



Luento 8: Verkkokerros

To 21.11.2013

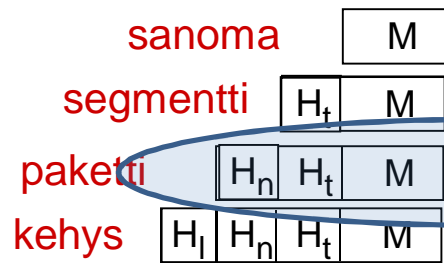
Tiina Niklander

Kurose&Ross
Ch4.1-4.5

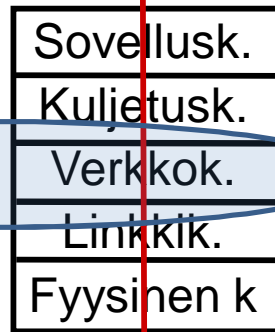
Pääasiallisesti kuvien
© J.F Kurose and K.W. Ross,
All Rights Reserved

Luennon sisältöä

Lähettäjä (sender)



message,
segment
datagram
frame



kytkin

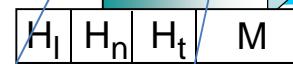
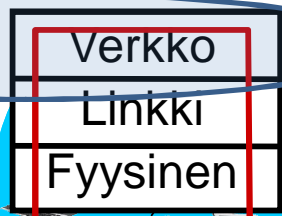
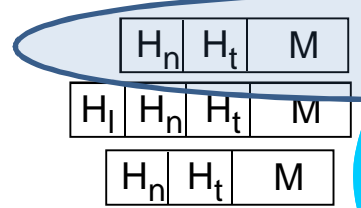
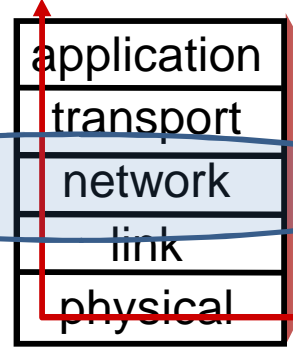
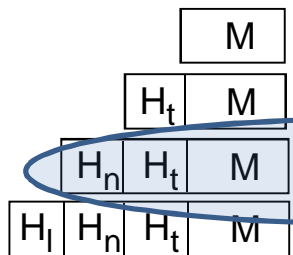


Fig 1.24 [KR12]

Vastaanottaja (recipient)



reititin



Sisältöä

Verkkokerros
Reititin
IP-protokolla
IP-osoitteet, DHCP, NAT
Reititysalgoritmit



Oppimistavoitteet:

- Osaa selittää, kuinka IP-paketteja välitetään verkossa
- Tietää, mitä tietoja sisältyy IP-pakettiin (ja miksi)
- Osaa selittää reitittimen rakenteen ja toiminnan
- Osaa kuvailla, kuinka reitittimet kokoavat reititystietonsa
= linkkitila- ja etäisyysvektorialgoritmien toimintaideat



Verkkokerros

IP-osoitteet



IP-osoitteet (IPv4)

Fig 4.15 [KR12]

32 bittinen tunniste

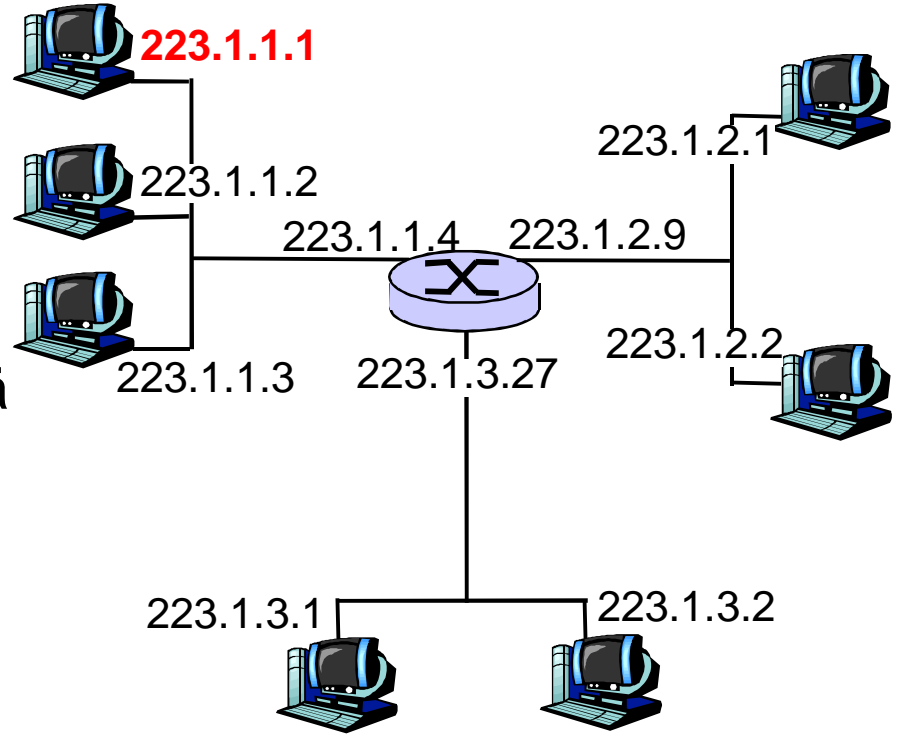
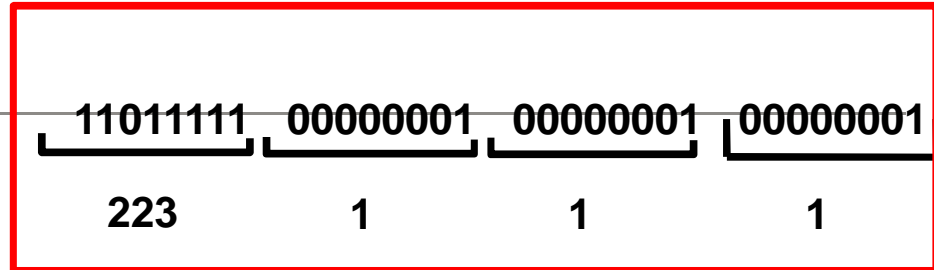
isäntäkoneille ja reitittimien linkeille

- verkkoliittymän tunniste

Reitittimellä useita liittymiä

- kullakin oma IP-osoite

Myös isäntäkone voi olla liitettynä useaan verkkoon



ICANN Internet Corporation for Assigned names and Numbers verkkonumerot palveluntarjoajille, joilta nämä edelleen aliverkoiksi



Aliverkot

HUOM painovirhe: 6ed kuva 4.16
Aliverkon tunnukset.
223.1.1.0/24, 223.1.2.0/24 ja
223.1.3.0/24 (ei 23 kuten kuvassa)

Osoitteen osat

aliverkon numero (alkuosa)

koneen numero (loppuosa)

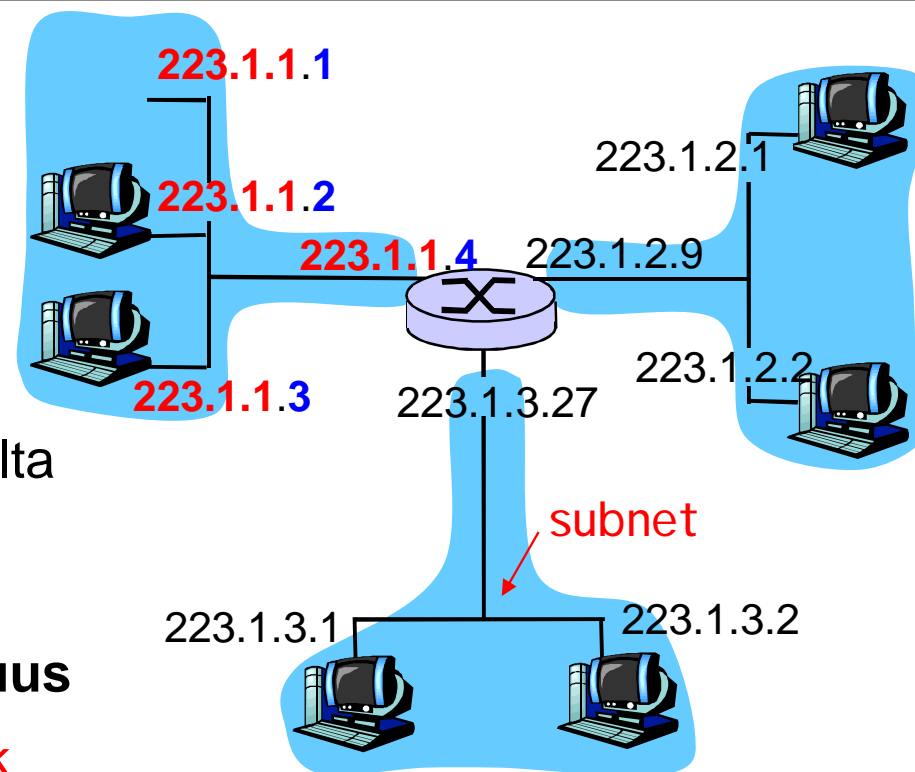
Aliverkon koneet voivat kommunikoida ilman reititystä

Linkkikerros osaa lähettää koneelta toiselle (esim. ethernet)

Aliverkkoa merkitään notaatiolla, jossa lopussa on verkko-osan pituus

Esim. 223.1.1.0 /24 subnet mask

eli verkko-osoite 24 bittiä ja Aliverkon peite
koneosoite 8 bittiä



network consisting of 3 subnets

Fig 4.16 [KR12]



Aliverkot

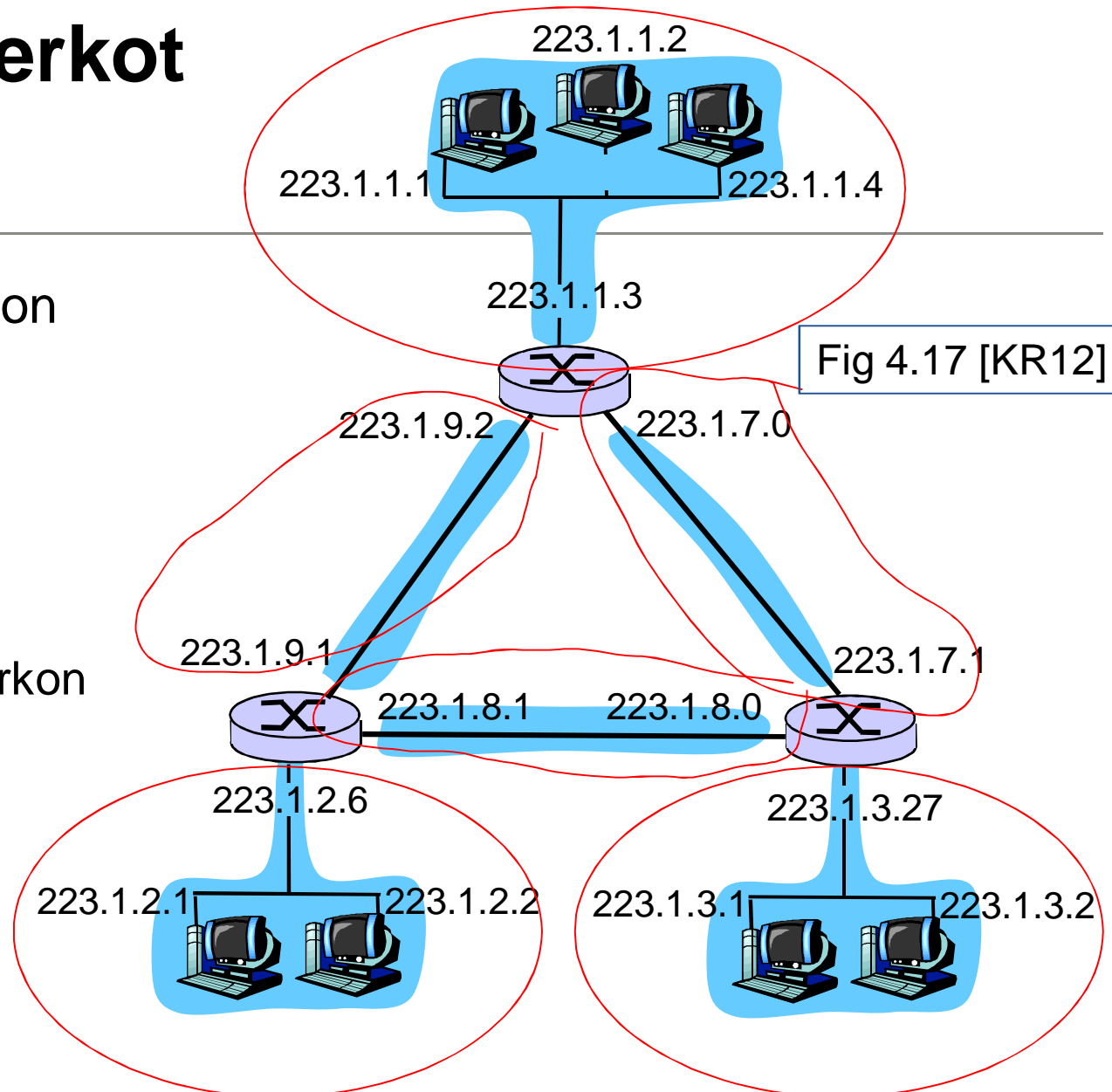
Montako aliverkkoa on tässä kuvassa?

Yleislähetysosoite

255.255.255.255

Paketti kaikille aliverkon koneille.

Mahdollisesti reitittimen kautta muillekin





Aliverkkojen osoitteet:

- Vanha luokallinen osoite: A-luokka 8 b, B-luokka 16 b, C-luokka 24 b
 - Esim. FUNET 128.214.0.0/16 (HY:llä näitä)
 - Edellä esiteltyä subnet maskia käytetään vain osoiteluokan sisällä
- Ongelma: Internet kasvoi 1993 lähtien eksponentiaalisesti ja reititystaulut räjähtivät
 - C-luokat ovat suosittuja ja niitä on paljon
 - Tarvittiin ratkaisu, joka aggregoi reititystietoa paremmin kuin A,B,C jako -> CIDR
- Kaikki IPv4 osoitteet on nyt jaettu!



CIDR: Classless InterDomain Routing

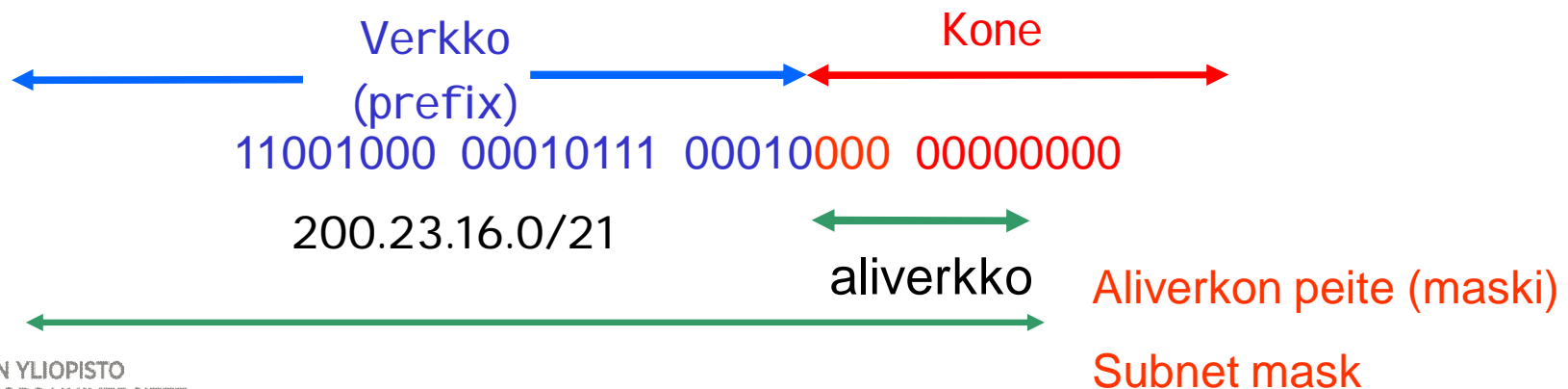
Verkko-osa voi olla minkä tahansa kokoinen

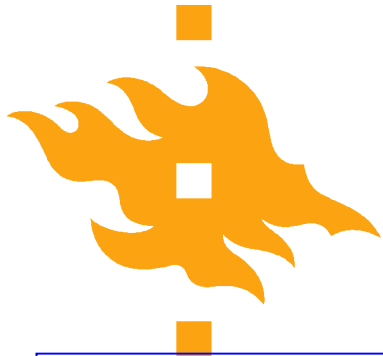
Formaatti: a.b.c.d/x

x ilmoittaa verkko-osan bittien lukumäärän (prefix)

Esim. Organisaatio, jolla 2000 konetta varaa $2048 = 2^{11}$ konenumeroa, jolloin verkko-osaa varten jää 21 bittiä

Yritys voi vielä itse jakaa viimeiset 11 bittiä aliverkko-osoitteeksi ja koneosoitteeksi. Tämä jako ei näy ulkopuolelle. (subnet mask)





Koneen IP-osoite

Palveluntarjoaja saa verkkonumeronsa ICANN:lta isona lohkona
voi jakaa saamansa osoiteavaruuden (osoitelohkon) edelleen
aliverkkoihin

esim. Kukin organisaatio saa aliverkon, jossa on numerot 512 koneelle

$$2^{12} = 4096 = 8 \cdot 512$$

| | | | | |
|----------------|-----------------|-----------------|--------------------------|----------------|
| | ← 20 kpl | → 12 kpl | | |
| ISP's block | <u>11001000</u> | <u>00010111</u> | <u>00010000</u> 00000000 | 200.23.16.0/20 |
| Organization 0 | <u>11001000</u> | <u>00010111</u> | <u>00010000</u> 00000000 | 200.23.16.0/23 |
| Organization 1 | <u>11001000</u> | <u>00010111</u> | <u>00010010</u> 00000000 | 200.23.18.0/23 |
| Organization 2 | <u>11001000</u> | <u>00010111</u> | <u>00010100</u> 00000000 | 200.23.20.0/23 |
| ... | | | | |
| Organization 7 | <u>11001000</u> | <u>00010111</u> | <u>00011110</u> 00000000 | 200.23.30.0/23 |

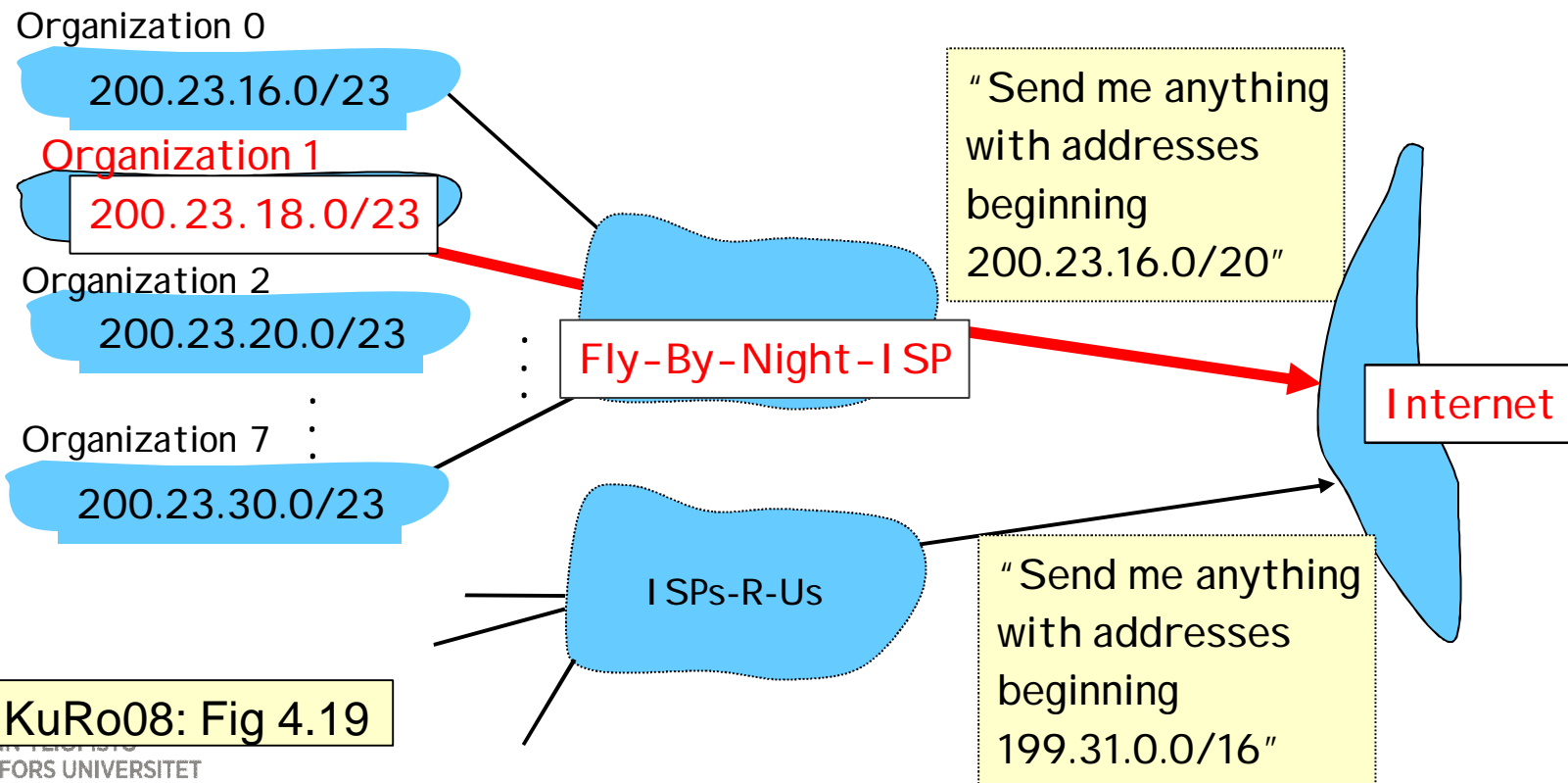


Hierarkkinen osoite

Fig 4.18 [KR12]

CIDR luo reititystä helpottavan hierarkian

Aggregointi (yhdistäminen): yhteinen alkuosa => samaan suuntaan



KuRo08: Fig 4.19



Jos palveluntarjoaja (ISP) vaihtuu?

Fig 4.19 [KR12]

IP-osoitteet voi säilyttää

Uudelta ISP:ltä tarkempi reititysohje

Pisin sopiva alkuosa määrää reitityksen (longest prefix match)

Organization 0

200.23.16.0/23

Organization 2

200.23.20.0/23

Organization 7

200.23.30.0/23

Organization 1

200.23.18.0/23

Fly-By-Night-ISP

ISPs-R-Us

"Send me anything
with addresses
beginning
200.23.16.0/20"

Internet

"Send me anything
with addresses
beginning 199.31.0.0/16
or 200.23.18.0/23"



Koneen IP-osoite

- Koneen IP-osoite konfiguroidaan joko käsin koneelle tai
- saadaan automaattisesti käyttäen **DHCP**:tä (Dynamic Host Configuration Protocol)
 - Eri osoite eri kerroilla tai pysyvämpi osoite
 - DHCP-palvelija vastaa
 - antaa koneen käyttöön IP-osoitteen (rajallinen elinaika)
 - antaa DNS-tiedot, yms.
 - Palvelun tarjoaja: pienempi numeromäärä riittää
 - Toteutus UDP monilähetys (broadcast)
 - "wash-and-go", "plug-and-play"



DHCP: Osoitekysely

Seppo Syrjäsen
luonnehdinnat:

"Hei, onks kellään?"

"Tässois tämmönen..."

"Oi, saanks mä tän?"

"Joo, pidä vaan."

DHCP server: 223.1.2.5



DHCP discover

```
src : 0.0.0.0, 68
dest.: 255.255.255.255,67
yiaddr: 0.0.0.0
transaction ID: 654
```

arriving
client



DHCP offer

```
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 654
lifetime: 3600 secs
```

DHCP request

```
src: 0.0.0.0, 68
dest.: 255.255.255.255, 67
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs
```

*Mahd.
useita
vastauksia,
pitää valita
yksi*

DHCP ACK

```
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs
```



DHCP: Muutakin tietoa kuin IP-osoite

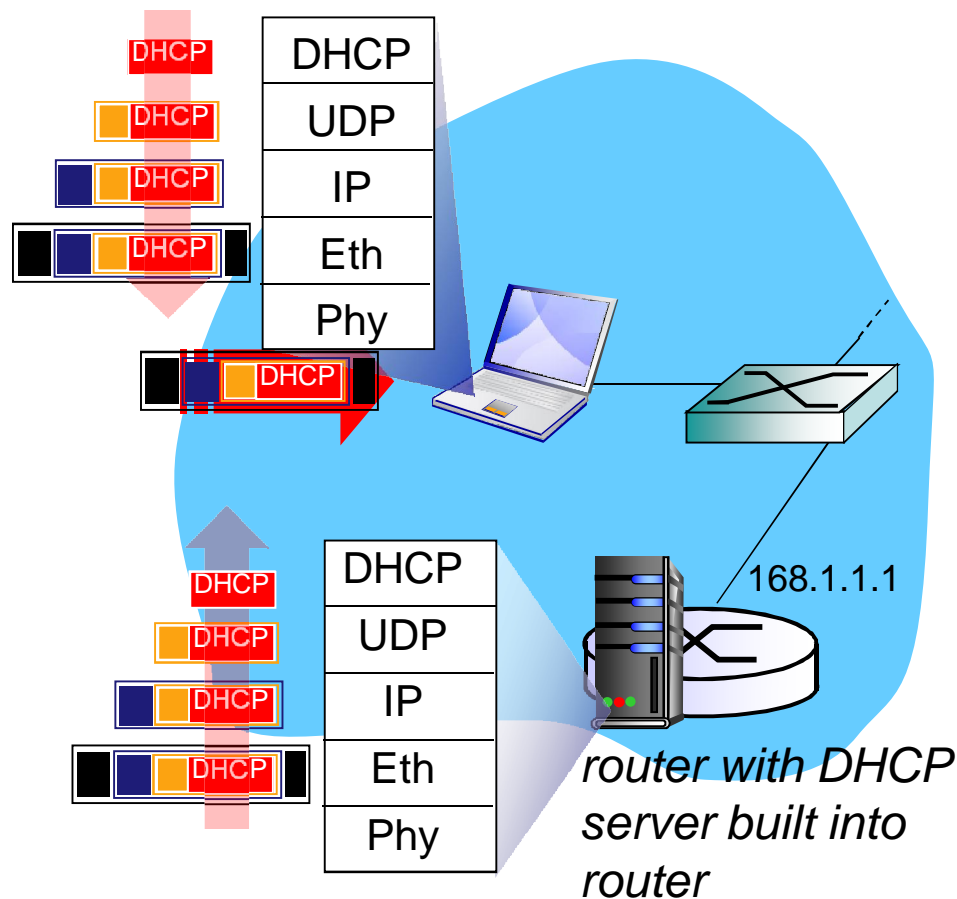
DHCP antaa osoitteen lisäksi yleensä muutakin tärkeää tietoa:

- Reitittimen osoite
 - Yhteyspiste ulos tästä aliverkosta
- Nimipalvelimen (DNS) nimi ja IP-osoite
 - Täältä voi tiedustella muiden osoitteita
- Aliverkon peite (subnet mask)
 - Kertoo mitkä osoitteen bitit ovat verkon osia ja mitkä koneen tunniste aliverkossa



Sanomanvälitys

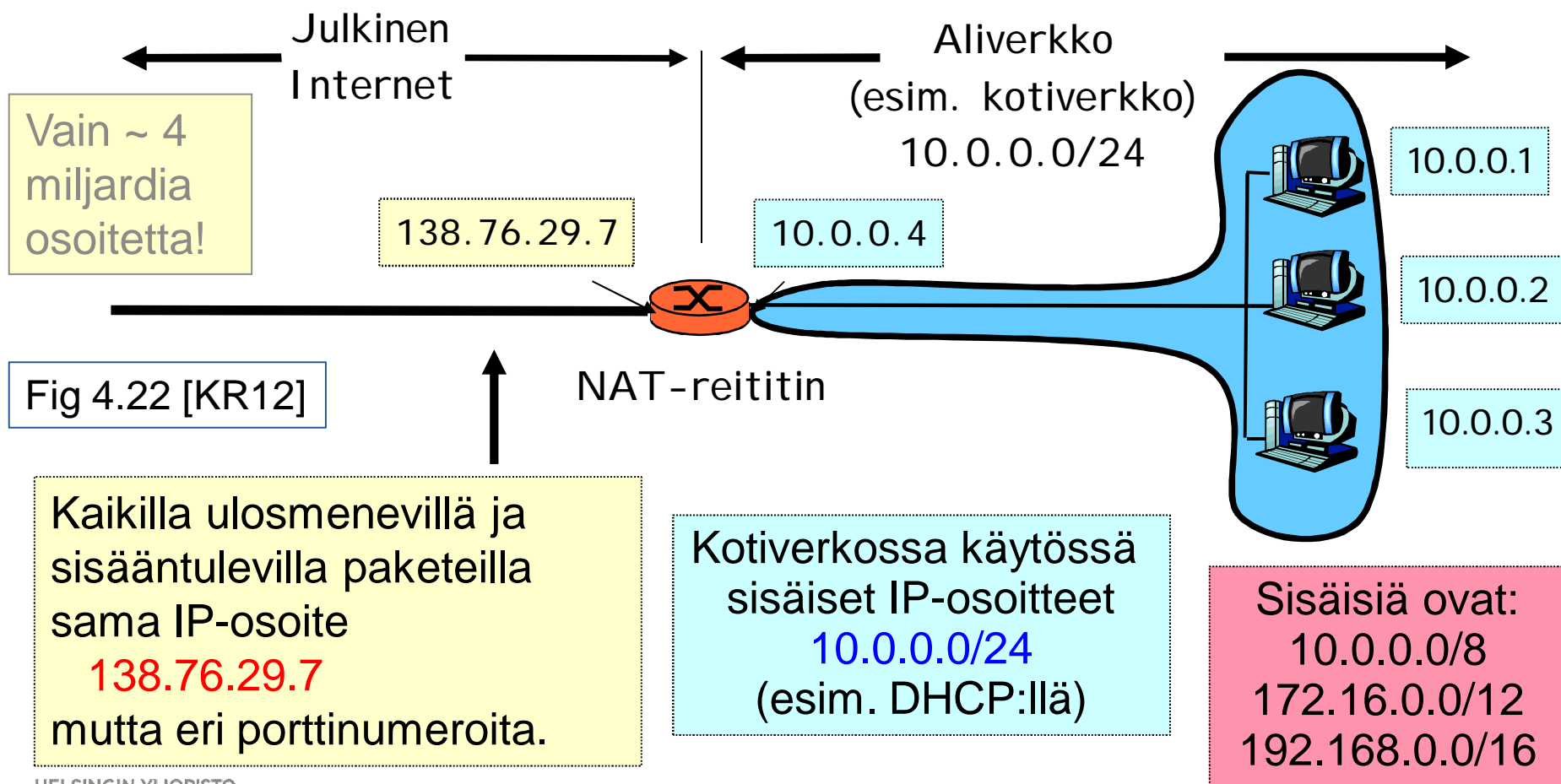
Esimerkkinä DHCP request



- Kannettavan pyyntö: DHCP request
 - sovelluskerros: DHCP request,
 - kuljetuskerros paketoitui UDP-segmentiksi,
 - verkkokerros paketoitui IP-datagrammiksi,
 - linkkikerros paketoitui kehykseksi esim. 802.1
- Palvelimen päässä pino toisinpäin, eli aina puretaan oma paketti ja annetaan ylemmäs sen sanoma.



Aliverkon osoitteiden piilottaminen yhden julkisen IP-osoitteen taakse





NAT-reititin

(Network Address Translation)

Ulosmenevät paketit

Korvaa lähdekoneen IP-osoite ja porttinumero NAT-koneen IP-osoitteella ja NAT-koneen valitseamalla porttinumerolla

Päivitä NAT-muunnostaulu

Sisääntulevat paketit

NAT-koneelle NAT:n antamaan porttiin

Korvaa NAT:n muunnostaulun avulla paketissa oleva IP-osoite ja portti

Välitä paketti perille

NAT-muunnostaulu

(IP-osoite, portti) (NAT-koneen osoite, NAT:n portti)

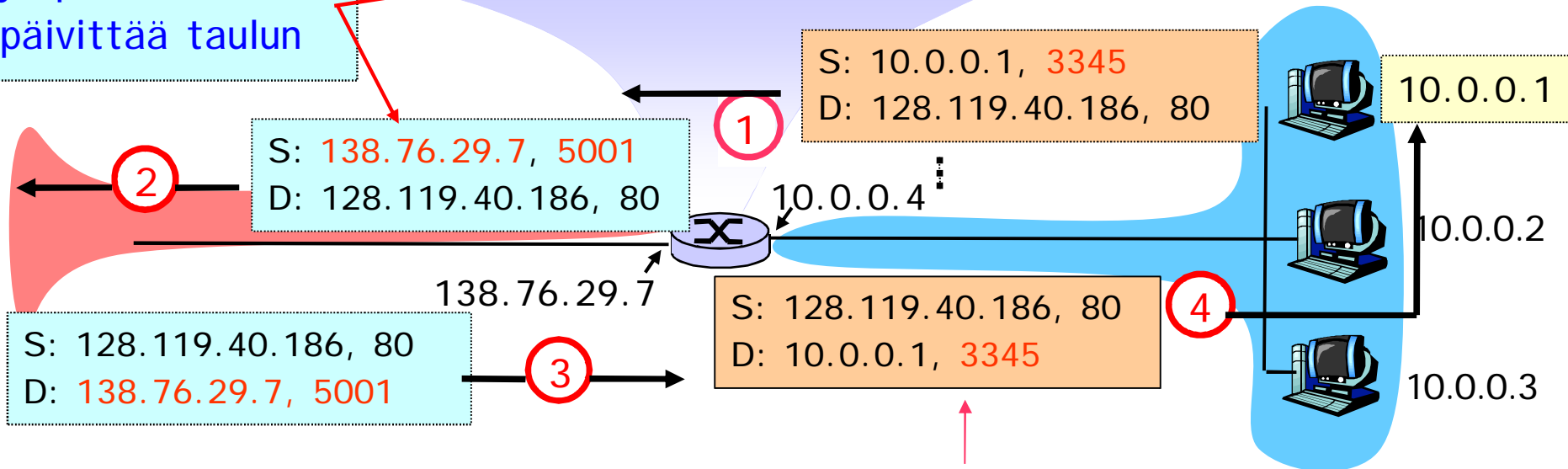


NAT: Esimerkki

| NAT-muunnostaulu | |
|----------------------|-------------------|
| WAN-osoite | LAN-osoite |
| 138.76.29.7, 5001 | 10.0.0.1, 3345 |

2: NAT vaihtaa lähdeosoitteeksi 138.76.29.7, ja portiksi 5001, päivittää taulun

1: kone 10.0.0.1 lähettää paketin 128.119.40.186, 80



3: Vastaus NAT-koneelle: 138.76.29.7, 5001

4: NAT vaihtaa kohdeosoitteeksi 10.0.0.1 ja portiksi 3345



NAT: Kommentteja / kritiikkiä

Hyödyt

Kotiverkko tarvitsee ISP:ltä vain yhden IP-osoitteen

Voi muuttaa vapaasti kotikoneiden IP-osoitteita

Turvallisuus: ulospäin muille näkyy vain yksi kone

Kritiikkiä

Reitittimien tulisi toimia vain verkkotasolla, porttinumerot ovat kuljetuskerroksen asioita

Rikkoo päästä-päähän idean (prosessien välinen yhteys)

Onko ohjelmoijan huomioitava NAT:n olemassaolo?

- Peer-to-peer
- NAT:n takana oleva palvelin (esim. www portissa 80)?

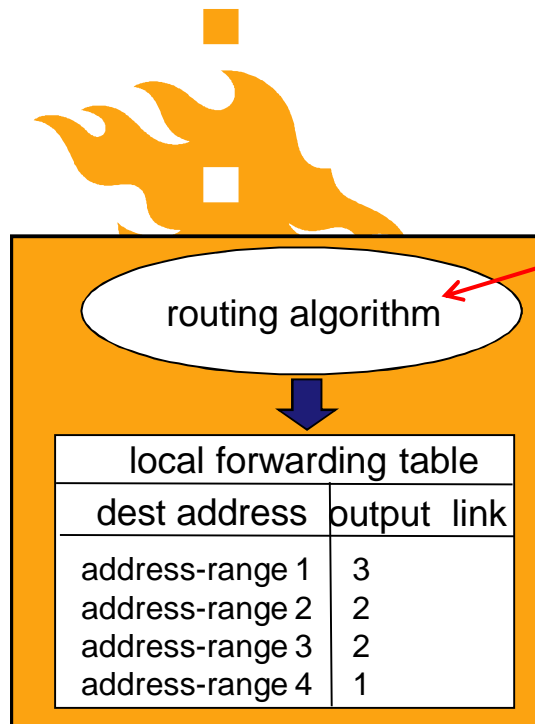
Pula IP-osoitteista hoidettava ottamalla käyttöön IPv6, jossa 128 bitin osoitteet



Verkkokerros

Reititysalgoritmit

Reititysalgoritmi



IP destination address in arriving packet's header

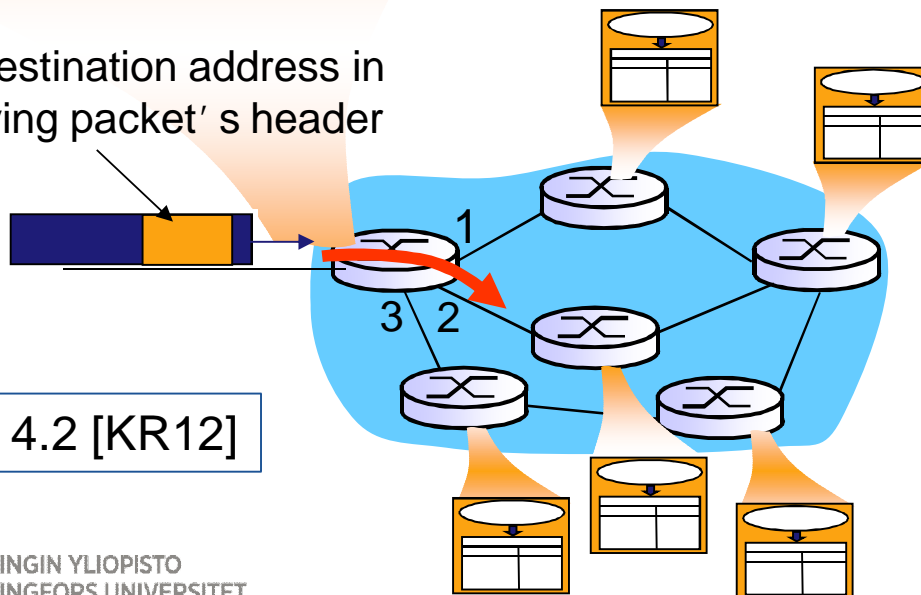


Fig 4.2 [KR12]

- Etsii edullisimmat reitit lähdekoneelta kohdekoneille
 - Mikä ja millainen algoritmi? Mistä tiedot? ...
- Muodostaa reititystaulun!
 - Mille linkille paketti seuraavaksi siirretään tältä reitittimeltä
- Pakettien edelleenlähetys (forwarding) tarvitsee reititystaulua, jotta paketit löytävät perille



Reititysalgoritmeja

Reititysalgoritmi, joka tarvitsee täydellisen tiedon verkosta

Ennen laskentaa käytössä koko kuva verkosta:

- Kaikki linkkiyhteydet solmujen välillä ja niiden kustannukset
- Käytännössä vain tietystä autonomisesta alueesta

Parhaat reitit lasketaan joko keskitetysti tai hajautetusti

Linkkitila-algoritmi (link-state algorithm)

Reititysalgoritmi, jolle riittää epätäydellinen kuva verkosta

Aluksi reititin tietää vain niistä koneista, joihin itse on yhdistetty

Iteratiivinen algoritmi: reititin vaihtaa tietoja naapuriensa kanssa ja saa tietoa muusta verkosta

Etäisyysvektorialgoritmi
(distance vector algorithm)



Reititysalgoritmin muita ominaisuuksia

Dynaaminen vs. staattinen

Miten nopeasti huomaa linkkien muutokset ja muuttaa reititystä

Miten tiuhaan tietoja päivitetään

Miten usein muutoksia

Kuormituksen huomioiva vai ei

Linkin ruuhkautuneisuus voi vaikuttaa sen kustannukseen

Nykyalgoritmit eivät ota kuormitusta huomioon

- Tosin kyllä epäsuorasti linkin hitautena ('kustannuksena')



Reititystiedon kerääminen

Reititin saa tietoja seuraavasti:

Linkkikerros tarjoaa yhteyden naapureihin

– MAC-osoite → IP-osoite

Naapurit havaitaan saapuvista kehyksistä ja paketeista (broadcast, multicast, unicast)

Naapurit kertovat omasta reititystaulustaan

Linkkitila: kaikki tiedot

Etäisyysvektori: lyhimät etäisyydet kohteisiin naapurien kautta

Tietojen päivitys: reaktiivinen tai ajastettu



Verkko graafina (graph)

Fig 4.27 [KR12]

Verkko $G = (N, E)$

N = solmujen (nodes) joukko

E = linkkien (edges) joukko

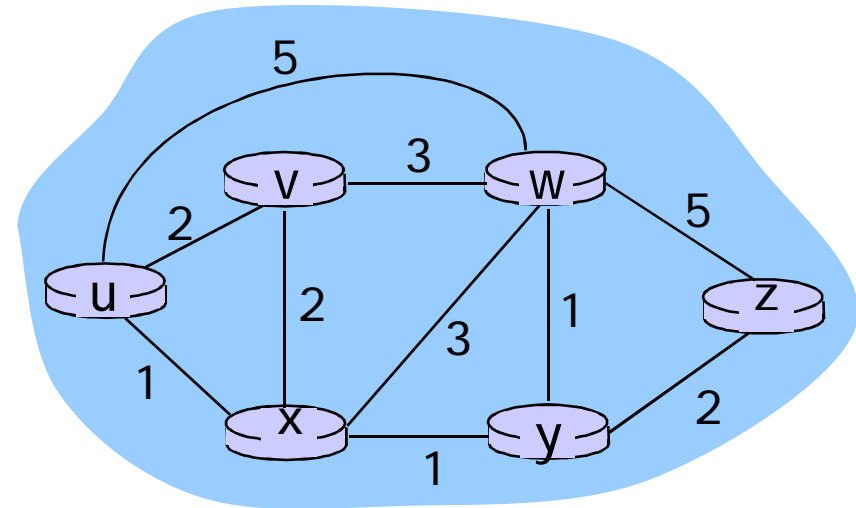
(x, y) on linkki solmujen x ja y välillä

$c(x, y)$ = linkin kustannus

kaistanleveys, ruuhkaisuus, raha, ..

$C(x_1, x_2, \dots, x_p)$ = reitin (route) kustannus

= $C(x_1, x_2) + C(x_2, x_3) + \dots + C(x_{p-1}, x_p)$



Mikä on huokein reitti kuvan solmusta u solmuun z ?



1) Linkkitila: Dijkstran algoritmi

Aluksi kaikilla reitittimillä on tiedossa verkon rakenne ja kaikkien linkkien kustannukset

Kaikki reitittimet lähettävät tietonsa naapureistaan ja niihin johtavien linkkien kustannuksista (mitatut/havaitut) joko kaikille muille tai jollekin keskussolmulle, joka välittää tiedon muille

Reititin laskee Dijkstran algoritmilla edullisimman kustannuksen kaikkiin muihin kohteisiin

Kokoaa näistä oman reititystaulunsa



Dijkstran algoritmi

Merkinnät

$C(x,y)$ linkin x,y kustannus; jos eivät naapureita = ∞

$D(v)$ toistaiseksi edullisin kustannus solmuun v

$p(v)$ solmun v edeltäjä reitillä

N = solmujen joukko,

N' = jo käsiteltyjen solmujen joukko



Dijkstran algoritmi

$D(v)=2, D(w) = 5, D(x)=1$
 $D(y) = \infty, D(z)= \infty$

1 **Initialization:**

2 $N' = \{u\}$

3 for all nodes a

4 if a **adjacent** to u

5 then $D(a) = c(u,a)$

6 else $D(a) = \infty$

7

8 **Loop**

9 find b not in N' such that $D(b)$ is a minimum

10 **add b to N'**

11 update $D(k)$ for all k adjacent to b and not in N' :

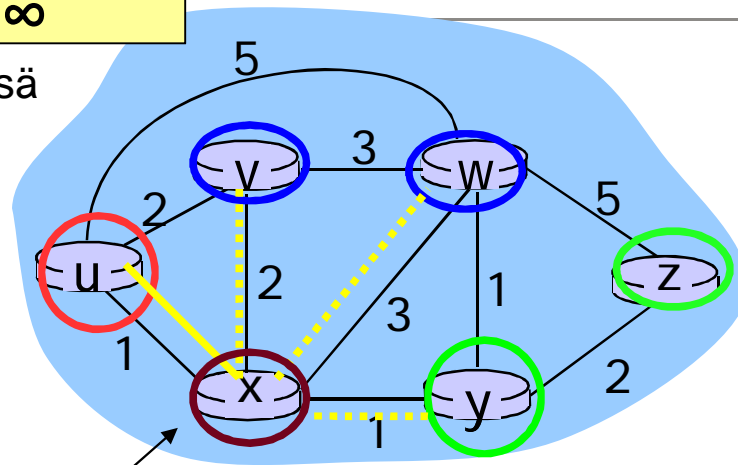
12 $D(k) = \min(D(k), D(b) + c(b,k))$

13 /* new cost to k is either old cost to k or known

14 shortest path cost to b plus cost from b to k */

15 **until all nodes in N'**

1. Eli jos u:n vieressä



2. Aina valitaan käsittelemätön, jonka etäisyys u:sta on pienin

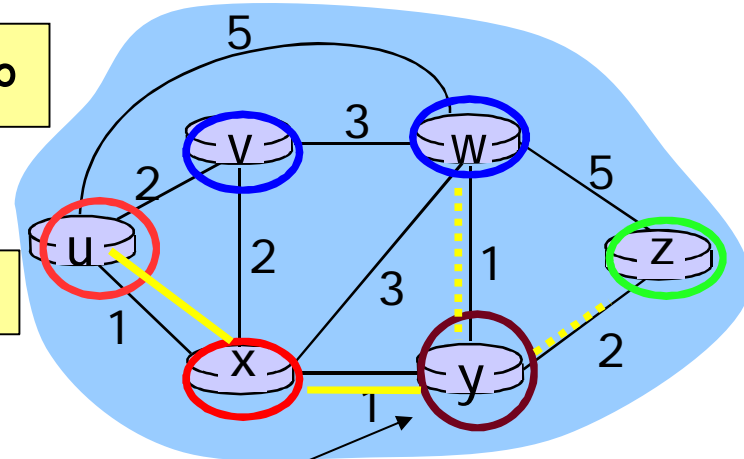
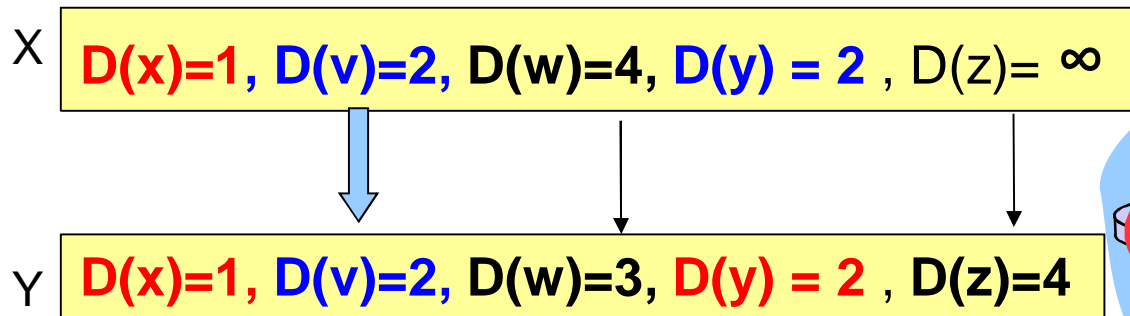
3. Päivitetään etäisyys b:n naapureille, joita ei vielä ole käsitelty

$N'=(u,x)$



Dijkstran algoritmi (kierros 2)

Huom. Etäisyys
aina $u \rightarrow$ solmu



8 Loop

- 9 find b not in N' such that $D(b)$ is a minimum
- 10 add b to N'
- 11 update $D(k)$ for all k adjacent to b and not in N' :
- 12 $D(k) = \min(D(k), D(b) + c(b,k))$
- 13 /* new cost to k is either old cost to k or known
- 14 shortest path cost to b plus cost from b to k */
- 15 until all nodes in N'

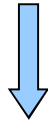
Tässä olisi voitu
valita y tai v , mutta
valittiin nyt y

$N'=(u,x,y)$

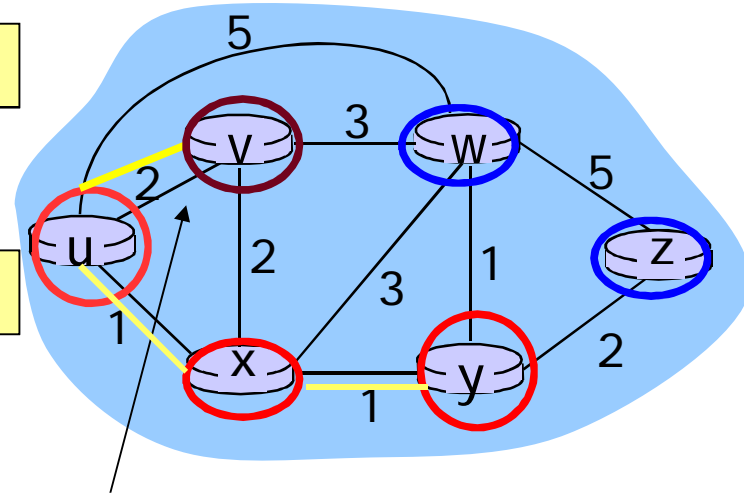


Dijkstran algoritmi (kierros 3)

Y $D(x)=1, D(y) = 2, D(v)=2, D(w)=3, D(z)=4$



V $D(x)=1, D(y) = 2, D(v)=2, D(w)=3, D(z)=4$



8 Loop

9 find b not in N' such that $D(b)$ is a minimum

10 add b to N'

11 update $D(k)$ for all k adjacent to b and not in N' :

12 $D(k) = \min(D(k), D(b) + c(b,k))$

13 /* new cost to k is either old cost to k or known

14 shortest path cost to b plus cost from b to k */

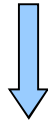
15 until all nodes in N'

$N'=(u,x,y,v)$

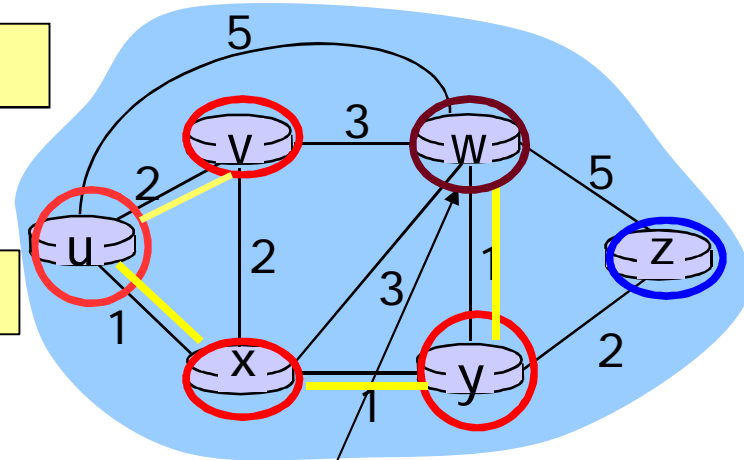


Dijkstran algoritmi (kierros 4)

V $D(x)=1, D(y) = 2, D(v)=2, D(w)=3, D(z)=4$



W $D(x)=1, D(y) = 2, D(v)=2, D(w)=3, D(z)=4$



8 Loop

9 find b not in N' such that $D(b)$ is a minimum

10 add b to N'

11 update $D(k)$ for all k adjacent to b and not in N' :

12 $D(k) = \min(D(k), D(b) + c(b,k))$

13 /* new cost to k is either old cost to k or known

14 shortest path cost to b plus cost from b to k */

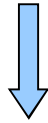
15 until all nodes in N'

$N'=(u,x,y,v,w)$

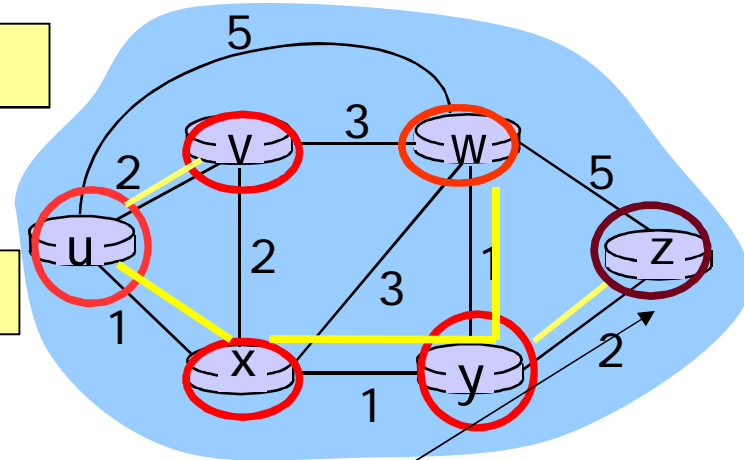


Dijkstran algoritmi (kierros 5)

W $D(x)=1, D(y) = 2, D(v)=2, D(w)=3, D(z)=4$



Z $D(x)=1, D(y) = 2, D(v)=2, D(w)=3, D(z)=4$



8 Loop

9 find b not in N' such that $D(b)$ is a minimum

10 add b to N'

11 update $D(k)$ for all k adjacent to b and not in N' :

12 $D(k) = \min(D(k), D(b) + c(b,k))$

13 /* new cost to k is either old cost to k or known

14 shortest path cost to b plus cost from b to k */

15 until all nodes in N'

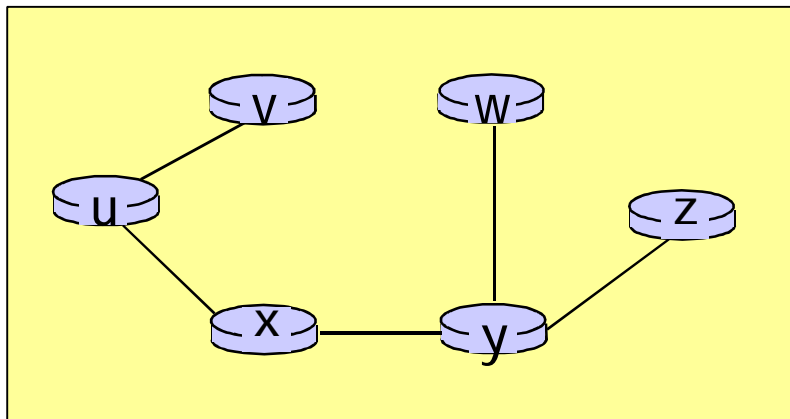
$N'=(u,x,y,v,w,z)$



Lyhyimmät reitit ja reititystaulukko

Fig 4.27 [KR12]

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) |



Dijkstran algoritmi: Havaintoja

- Dijkstran algoritmi ottaa käsittelyyn aina lähimmän käsittelemättömän solmun
- Hakee lyhintä mahdollista etäisyyttä naapurin naapurien kautta
- Tästä seuraa lyhin virittävä puu → reititystaulu



2) Etäisyysvektoreititys (distance vector)

- Arpanet-verkon alkuperäinen reititysalgoritmi
 - Käytössä useissa Internetin reititysprotokollissa, kuten RIP, BGP, Novell IPX, ISO IDR
- Interaktiivinen, hajautettu ja asynkroninen
- Tiedot tarkentuvat asteittain, iteratiivisesti
 - Tietyin väliajoin,
 - linkin tilan vaihtuessa,
 - naapurin tietojen muuttuessa, ..
- Kukin solmu laskee itsenäisesti, mutta saa tietoa naapureiltaan
 - Tietää / arvioi kustannuksen omiin naapureihinsa
 - Kuulee naapureiden kustannukset muihin kohdesolmuihin, jotka nämä puolestaan ovat kuulleet omilta naapureiltaan
 - Valitsee kullekin kohdesolmulle kuulemansa edullisimman reitin



Etäisyysvektoreititys (jatkuu)

Kullakin reitittimellä etäisyysvektori sen tuntemiin solmuihin

Reititystaulu, jossa kullekin kohteelle ulosmenolinkki ja kustannus (etäisyys)

- Aika /etäisyys kohteeseen, hyppyjen lukumäärä, arvioitu viive,..

Reititin tietää/mittaa kustannuksen omiin naapureihinsa

Jos muutoksia, lähettää etäisyysvektorinsa naapureilleen

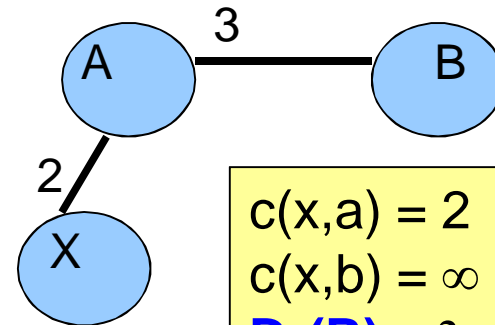
Kun saa naapurinsa etäisyysvektorin, päivittää oman etäisyysvektorinsa

Tietoja uusista solmuista => lisää taulukkoon uudet kohteet

Tietoja jo tunnetuista solmuista: valitse kustannuksiltaan edullisin reitti



Etäisyysvektori- reititys



$$\begin{aligned}c(x,a) &= 2 \\c(x,b) &= \infty \\D_a(B) &= 3 \\D_x(B) &= 2 + 3 = 5\end{aligned}$$

Merkinnät

$c(x,v)$ kustannus solmusta x naapuriin v ,
jos v ei ole x :n naapuri, $c(x,v) = \infty$

$D_x(y)$ edullisimman x :stä y :hyn johtavan reitin kustannus

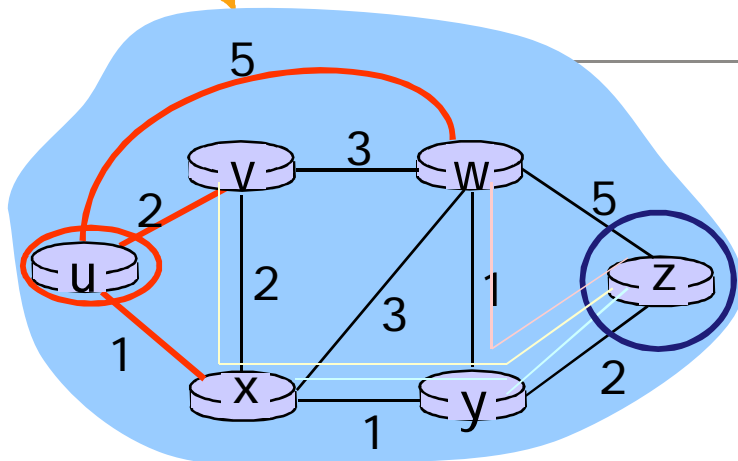
Solmun x oma etäisyysvektori $D_x = [D_x(y): y \in N]$
edullisin tiedetty kustannus solmusta x kuhunkin tunnettuun solmuun y

Naapureiden etäisyysvektorit $D_v(y) = [D_v(y): y \in N] =$
Naapurin v tiedot edullisimmista kustannuksista kuhunkin solmuun y

$$D_x(y) = \min \{c(x,v) + D_v(y)\} \quad (\text{Bellman-Ford})$$

Kustannus solmusta x naapurisolmuun v ja sieltä edelleen solmuun y
Jos useita reittejä (eri naapureiden kautta); valitaan edullisin eli pienin kustannus

Esimerkki 1



Jos on jo saatu selville, että
 $D_v(z) = 5$, $D_x(z) = 3$, $D_w(z) = 3$

$$D_u(z) = \min \{ c(u,v) + d_v(z), \\ c(u,x) + d_x(z), \\ c(u,w) + d_w(z) \}$$

$$= \min \{ 2 + 5, \\ 1 + 3, \\ 5 + 3 \} = 4$$

| Kohde | kust. | linkki |
|----------|----------|----------|
| Z | 4 | X |

Kun paketti on matkalla solmusta u solmuun z, se tulee seuraavaksi lähettää solmuun x, joka tuotti tuon minimin => talleta tieto omaan etäisyysvektoriin (= reititystauluun)



ESIMERKKI 2.

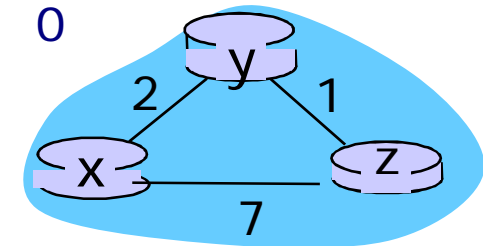
Alussa kukin solmu tuntee vain etäisyydet naapureihinsa itsensä kautta:

| | | cost to | | |
|------|---|----------|----------|----------|
| X: | | x | y | z |
| from | x | 0 | 2 | 7 |
| | y | ∞ | ∞ | ∞ |
| | z | ∞ | ∞ | ∞ |

| | | cost to | | |
|------|---|----------|----------|----------|
| y: | | x | y | z |
| from | x | ∞ | ∞ | ∞ |
| | y | 2 | 0 | 1 |
| | z | ∞ | ∞ | ∞ |

| | | cost to | | |
|------|---|----------|----------|----------|
| Z: | | x | y | z |
| from | x | ∞ | ∞ | ∞ |
| | y | ∞ | ∞ | ∞ |
| | z | 7 | 1 | 0 |

Sitten solmut lähettävät omat reittinsä toisilleen ja laskevat uudet parhaat reitit.



Esimerkiksi solmu x:

| | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 7 | 1 | 0 |

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

$$= \min\{2+0, 7+1\} = 2$$

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

$$= \min\{2+1, 7+0\} = 3$$

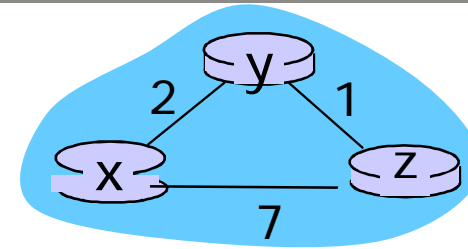


Esimerkki 2 jatkuu:

Samalla tavalla toimivat solmut y ja z:

| | | cost to | | |
|------|----|---------|---|---|
| | | x | y | z |
| from | y: | 0 | 2 | 7 |
| | x | 2 | 0 | 1 |
| | z | 7 | 1 | 0 |

| | | cost to | | |
|------|----|---------|---|---|
| | | x | y | z |
| from | Z: | 0 | 2 | 7 |
| | x | 2 | 0 | 1 |
| | z | 3 | 1 | 0 |



Solmut lähettävät taas tietonsa toisilleen ja laskevat uudet lyhimmät reitit.

| | | cost to | | |
|------|----|---------|---|---|
| | | x | y | z |
| from | X: | 0 | 2 | 3 |
| | y | 2 | 0 | 1 |
| | z | 3 | 1 | 0 |

| | | cost to | | |
|------|----|---------|---|---|
| | | x | y | z |
| from | Y: | 0 | 2 | 3 |
| | x | 2 | 0 | 1 |
| | z | 3 | 1 | 0 |

| | | cost to | | |
|------|----|---------|---|---|
| | | x | y | z |
| from | Z: | 0 | 2 | 3 |
| | y | 2 | 0 | 1 |
| | z | 3 | 1 | 0 |



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

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

X:

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

| | | cost to | | |
|------|---|---------|---|---|
| | | x | y | z |
| from | x | 0 | 2 | 3 |
| | y | 2 | 0 | 1 |
| | z | 7 | 1 | 0 |

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

y:

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

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

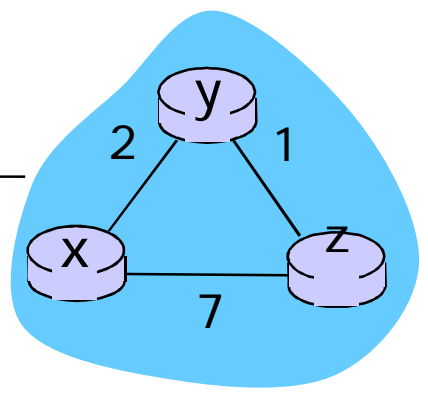
| | | cost to | | |
|------|---|---------|---|---|
| | | x | y | z |
| from | x | 0 | 2 | 3 |
| | y | 2 | 0 | 1 |
| | z | 3 | 1 | 0 |

z:

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

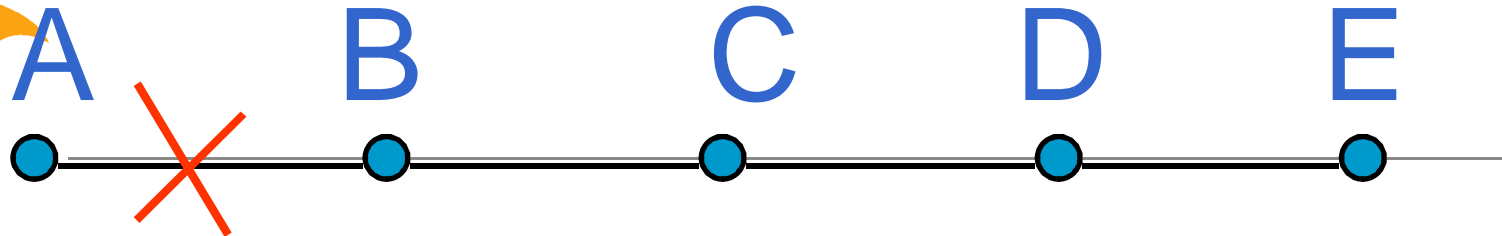
| | | cost to | | |
|------|---|---------|---|---|
| | | x | y | z |
| from | x | 0 | 2 | 7 |
| | y | 2 | 0 | 1 |
| | z | 3 | 1 | 0 |

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





Hyvä uutinen etenee nopeasti



Aluksi yhteys

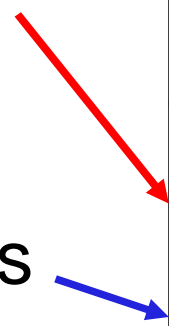
A:han

on poikki

ja sitten

linkki AB

toimii taas



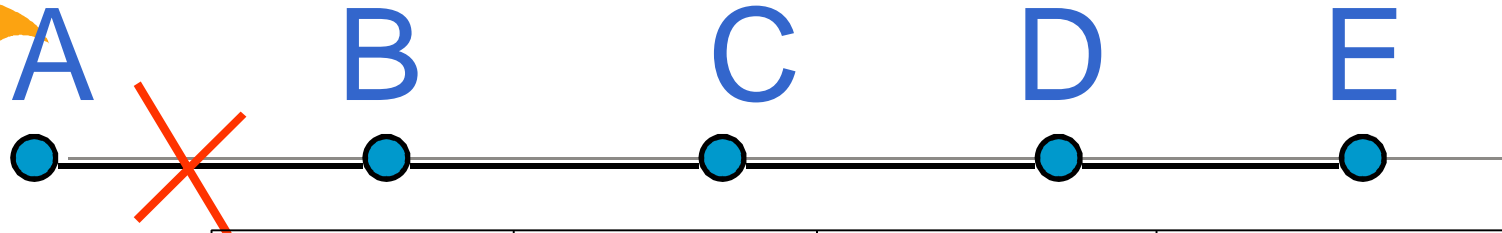
Etäisyys A:han

| $D_B(A)$ | $D_C(A)$ | $D_D(A)$ | $D_E(A)$ |
|----------|----------|----------|----------|
| ääretön | ääretön | ääretön | ääretön |
| 1 | ääretön | ääretön | ääretön |
| 1 | 2 | ääretön | ääretön |
| 1 | 2 | 3 | ääretön |
| 1 | 2 | 3 | 4 |

Tieto etenee joka vaihdossa yhden linkin yli



Huono uutinen etenee hitaasti!



Linkki AB
katkeaa =>

Etäisyys
äärettömäksi

Joka vaihdossa
'paras arvio'
huononee vain
yhdellä =
reitityssilmukka

Count-to-infinity -
ongelma

| | $D_B(A)$ | $D_C(A)$ | $D_D(A)$ | $D_E(A)$ |
|--|----------|----------|----------|----------|
| | ∞ | 2 | 3 | 4 |
| | 3 | 2 | 3 | 4 |
| | 3 | 4 | 3 | 4 |
| | 5 | 4 | 5 | 4 |
| | 5 | 6 | 5 | 6 |
| | 7 | 6 | 7 | 6 |
| | 7 | 8 | 7 | 8 |
| | jne | | | |

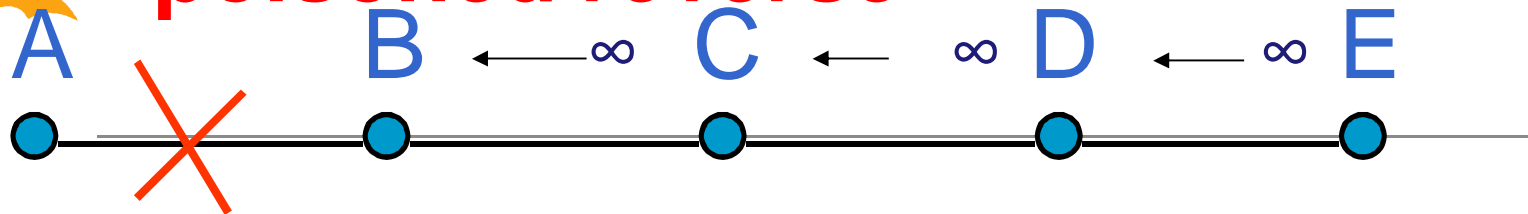
$D_C(A)$ mainostaa kahden hypyn linkkiä A:han

Etäisyys A:han



Huono uutinen etenee nopeasti:

"poisoned reverse"



Ratkaisu count-to-infinity-ongelmaan!

Ilmoita etäisyys äärettömäksi naapurille, jonka kautta linkki kulkee. Kerro muille oikea etäisyys.

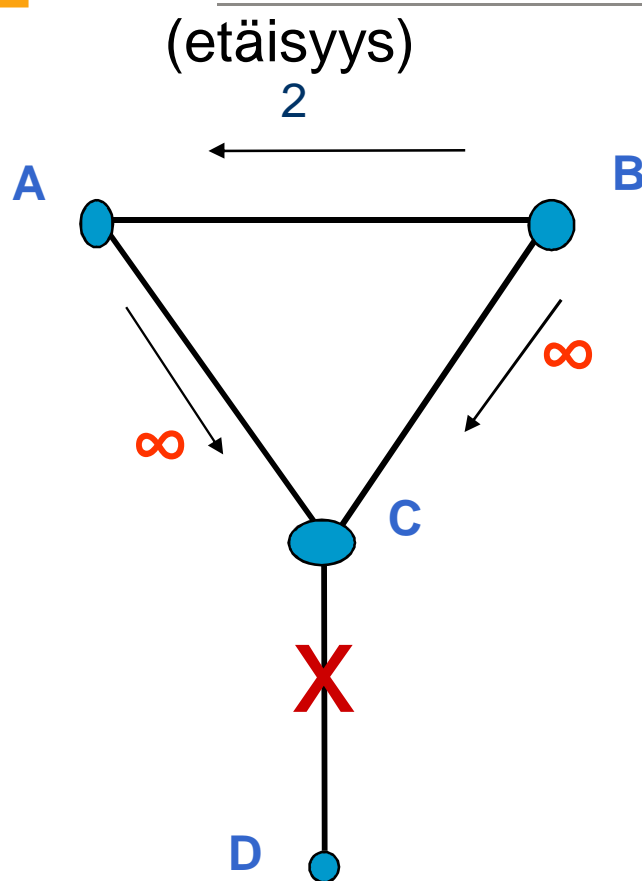
Etäisyys A:han

| $D_B(A)$ | $D_C(A)$ | $D_D(A)$ | $D_E(A)$ |
|----------|----------|----------|----------|
| ∞ | 2 | 3 | 4 |
| ∞ | ∞ | 3 | 4 |
| ∞ | ∞ | ∞ | 4 |
| ∞ | ∞ | ∞ | ∞ |

Tieto etenee joka vaihdossa yhden linkin yli



Ratkaisu ei toimi aina!



Linkki CD katkeaa, A ja B ilmoittavat C:lle, ettei D:hen pääse (käytössä 'poisonous reverse' eli etäisyys "ääretön")

C päätelee (oikein), että D:tä ei voi saavuttaa ja kertoo tämän A:lle ja B:lle eli että $c(C,D) = \infty$

Mutta A kuulee B:ltä, että sillä on etäisyys 2 D:hen \Rightarrow A:n oma etäisyys D:hen $:= 3$ ja tämä reitti ei kulje C:n kautta! \Rightarrow kerrotaan C:lle.

C kertoo B:lle, ...



Algoritmien aikavaativuus

Dijkstra

Naive: $O(V^2)$

Efficient with binary heap: $O(E + \log V)$,

V on solmujen lukumäärä ja E kaarien lukumäärä

Distance vector (Bellman-Ford)

$O(E \cdot V)$



3) Hierarkkinen reititys

Reitityksen skaalautuvuus?

Isossa verkossa runsaasti reitittämiä

- Kaikki eivät voi tuntea kaikkia muita
- Reititystaulut suuria, reittien laskeminen raskasta
- Reititystietojen vaihtaminen kuluttaa linjakapasiteettiä

Autonomiset järjestelmät AS
(Autonomous Systems)

Internet ~ verkkojen verkko

Intra-AS routing

Kukin verkko päättää itse sisäisestä reitityksestään

RIP, OSPF

Inter-AS routing

AS:t ilmoittelevat toisilleen, mihin muihin AS:iin niistä pääsee

BGP (Border Gateway Protocol)



Hierarkkinen reititys

Yhdyskäytävä (gateway router)

Sovittu, mikä reititin keskustelelee naapuriverkon
(-verkkojen) kanssa

ulkoatuleva/ ulosmenevä paketti reitittyy yhdyskäytävään
AS:n sisäinen reititys huolehtii paketin AS:n koneelle tai
AS:n läpi toiselle AS:lle

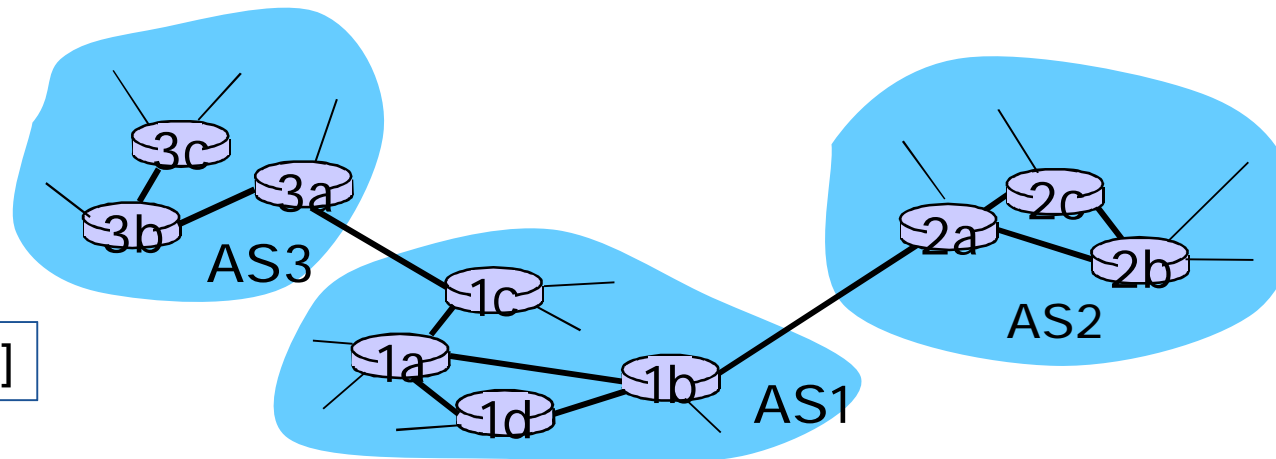


Fig 4.32 [KR12]



Kertauskysymyksiä

ks. kurssikirja s. 439-441

Keskeisimmät IP-otsakkeen tiedot?

Paketin paloittelu

Millainen on IP-osoite?

Reitittimen arkkitehtuuri?

Longest prefix match?

Aliverkon peite (mask)

NAT:n toiminta

Miten reititin saa reititystiedot?

Linkkitila-algoritmi, Dijkstran algoritmi

Etäisyysvektorialgoritmi, count-to-infinity-ongelma

