvektoriprotokollien periaatteet

Reittisilmukat ja niiden poistaminen

Bellman-Ford algoritmi

RIP-protokolla

Etäisyysvektoriprotokollien periaatteet

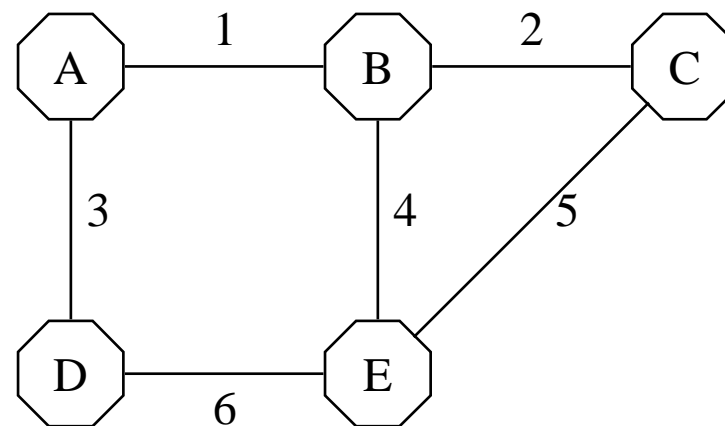
RIP – Routing Information Protocol on sisäisen reitityksen perusprotokolla

- RIP on etäisyysvektoriprotokolla
 - Perustuu Bellman-Ford algoritmiin
- Reititystaulu sisältää tietoja muista tunnetuista solmuista
 - etäisyys
 - linkki
- Solmut lähettävät säännöllisesti reititystauluun perustuvat etäisyysvektorit jokaiselle linkilleen
- Solmut päivittävät reititystauluaan vastaanotettujen etäisyysvektoreiden avulla

E to	Link	Distance
E	-	0
B	4	1
A	4	2
D	6	1
C	5	1

Tarkastellaan etäisyysvektoriprotokollien toimintaperiaatetta

Esimerkkiverkko, jossa solmut A, B, C, D, E ja linkit 1, 2, 3, 4, 5, 6.



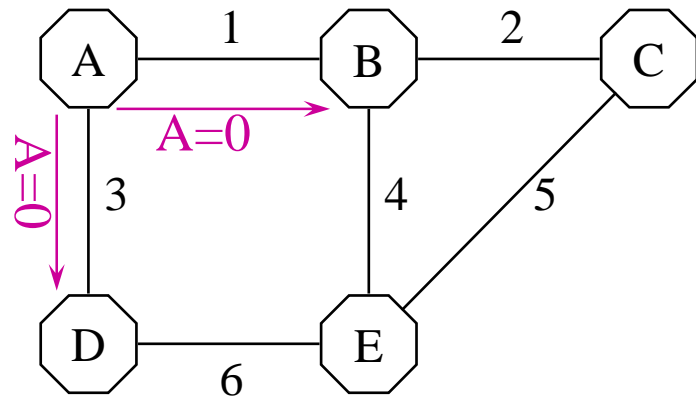
Alkutila: Solmut tuntevat omat osoitteensa ja liitântänsä, mutta ei muuta.

Solmun A reititystaulu:

Solmusta A solmuun ...	Linkki	Kustannus
A	-	0

Taulua vastaa etäisyysvektori $A=0$.

Reititystaulujen muodostus käynnistyy, kun kaikki solmut lähettävät toisilleen omat etäisyysvektorinsa kaikista liitännöistä



Tarkastellaan vastaanottoa solmussa B. Aluksi B:n taulu:

Solmusta B solmuun	Linkki	Kustannus
B	-	0

1. B vastaanottaa etäisyysvektorin $A=0$
2. B lisää heti $+1$ etäisyysvektoriin $\Rightarrow A=1$
3. B etsii tulosta omasta taulusta, ei löydy
4. B lisää saamansa tiedon reititystauluunsa, tulos on

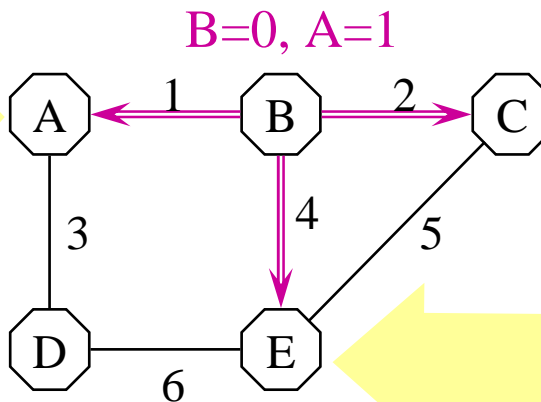
Solmusta B solmuun	Linkki	Kustannus
B	-	0
A	1	1

5. B muodostaa etäisyysvektorinsa $B=0, A=1$

B muodostaa oman vektorinsa ja lähettää sen kaikille naapureille

A →	Linkki	Kustannus
A	-	0
B	1	1

$A=2 > A=0$

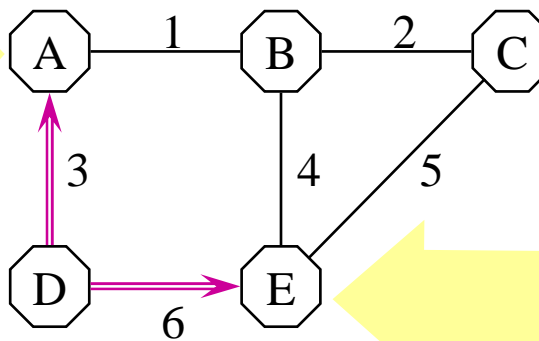


C →	Linkki	Kustannus
C	-	0
B	2	1
A	2	2

E →	Linkki	Kustannus
E	-	0
B	4	1
A	4	2

D lähettää vektorinsa kaikille naapureille

A →	Linkki	Kustannus
A	-	0
B	1	1
D	3	1

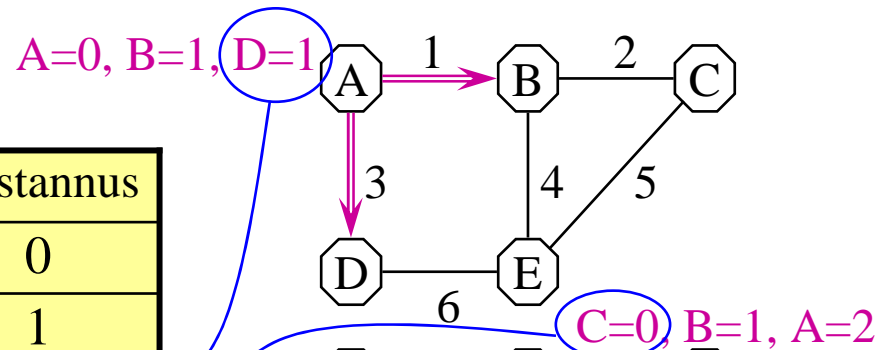


D=0, A=1

E →	Linkki	Kustannus
E	-	0
B	4	1
A	4	2
D	6	1

A=2 == A=2 ⇒ no change

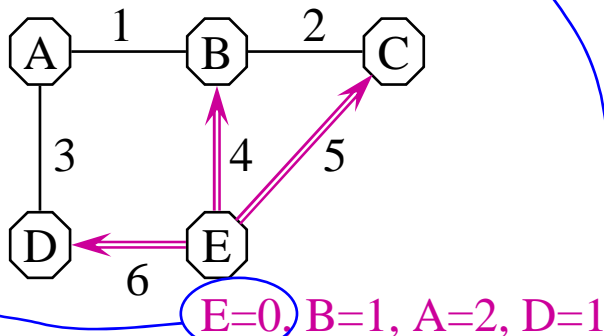
Solmut, joiden reititystaulut muuttuivat lähettävät uudet etäisyysvektorit naapureille



B →	Linkki	Kustannus
B	-	0
A	1	1
D	1	2
C	2	1
E	4	1



D →	Linkki	Kustannus
D	-	0
A	3	1
B	3	2
E	6	1

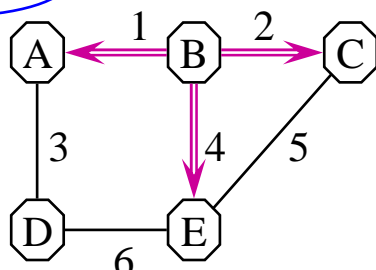


E →	Linkki	Kustannus
E	-	0
B	4	1
A	4	2
D	6	1
C	5	1

Muutokset lähetetään taas ...

$B=0, A=1, D=2, C=1, E=1$

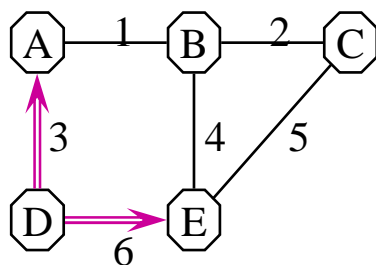
A →	Linkki	Kustannus
A	-	0
B	1	1
D	3	1
C	1	2
E	1	2



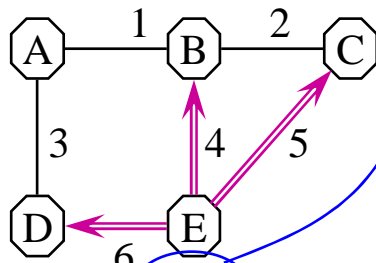
C →	Linkki	Kustannus
C	-	0
B	2	1
A	2	2
E	5	1
D	5	2

$D=0, A=1, B=2, E=1$

D →	Linkki	Kustannus
D	-	0
A	3	1
B	3	2
E	6	1
C	6	2

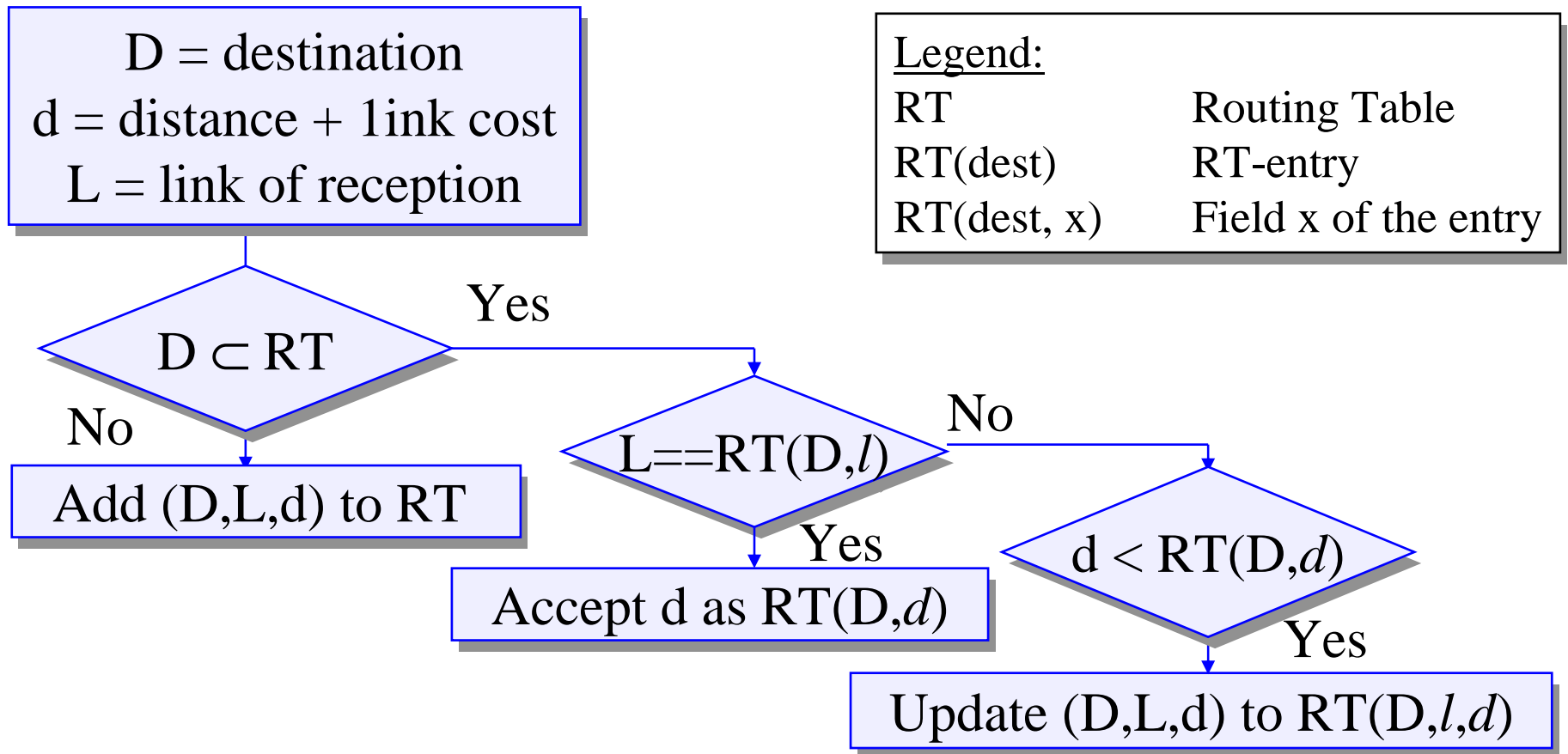


$E=0, B=1, A=2, D=1, C=1$



A, D, ja C muodostavat vielä vektorit, lähettävät ne mutta ne eivät enää aiheuta muutoksia.

Processing of received distance vectors



Note: this is simplified, shows only the principle!

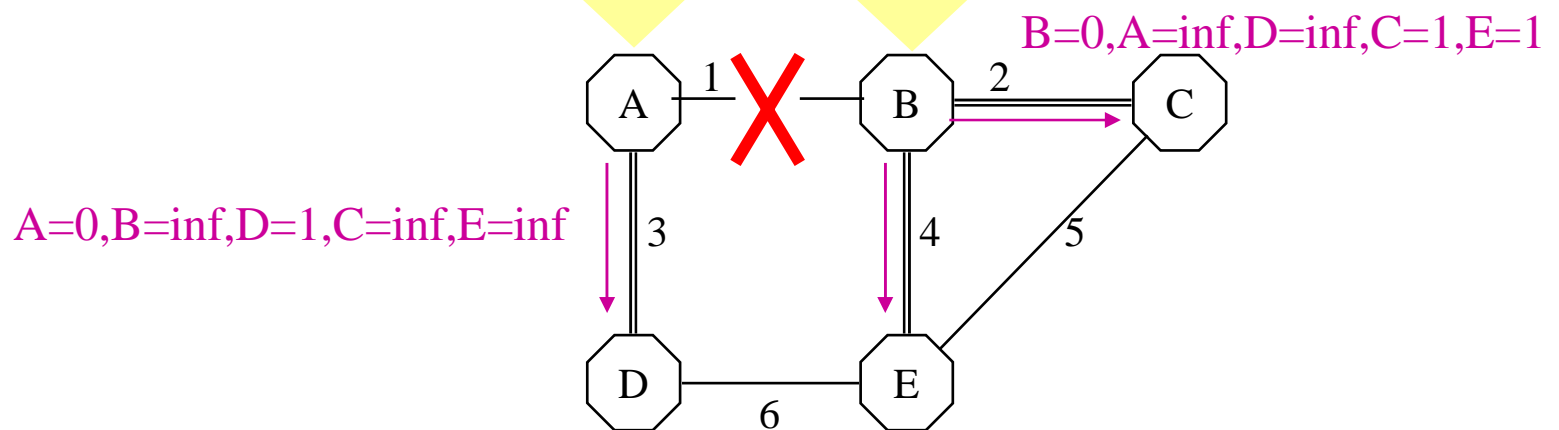
Linkin katkeaminen

Linkin katkeaminen käynnistää päivityskierroksen

A laittaa linkin 1 kautta saavutettavien solmujen etäisyydeksi ääretön.

A →	Linkki	Kustannus
A	-	0
B	1	Inf.
D	3	1
C	1	Inf.
E	1	Inf.

B →	Linkki	Kustannus
B	-	0
A	1	Inf.
D	1	Inf.
C	2	1
E	4	1



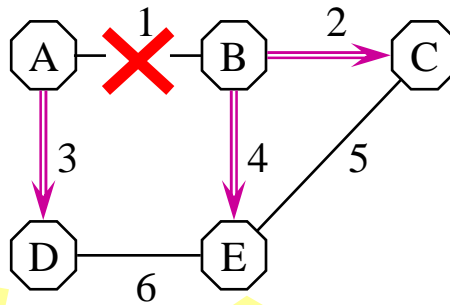
D, E ja C päivittävät reititystaulunsa

$B=0, A=\text{inf}, D=\text{inf}, C=1, E=1$

$A=0, B=\text{inf}, D=1, C=\text{inf}, E=\text{inf}$

+1

$A=1, B=\text{inf}, D=2, C=\text{inf}, E=\text{inf}$



C →	Linkki	Kustannus
C	-	0
B	2	1
A	2	Inf.
E	5	1
D	5	2

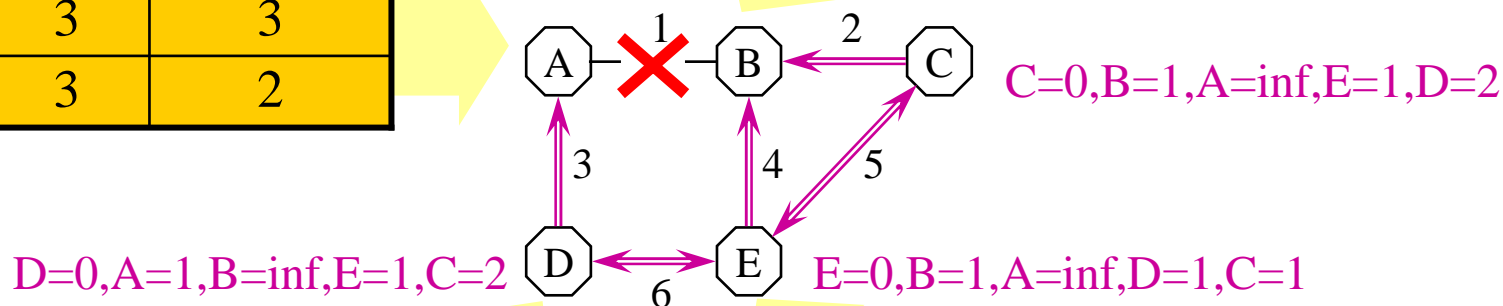
D →	Linkki	Kustannus
D	-	0
A	3	1
B	3	Inf.
E	6	1
C	6	2

E →	Linkki	Kustannus
E	-	0
B	4	1
A	4	Inf.
D	6	1
C	5	1

D, C, E generoivat etäisyysvektorinsa...

A →	Linkki	Kustannus
A	-	0
B	1	Inf.
D	3	1
C	3	3
E	3	2

B →	Linkki	Kustannus
B	-	0
A	1	Inf.
D	4	2
C	2	1
E	4	1



D →	Linkki	Kustannus
D	-	0
A	3	1
B	6	2
E	6	1
C	6	2

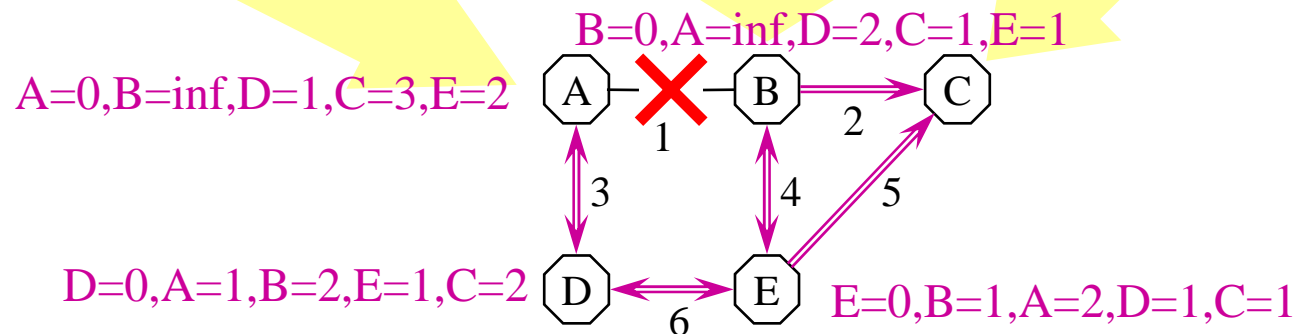
E →	Linkki	Kustannus
E	-	0
B	4	1
A	6	2
D	6	1
C	5	1

A, B, D, E generoivat etäisyysvektorinsa

A →	Linkki	Kustannus
A	-	0
B	3	3
D	3	1
C	3	3
E	3	2

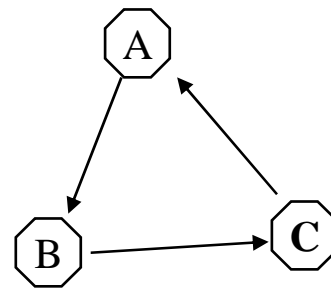
B →	Linkki	Kustannus
B	-	0
A	4	3
D	4	2
C	2	1
E	4	1

C →	Linkki	Kustannus
C	-	0
B	2	1
A	5	3
E	5	1
D	5	2

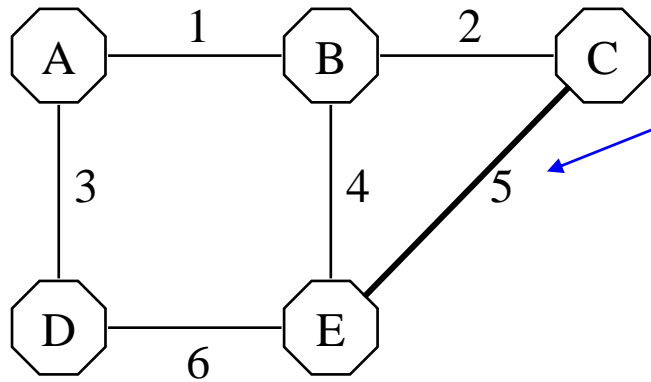


Tuloksena on, että kaikki voivat taas kommunikoida kaikkien kanssa.

Reittisilmukat



Etäisyysvektoriprotokolla voi synnyttää tilapäisen reittisilmukan

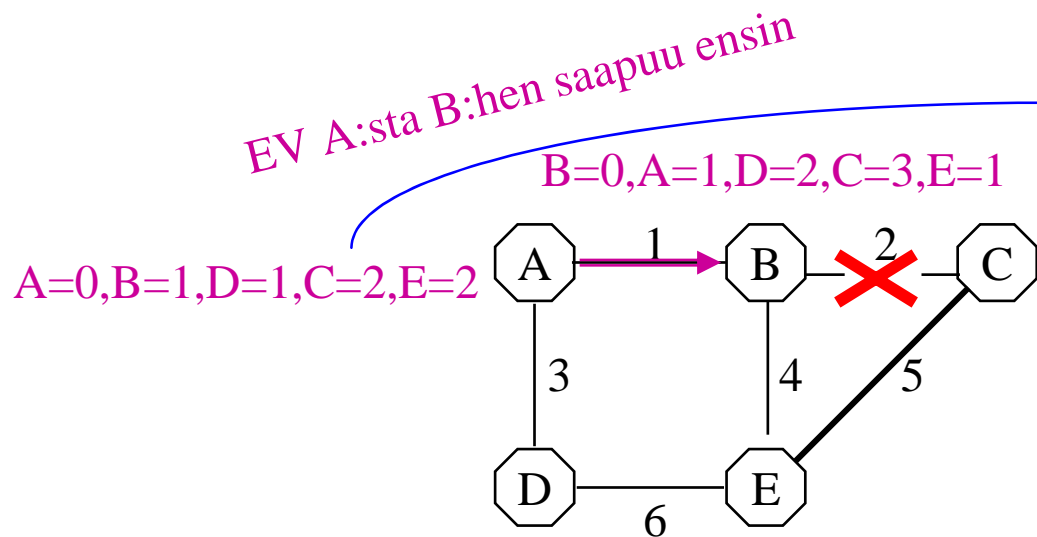


Oletetaan, että linkin 5 kustannus on 8
Stabiili lähtötila reiteillä C:hen olisi:

$x \rightarrow C$	Linkki x:stä	Kustannus
A→C	1	2
B→C	2	1
C→C	-	0
D→C	3	3
E→C	4	2

Keskitytään vain kunkin reitin ensimmäiseen linkkiin

Linkki 2 vioittuu



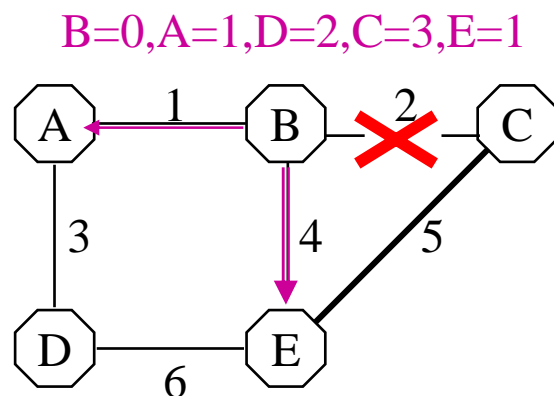
x → C	Linkki x:stä	Kustannus
A→C	1	2
B→C	2	Inf.
C→C	-	0
D→C	3	3
E→C	4	2

Välitila

x → C	Linkki x:stä	Kustannus
A→C	1	2
B→C	1	3
C→C	-	0
D→C	3	3
E→C	4	2

Kaikki viestit C:hen ohjataan B:lle, joka lähettää ne A:lle, joka lähettää ne B:lle... kunnes TTL=0. (Bouncing effect - pallottelu)

Linkki 2 vioittuu



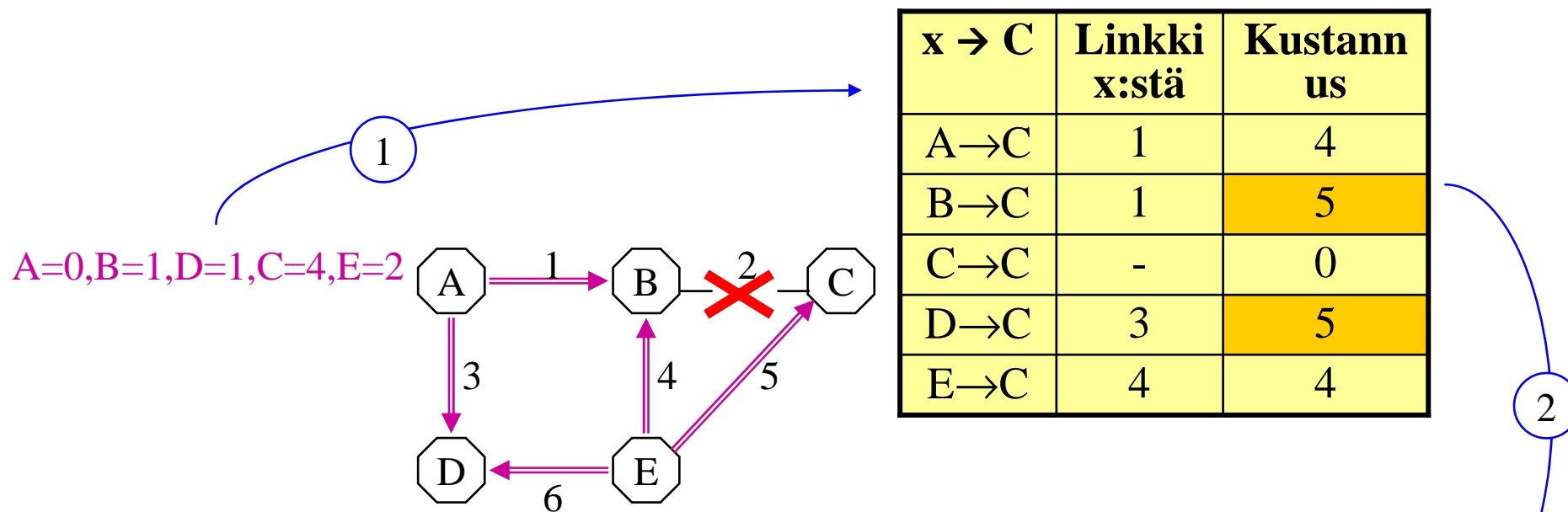
B lähettää etäisyysvektorinsa
 $B=0, A=1, D=2, C=3, E=1$

\Rightarrow A näkemä etäisyys C:hen kasvaa 4:ään

Kaikki viestit C:hen ohjataan B:lle, joka lähettää ne A:lle, joka lähettää ne B:lle... kunnes TTL=0. (Bouncing effect - pallottelu)

$x \rightarrow C$	Linkki x:stä	Kustannus
A \rightarrow C	1	4
B \rightarrow C	1	3
C \rightarrow C	-	0
D \rightarrow C	3	3
E \rightarrow C	4	4

A ja E lähettävät etäisyysvektorinsa



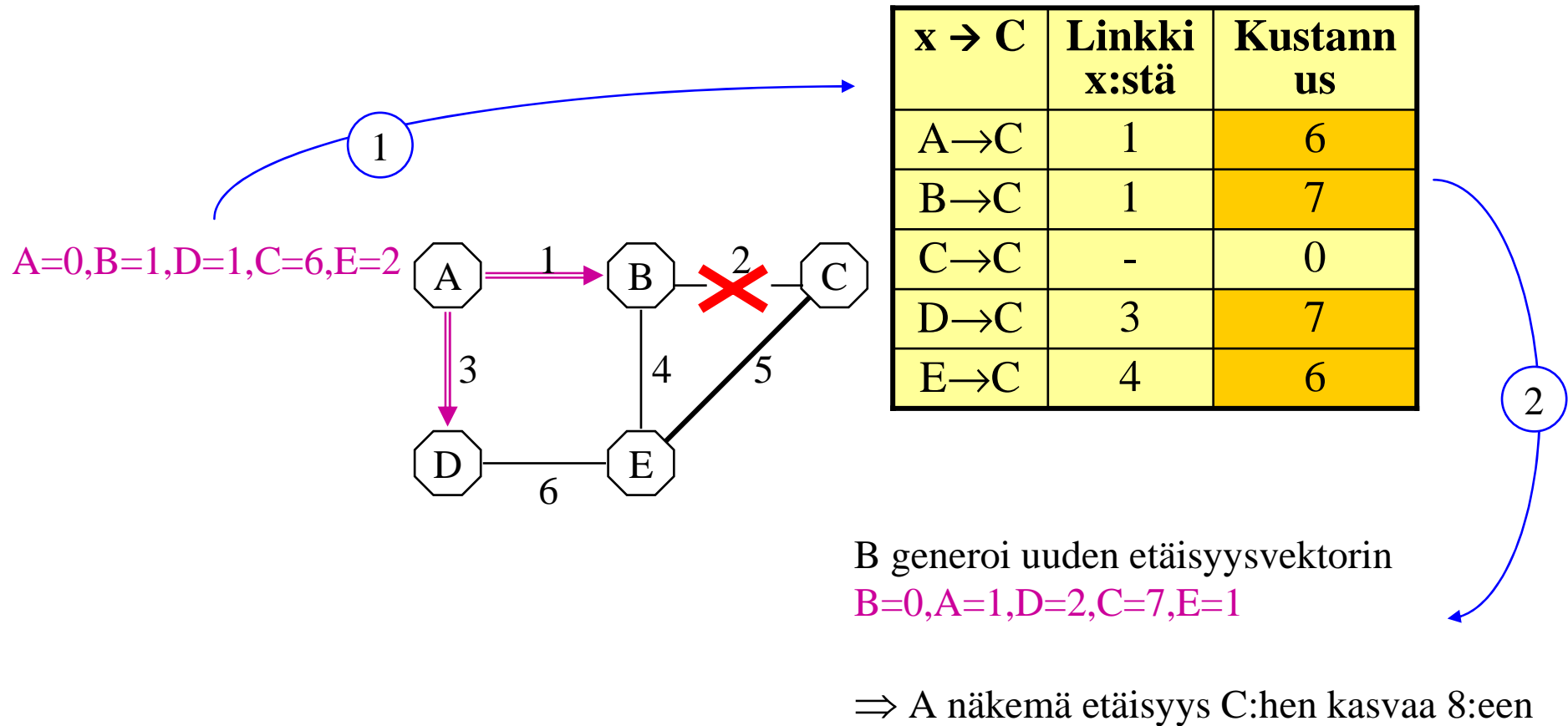
A=0, B=1, D=1, C=4, E=2

B generoi uuden etäisyysvektorin
 B=0, A=1, D=2, C=5, E=1

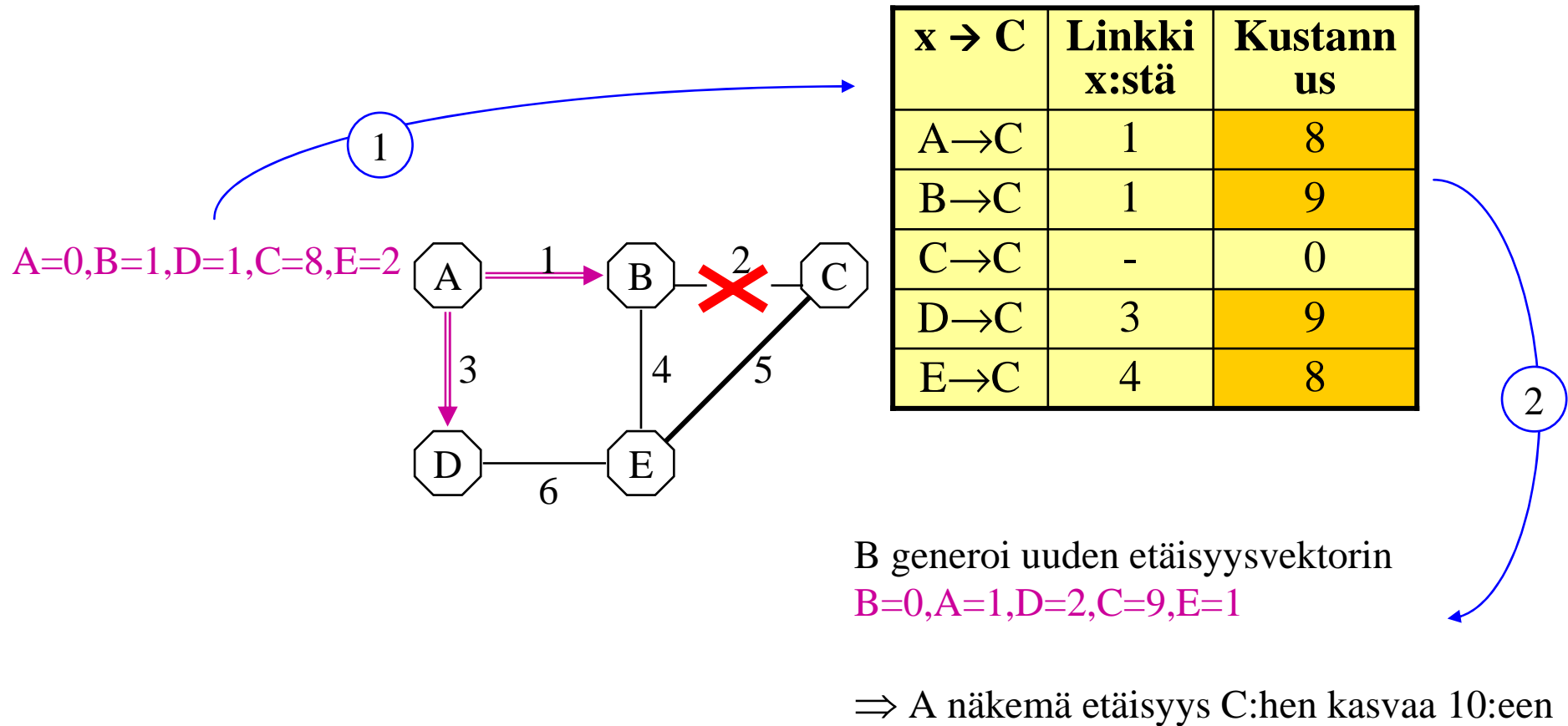
⇒ A näkemä etäisyys C:hen kasvaa 6:een

C:n lähettämät etäisyysvektorit eivät aiheuta muutoksia linkin kustannuksen takia

A lähettää uuden etäisyysvektorin

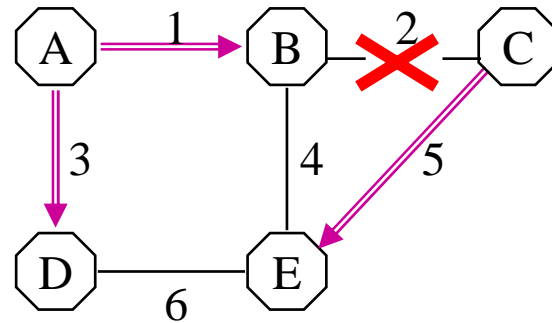


A lähettää uuden etäisyysvektorin



A lähettää uuden etäisyysvektorin

$A=0, B=1, D=1, C=10, E=2$



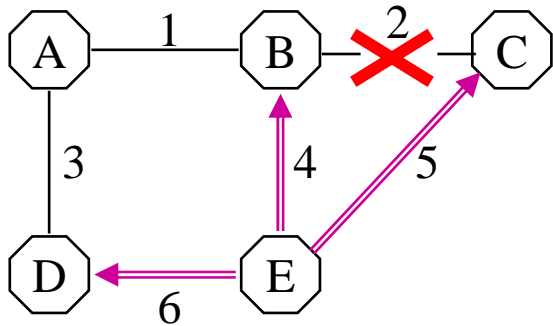
$x \rightarrow C$	Linkki x:stä	Kustannus
A→C	1	10
B→C	1	11
C→C	-	0
D→C	3	11
E→C	5	8

B generoi uuden etäisyysvektorin
 $B=0, A=1, D=2, C=11, E=1$

E generoi uuden etäisyysvektorin
 $E=0, B=1, A=2, D=1, C=8$

2

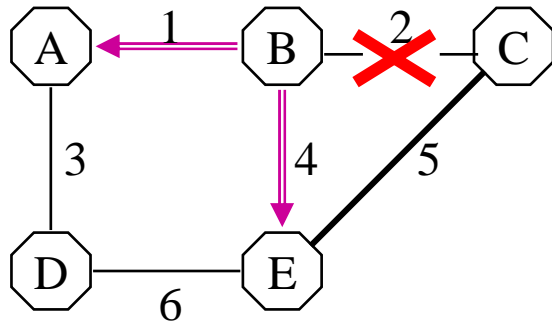
E lähettää uuden etäisyysvektorin



$E=0, B=1, A=2, D=1, C=8$

$x \rightarrow C$	Linkki x :stä	Kustannus
A→C	1	10
B→C	4	9
C→C	-	0
D→C	6	9
E→C	5	8

B lähettää EV:n, mutta taulut ovat jo OK



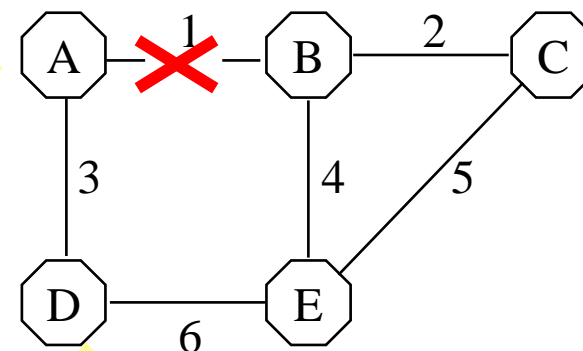
$x \rightarrow C$	Linkki x:stä	Kustannus
A→C	1	10
B→C	4	9
C→C	-	0
D→C	6	9
E→C	5	8

- Jokainen päivityskierros korjasi kustannuksia 2:lla.
- Prosessi etenee satunnaisessa järjestyksessä, koska siinä on aitoa rinnakkaisuutta.
- Prosessin aikana verkon tila on huono. EV-protokollasanomia voi hukkaa pallottelevien käyttäjäviestien aiheuttamassa ruuhkassa.

Irralliset saarekkeet aiheuttavat laskemisen äärettömään (1)

- Linkki 1 on vikaantunut, ja siitä on toivuttu.
- Kaikki linkki-kustannukset = 1

A →	Linkki	Kustannus
D	3	1
A	-	0
B	3	3
E	3	2
C	3	3

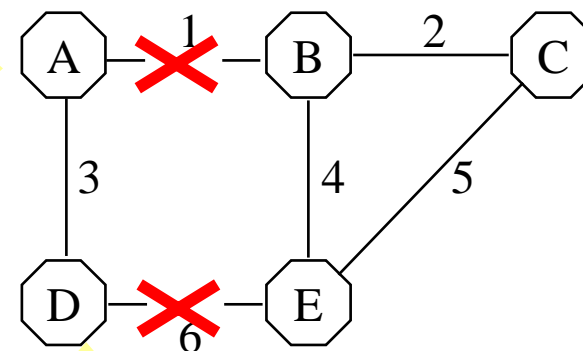


D →	Linkki	Kustannus
D	-	0
A	3	1
B	6	2
E	6	1
C	6	2

Irralliset saarekkeet aiheuttavat laskemisen äärettömään (2)

- Myös linkki 6 vikaantuu.
- D ei ole ehtinyt lähettää etäisyysvektoriaan.

A →	Linkki	Kustannus
D	3	1
A	-	0
B	3	3
E	3	2
C	3	3



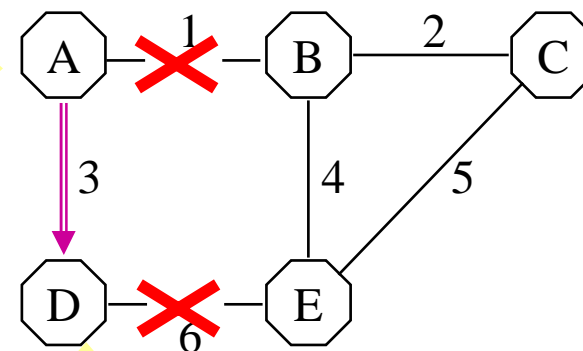
D →	Linkki	Kustannus
D	-	0
A	3	1
B	6	Inf.
E	6	Inf.
C	6	Inf.

Irralliset saarekkeet aiheuttavat laskemisen äärettömään (3)

- A ehtii lähettää etäisyysvektorinsa ensin:

$A=0, B=3, D=1, C=3, E=2$

A →	Linkki	Kustannus
D	3	1
A	-	0
B	3	3
E	3	2
C	3	3



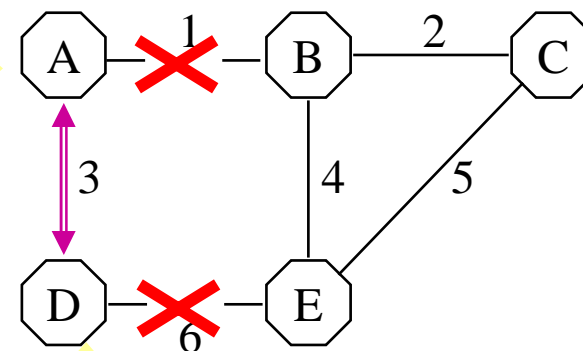
- D lisää A:n lähettämät tiedot reititystauluunsa.

D →	Linkki	Kustannus
D	-	0
A	3	1
B	3	4
E	3	3
C	3	4

Irralliset saarekkeet aiheuttavat laskemisen äärettömään (4)

- Tuloksena on silmukka. Kustannukset kasvavat 2:lla joka kierroksella.

A →	Linkki	Kustannus
D	3	1
A	-	0
B	3	5
E	3	4
C	3	5



- On sovittava, että inf on max-etäisyyttä suurempi kustannus

D →	Linkki	Kustannus
D	-	0
A	3	1
B	3	4
E	3	3
C	3	4

Silmukoita voidaan vähentää karsimalla etäisyysvektoreista tietoa

Karsintasääntö:

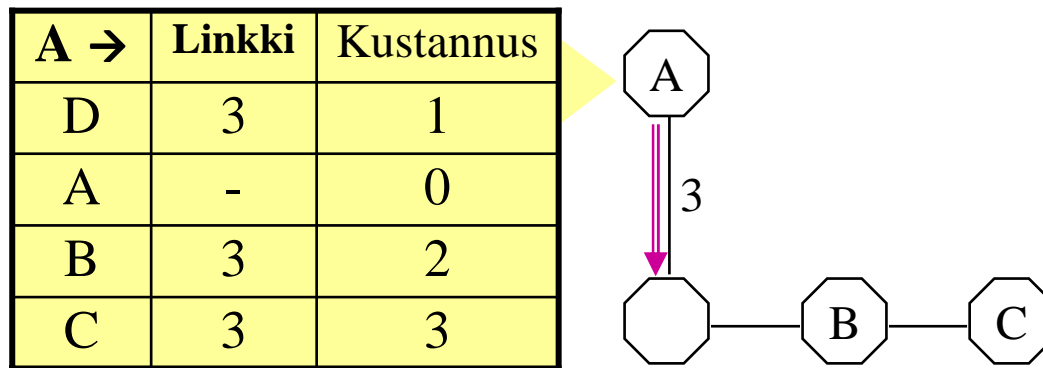
Jos solmu A lähettää solmulle X solmun B kautta, B:n ei kannata yrittää tavoitella X:ää A:n kautta.

⇒ A:n ei kannata mainostaa B:lle lyhyttä etäisyyttään X:ään

Toteutusvariaatiot:

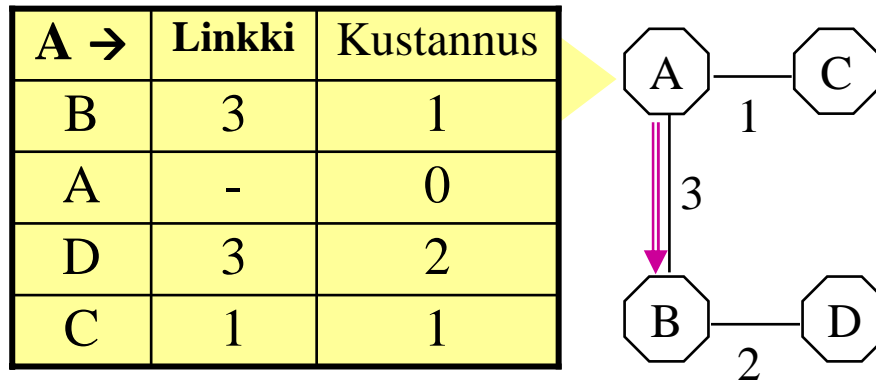
1. A ei mainosta etäisyyttään X:ään B:lle lainkaan (“*split horizon*”)
⇒ edellisen esimerkin silmukkaa ei synny
2. A mainostaa B:lle: $X = \text{inf}$. (“*split horizon with poisonous reverse*”, ”myrkytetyt vektorit”)
⇒ kahden solmun silmukat kuolevat heti.

Split horizon esimerkki



- Normally:
 - A sends: $A=0, B=2, C=3, D=1$
- Split horizon:
 - A sends: $A=0$
- Split horizon with poisonous reverse:
 - A sends: $A=0, B=\text{inf}, C=\text{inf}, D=\text{inf}$

Split horizon esimerkki

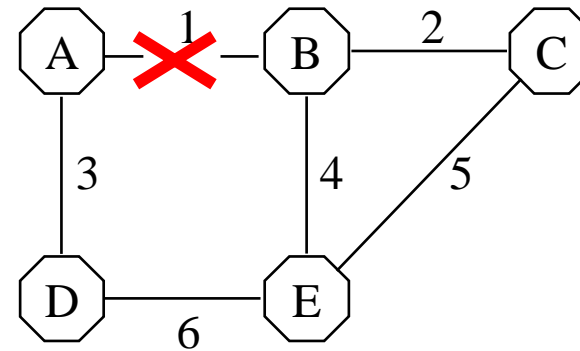


- Normaalisti:
 - A lähettää: $A=0, B=1, C=1, D=2$
- Split horizon:
 - A lähettää : $A=0, C=1$
- Split horizon with poisonous reverse:
 - A lähettää : $A=0, B=\text{inf}, C=1, D=\text{inf},$

Huom: Eri etäisyysvektori linkillä 1

Kolmen solmun silmukat ovat silti mahdollisia (1)

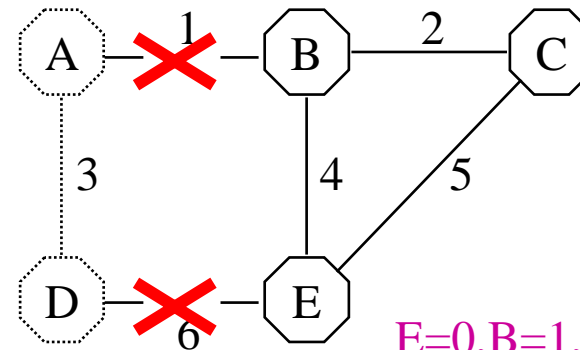
- Linkki 1 on vikaantunut, ja siitä on toivuttu.
- Kaikki linkki-kustannukset = 1



x → D	Linkki x:stä	Kustannus
B→D	4	2
C→D	5	2
E→D	6	1

Kolmen solmun silmukat ovat silti mahdollisia (2)

- Myös linkki 6 vikaantuu.
- E lähettää etäisyysvektorinsa B:hen ja C:hen



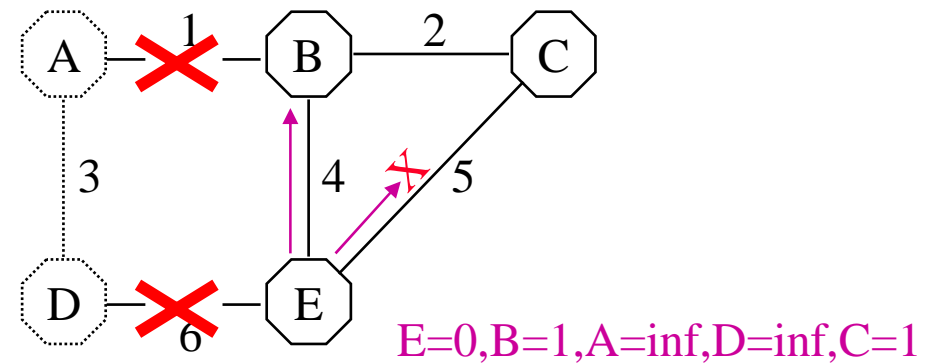
$E=0, B=1, A=\text{inf}, D=\text{inf}, C=1$

$E=0, B=1, A=\text{inf}, D=\text{inf}, C=1$

$x \rightarrow D$	Linkki x:stä	Kustannus
B→D	4	2
C→D	5	2
E→D	6	Inf.

Kolmen solmun silmukat ovat silti mahdollisia (3)

- Myös linkki 6 vikaantuu.
- E lähettää etäisyysvektorinsa B:hen ja C:hen



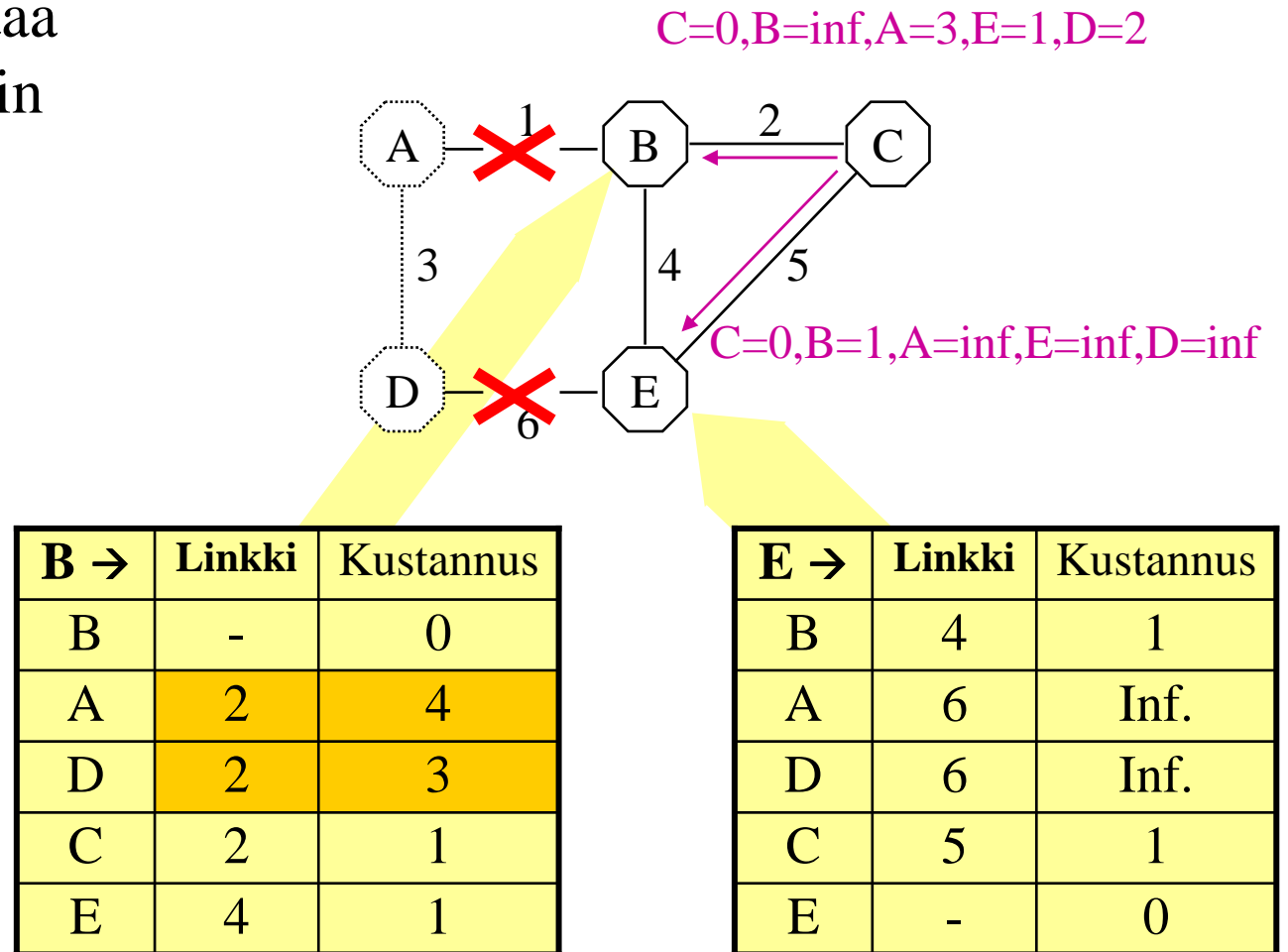
E=0, B=1, A=inf, D=inf, C=1

- ... Mutta C:hen lähetetty etäisyysvektori hukataan

x → D	Linkki x:stä	Kustannus
B → D	4	Inf.
C → D	5	2
E → D	6	Inf.

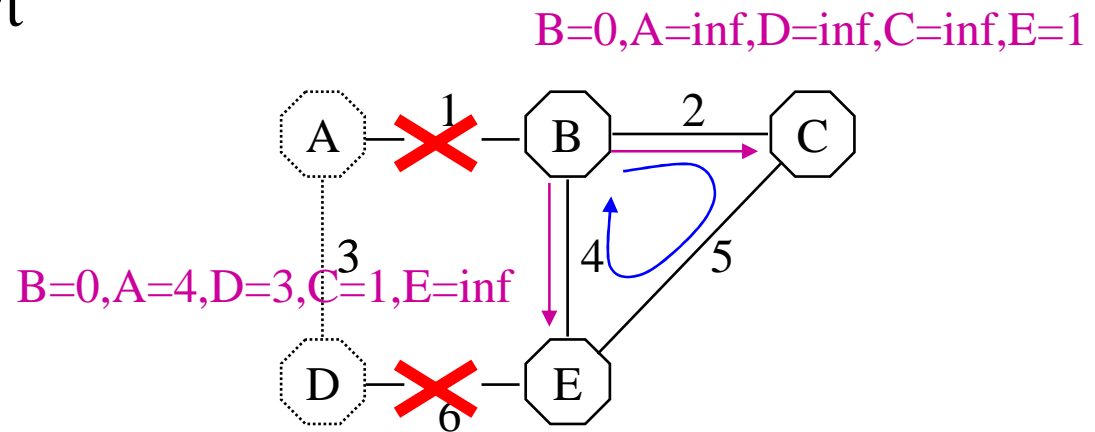
Kolmen solmun silmukat ovat silti mahdollisia (4)

- On C:n aika mainostaa myrkytetyin vektorein



Kolmen solmun silmukat ovat silti mahdollisia (5)

- B muodostaa myrkytetyt vektorit
- Kolmen solmun silmukka on valmis
- Reitit D:hen eivät enää muutu, lasketaan äärettömään, mikä viimein purkaa silmukan: linkillä 5 kerrotaan kustannus 4, C:n käsitys etäisyydestä D:hen alkaa kasvaa ...



$x \rightarrow D$	Linkki x:stä	Kustannus
B→D	2	3
C→D	5	2
E→D	4	4

Milloin EV-protokollan kannattaa lähettää

Lähetyshetki on kompromissi:

- + Tiedon välitön päivittäminen
- + Pakettien katoamisesta toipuminen
- + Naapureiden monitorointitarve
- Kaikkien muutosten lähettäminen yhtä aikaa
- Protokollan aiheuttama liikennekuorma

+ = Nopeasti
- = Hitaasti

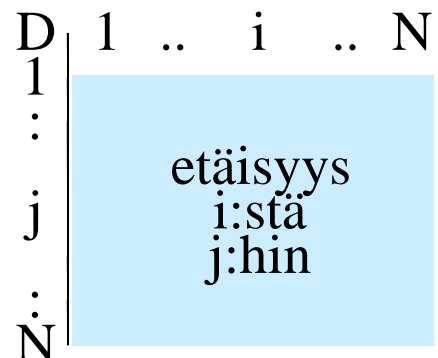
Silmukoita voidaan vähentää generoimalla EV:t heti taulun muututtua

- Reititystaulukon riveillä on **virikistys-** ja **vanhenemisajastin**
- RIP lähettää
 - aina virikistysajastimen lauetessa
 - heti kun muutos havaitaan
- Laukaistu päivitys nopeuttaa laskua äärettömään ja vähentää silmukoiden syntyä

Bellman-Ford algoritmi

Bellman-Ford algoritmi (1)

- EV-protokollat perustuvat Bellman-Ford algoritmiin
- Keskitetty versio:
 1. Olkoon N solmujen lukumäärä ja M linkkien lukumäärä.
 2. L on M -rivinen linkkitaulukko, $L[l].m$ - linkin mitta,
 $L[l].s$ - linkin alkupää,
 $L[l].d$ - linkin kohde
 3. D on $N \times N$ matriisi, jossa $D[i,j]$ on etäisyys i :stä j :hin
 4. H on $N \times N$ matriisi, jossa $H[i,j]$ on linkki, jolla i lähettää j :lle



Linkkitaulussa on molemmat suunnat erikseen!

Sarake vastaa solmun etäisyysvektoria!

Bellman-Ford algoritmi (2)

- Alustetut etäisyys- ja linkkimatriisit

	1	..	i	..	N
1	0	∞	∞	∞	∞
:	∞	0	∞	∞	∞
j	∞	∞	0	∞	∞
:	∞	∞	∞	0	∞
N	∞	∞	∞	∞	0

Etäisyysmatriisi D

	1	..	i	..	N
1	-1	-1	-1	-1	-1
:	-1	-1	-1	-1	-1
j	-1	-1	-1	-1	-1
:	-1	-1	-1	-1	-1
N	-1	-1	-1	-1	-1

Linkkimatriisi H

i:stä j:hin

Bellman-Ford algoritmi (3)

1. Alustetaan: Jos $i=j$ silloin $D[i,j] = 0$, muuten $D[i,j] = \text{inf}$.
Alustetaan $\forall H[i,j] = -1$. *(edellinen kalvo)*
2. $\forall l$ ja \forall kohteille k : aseta $i = L[l].s$, $j = L[l].d$ ja laske $d = L[l].m + D[j,k]$
3. Jos $d < D[i,k]$, aseta $D[i,k] = d$; $H[i,k] = l$.
4. Jos edes yksi $D[i,k]$ muuttui, toista kohta 2, muutoin algoritmi päättyy.

Bellman-Ford algoritmi (4)

- Aluksi D-matriisissa täyttyvät yhden linkin päässä olevien solmujen väliset etäisyydet, seuraavaksi kahden linkin päässä olevat, jne.
- Askelten lukumäärä $\leq N$
- Kompleksisuus: $O(M \cdot N^2)$
- Hajautetun algoritmin kompleksisuus: $O(M \cdot N)$

RIP Protokolla

RIP-protokollan peruspiirteitä (1)

- Yksinkertainen protokolla. Käytössä jo ennen standardointia.
- RIP versio 1
 - RFC 1058.
 - 1988
- RIP:iä käytetään autonomisen järjestelmän sisällä.
- RIP toimii sekä jaetun median verkoissa (esim. Ethernet) että yksipisteverkoissa (point-to-point).
- RIP toimii UDP:n ja IP:n päällä.

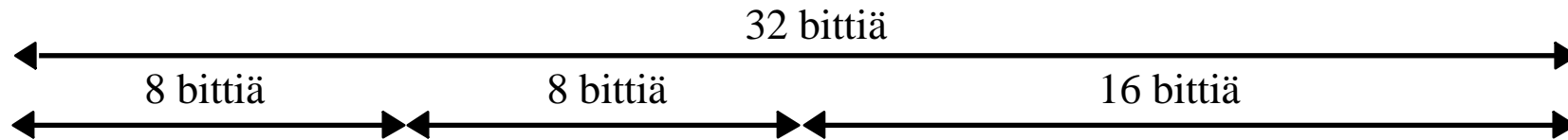
RIP-protokollan peruspiirteitä (2)

- Reititystaulun rivi esittää isäntäkonetta, verkkoa tai aliverkkoa (subnet)
 - $\langle \text{netid}, 0, 0 \rangle$ esittää verkkoa
 - $\langle \text{netid}, \text{subnetid}, 0 \rangle$ esittää aliverkkoa
 - $\langle \text{netid}, \text{subnetid}, \text{host} \rangle$ esittää isäntäkonetta
 - $\langle 0.0.0.0 \rangle$ esittää reittiä ulos autonomisesta järjestelmästä
- Maski pitää konfiguroida käsin
- Lähetykset naapuriverkkoon aggregoidaan

RIP-protokollan peruspiirteitä (3)

- Etäisyys (hop count) = polun peräkkäisten linkkien lukumäärä
 - Ei muita kustannusfunktioita
- Etäisyys 16 = ääretön
- RIP lähettää 30 s välein
 - Yli 180 s vanha reititysriivi \Rightarrow etäisyys asetetaan äärettömäksi
- Ajastimen käynnistämiä lähetyksiä täytyy satunnaistaa, jotta RIP liikenne tasoittuisi. 1-5 s.
- RIP lähettää myös 1-5 s päivityksen jälkeen.
- RIP käyttää myrkytettyjä vektoreita

RIP sanomaformaatti



Komento	Versio	Täytyy olla nolla
Osoiteperheen tunniste		Täytyy olla nolla
IP osoite		
Täytyy olla nolla		
Täytyy olla nolla		
Mitta		

IP=2

Komento: 1 = Pyyntö , 2 = Vaste
Versio = 1

Mitta: 0 ... 16

RIP reititystaulu

Reititystaulun rivi sisältää

- Kohteen osoite
- Etäisyys kohteeseen
- Seuraavan reitittimen osoite
- “Äsken päivitetty” -lippu
- Useita ajastimia (virkistys- , vanhenemis- ...)

Esimerkki

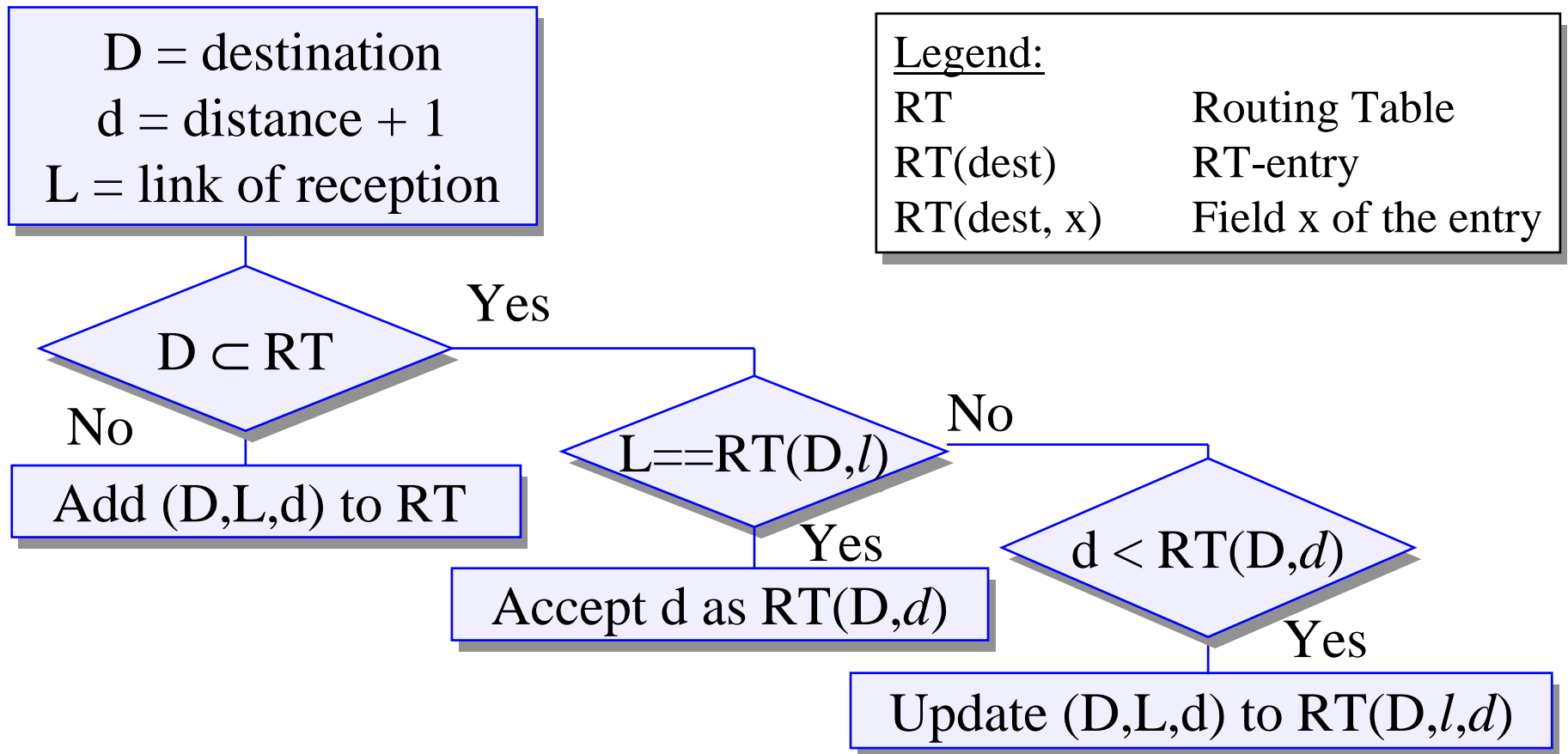
- Kernel reititystaulu

```
# netstat -nr
```

```
Kernel routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
127.0.0.1	*	255.255.255.255	UH	1	0	2130	lo0
191.72.1.0	*	255.255.255.0	U	1	0	3070	eth0
191.72.2.0	191.72.1.1	255.255.255.0	UG	1	0	1236	eth0
191.72.3.0	191.72.1.2	255.255.255.0	UG	1	0	3212	eth0

Processing of Received Distance Vectors



Note: this is simplified, shows only the principle!

RIP vastesanommat

- Etäisyysvektorit lähetetään vastesanomissa
- 30 sekunnin välein
 - Kaikki reititystaulun rivit
 - Eri linkeillä eri etäisyysvektori myrkytettyjen vektoreiden takia
 - Yli 25 riviä → useita sanomia
- Päivitysviestejä muutosten jälkeen
 - Muuttuneet rivit
 - 1-5 sekunnin viiveellä, jotta kaikki samaan muutokseen liittyvät päivitykset saataisiin samaan sanomaan
- Voidaan jättää pois kohteet joiden etäisyys on ääretön, jos next hop pysyy samana.

RIP pyyntösanomat

- Käynnistyessä reititin voi pyytää reititystaulut naapureiltaan
 - Täydellinen lista
 - Samanlainen kuin normaali päivitys (+myrkytetyt vektorit)
- Osa reititystaulusta
 - Virheiden etsintää varten
 - Ei myrkytettyjä vektoreita

Hiljaiset solmut

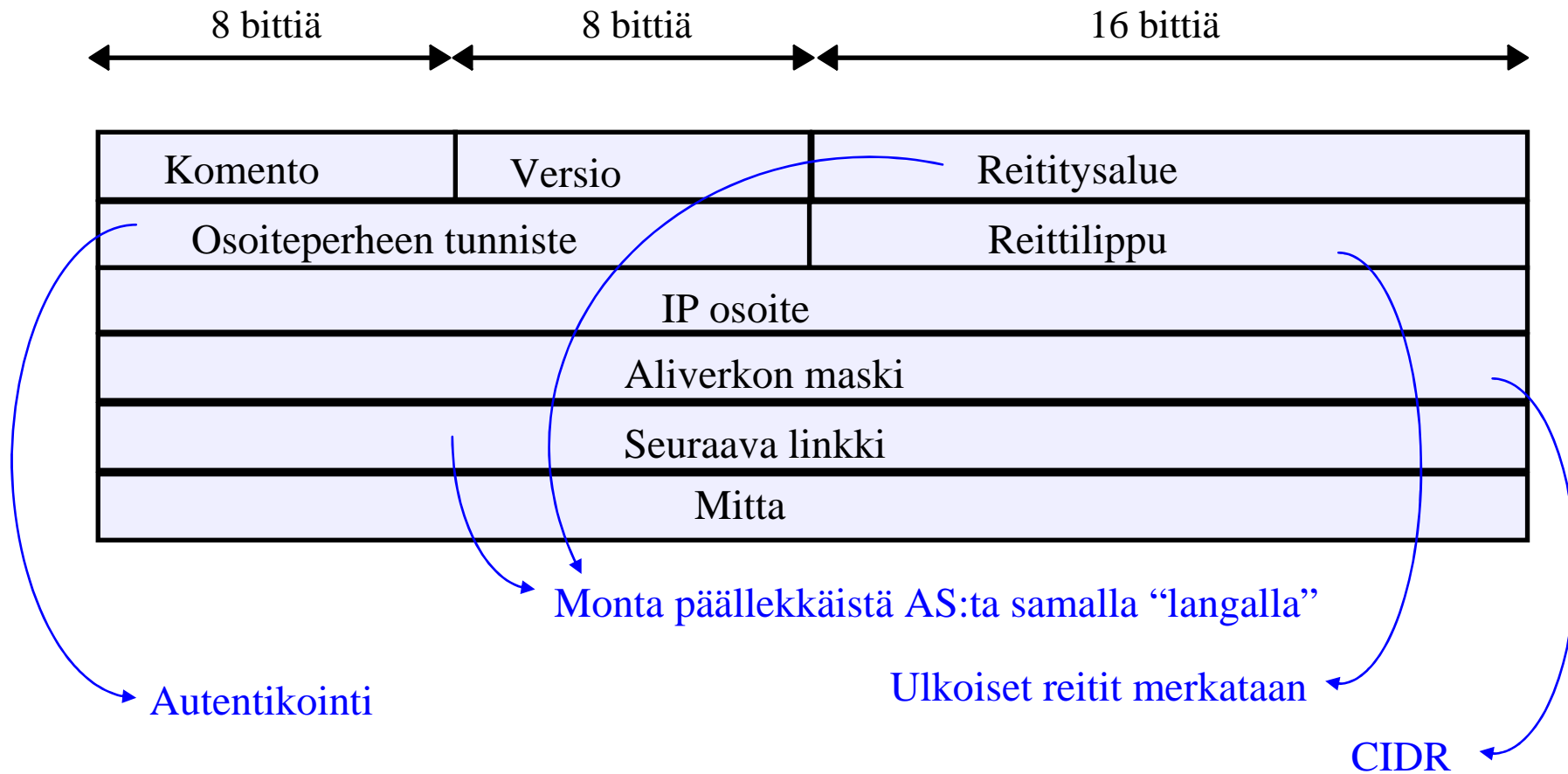
- Kun vain RIP oli käytössä, isäntäkoneet pystyivät kuuntelemaan reititysliikennettä ja ylläpitää omia reititystauluja
 - Mikä reititin on lähinnä kohdetta?
 - Mikä linkki, jos useita linkkejä?
- Nämä olivat ”hiljaisia solmuja”, jotka vain kuuntelivat reititysliikennettä
- Nykyään myös RIP-2, OSPF, IGRP, ...

RIP versio 2

RIP versio 2

- RFC-1388 (1387,1389)
- Miksi?
 - Yksinkertainen ja kevyt vaihtoehto OSPF:lle ja IS-IS:lle
- RIP-2 on rajoitetusti yhteensopiva päivitys
 - RIP-1 kone ymmärtää RIP-2 konetta osittain
- Parannuksia
 - Autentikointi
 - Seuraava linkki -kenttä
 - Aliverkkomaski
 - CIDR tuki
 - Ulkoisia reittejä
 - Päivitykset multicastilla

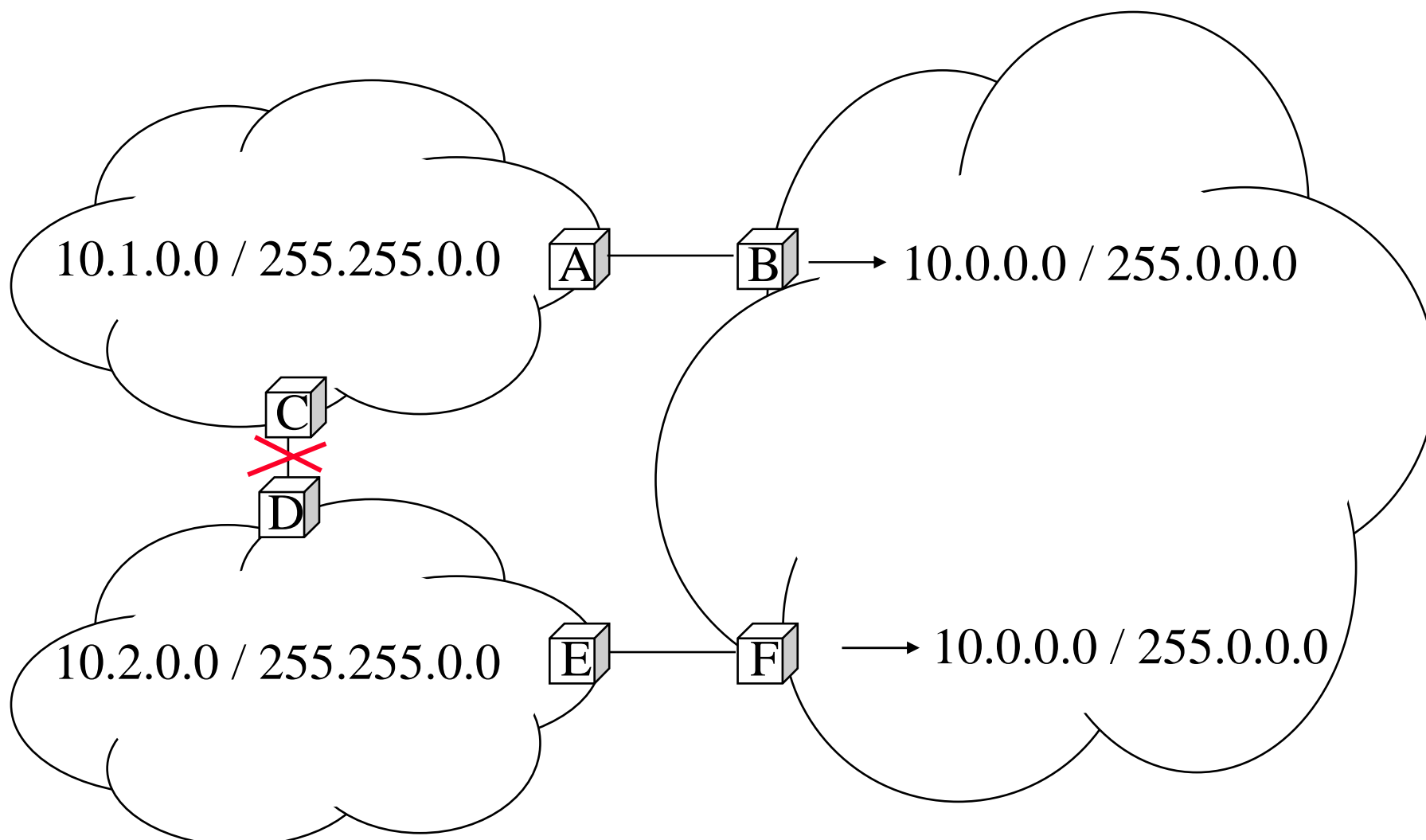
RIP versio 2 – sanomat



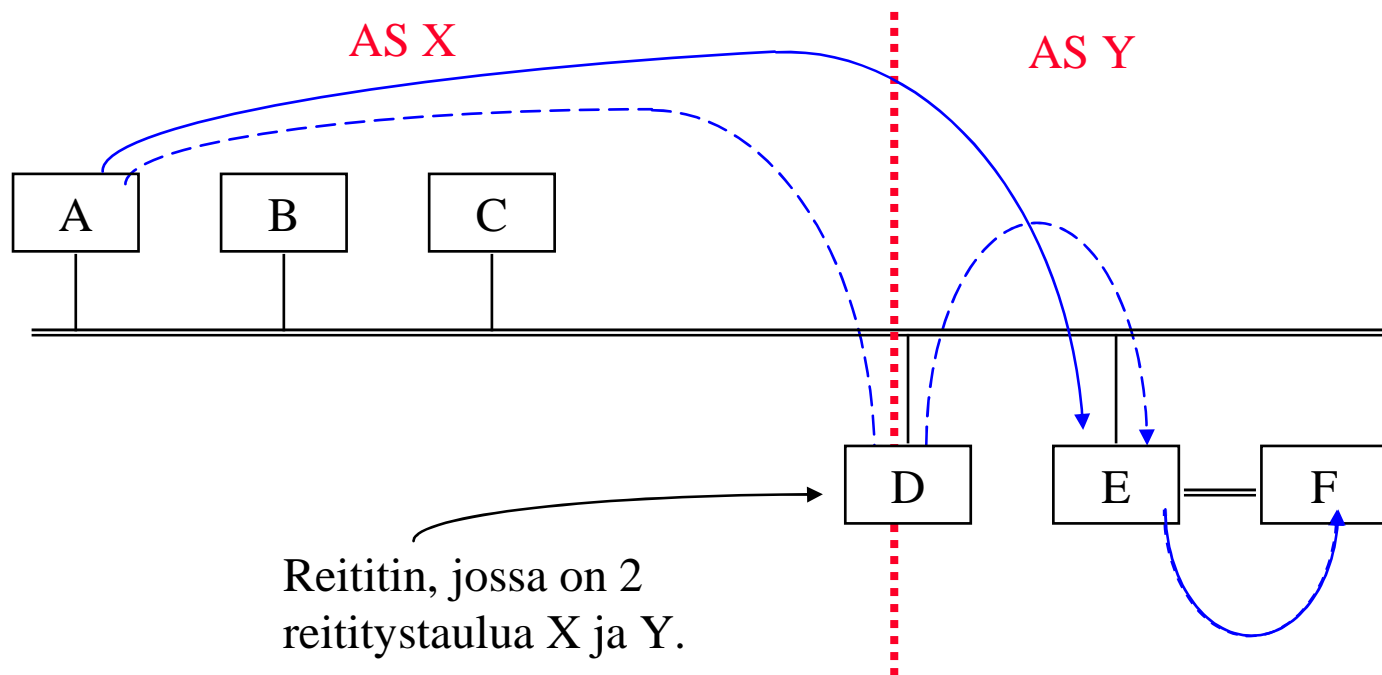
Reititys aliverkosta toiseen (1)

- RIP-1:n aliverkkomaski ei ole tunnettu aliverkon ulkopuolella, vaan ulos kerrotaan ainoastaan verkkonumero
 - ⇒ Aliverkkoja (ja isäntäkoneita) ei voi erottaa toisistaan
 - ⇒ Kaikki aliverkot pitää yhdistää kaikkiin ja ulkoa reititettävä verkon lähimpään reitittimeen aliverkosta riippumatta
- RIP-2 korjaa tilannetta kertomalla ulos aliverkon ja aliverkkomaskin
 - Eri pituisia maskeja saman verkon sisällä
 - CIDR
 - RIP-1 ei ymmärrä

Reititys aliverkosta toiseen (2)



Reititysalue ja seuraava linkki



Seuraava linkki \Rightarrow D mainostaa X:ssä, että etäisyys F:ään on f ja seuraava linkki on E!

Support for local multicast

- RIP-1 broadcasts advertisements to all addresses on the wire
 - Hosts must examine all broadcast packets
- RIP-2 uses a multicast address for advertisements
 - 244.0.0.9
 - No real multicast support needed, since packets are only sent on the local network
- Compatibility problems between RIP-1 and RIP-2

Huomioita RIP:stä (1)

- Reitittimillä on spontaani taipumus synkronoida lähetyshetkensä.
- Tämä lisää virheiden todennäköisyyttä verkossa.
- Siksi lähetysketket satunnaistetaan 15 s ... 45 s välille.
- Syy: lähetysväli = vakio + sanoman pakkausaika + yhtä aikaa tulleiden sanomien käsittelyaika.

Huomioita RIP:stä (2)

- Kun RIP:ä käytetään ISDN linkin yli
 - Uusi puhelu / 30 s \Rightarrow kallista
- Hidas alusverkko \Rightarrow jonojen pituudelle rajoituksia. RIP lähettää sanomansa (25 riviä/sanoma) putkeen \Rightarrow RIP sanomia voi kadota.
- Korjausehdotus perustuu lähetysten kuittausmoodiin, jossa periodisia lähetyksiä ei tapahdu
 - \Rightarrow RIP sanomien puuttuessa oletetaan, että naapuri on edelleen tavoitettavissa
 - \Rightarrow Tieto kaikista vaihtoehtoisista reiteistä talletetaan.