

Tietoliikenteen perusteet

Luento 8: Verkkokerros
IP-osoitteet,
reititysalgoritmit

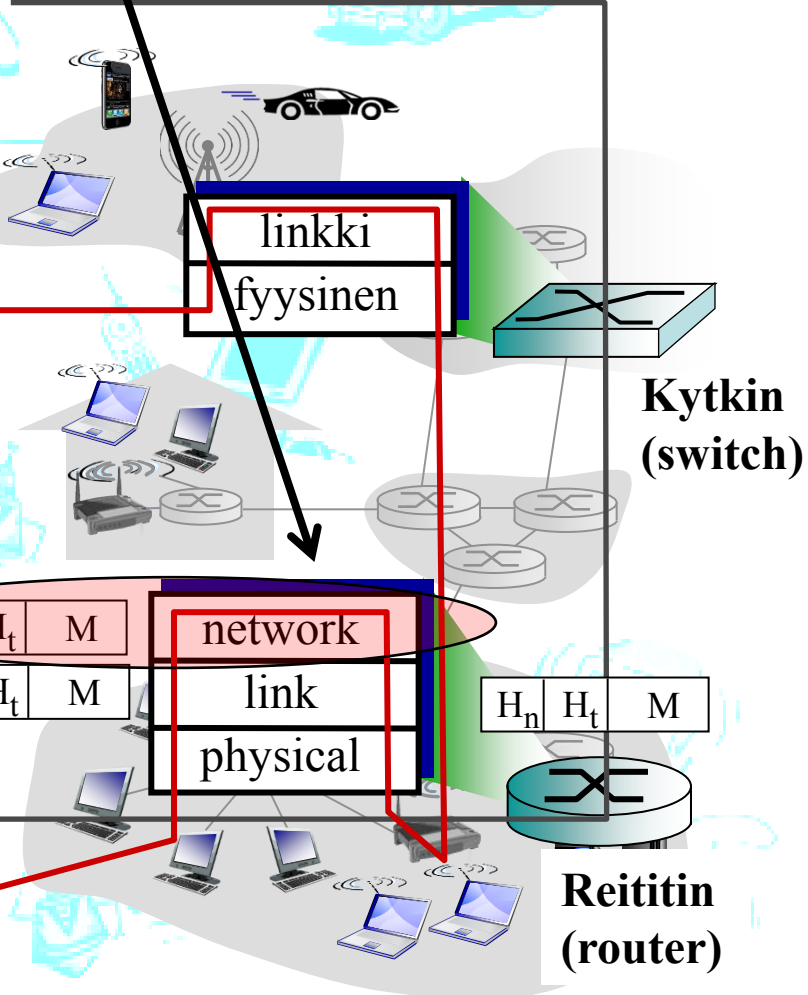
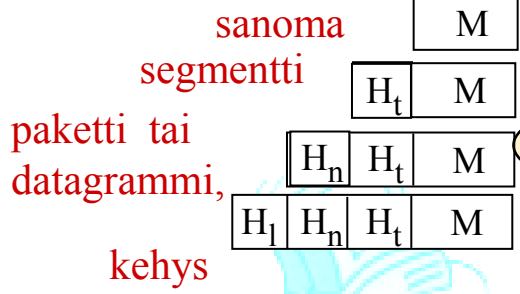
Syksy 2017, Timo Karvi

Kurose&Ross: Ch4

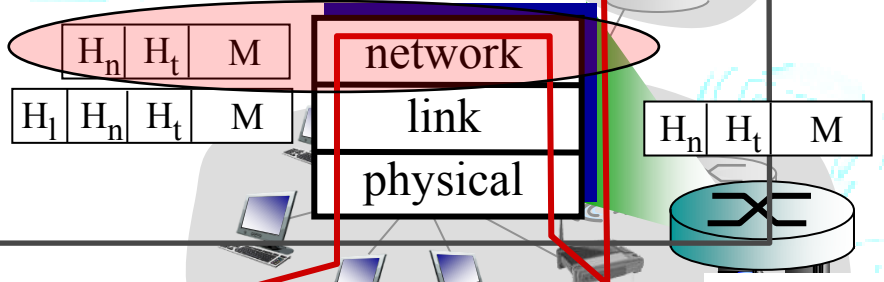
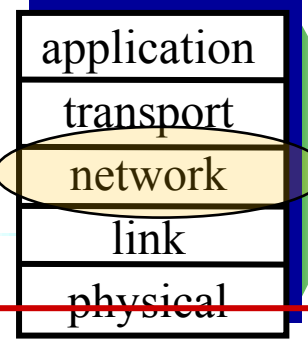
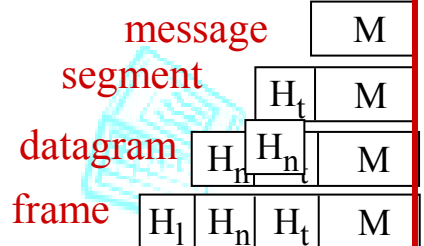
Pääasiallisesti kuvien
© J.F Kurose and K.W. Ross, All
Rights Reserved

Lähettäjä (source)

Luennon sisältöä



Vastaanottaja (destination)



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

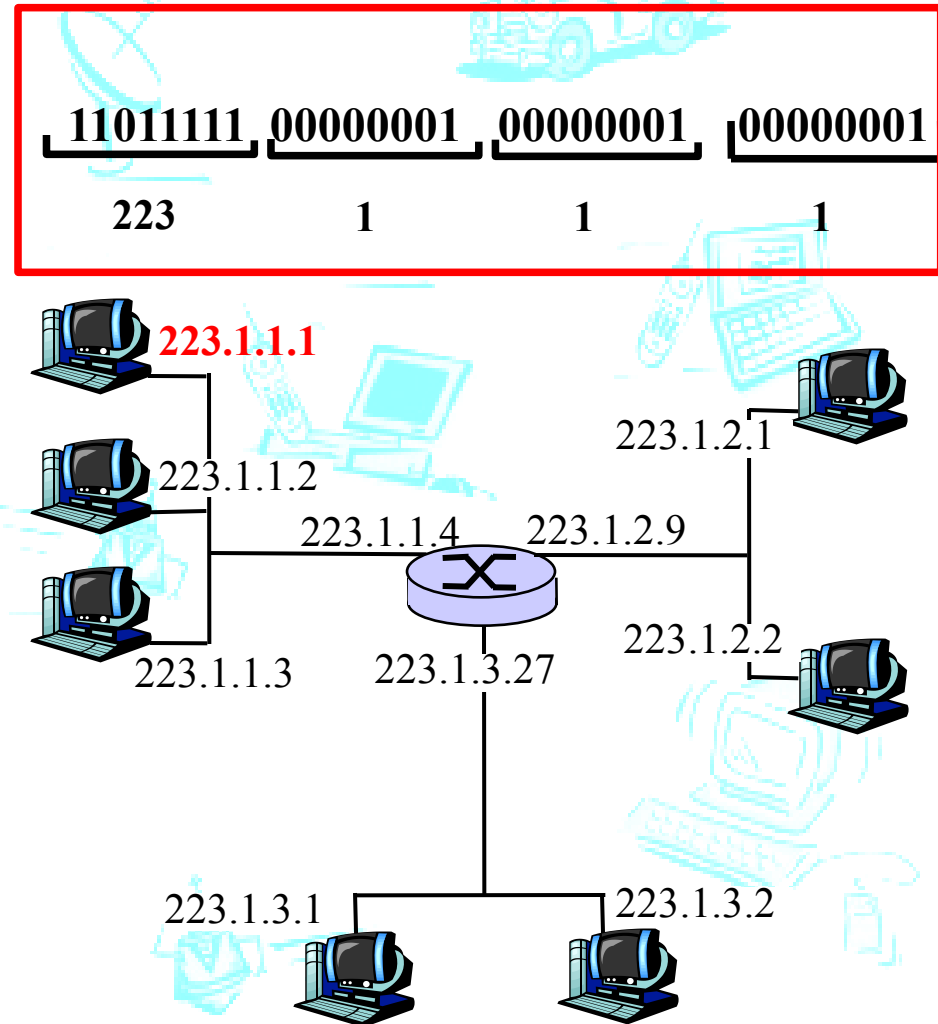


IP-OSOITTEET

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

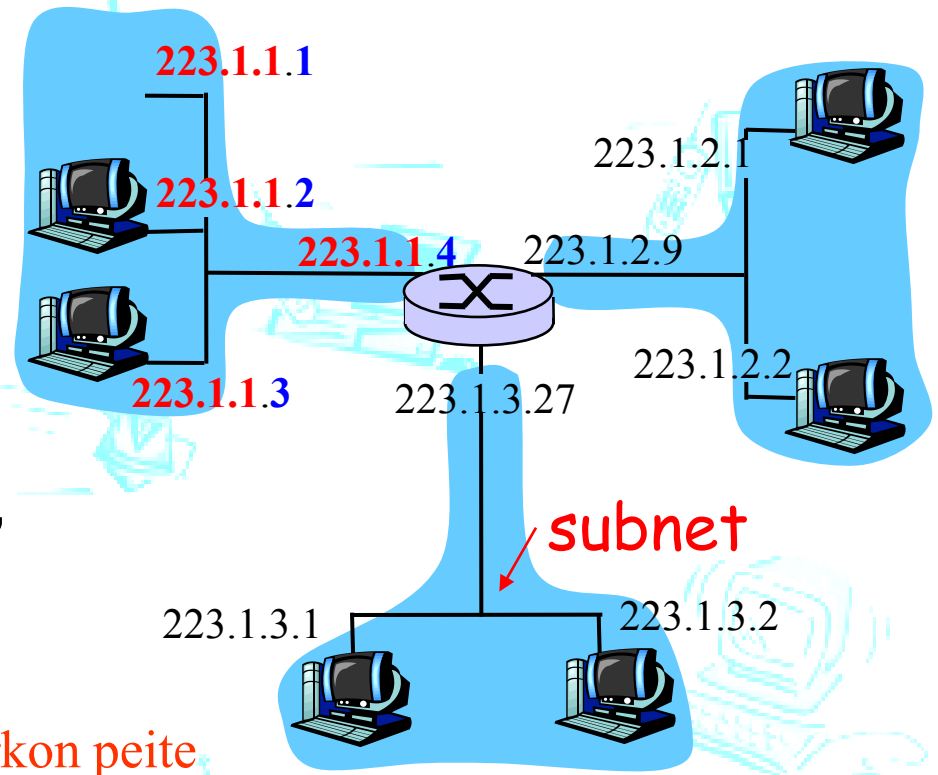


ICANN Internet Corporation for Assigned names and Numbers verkkonumerot palveluntarjoajille, joilta nämä edelleen aliverkoiksi

Aliverkot

- 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 koneosoite 8 bittiä

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)

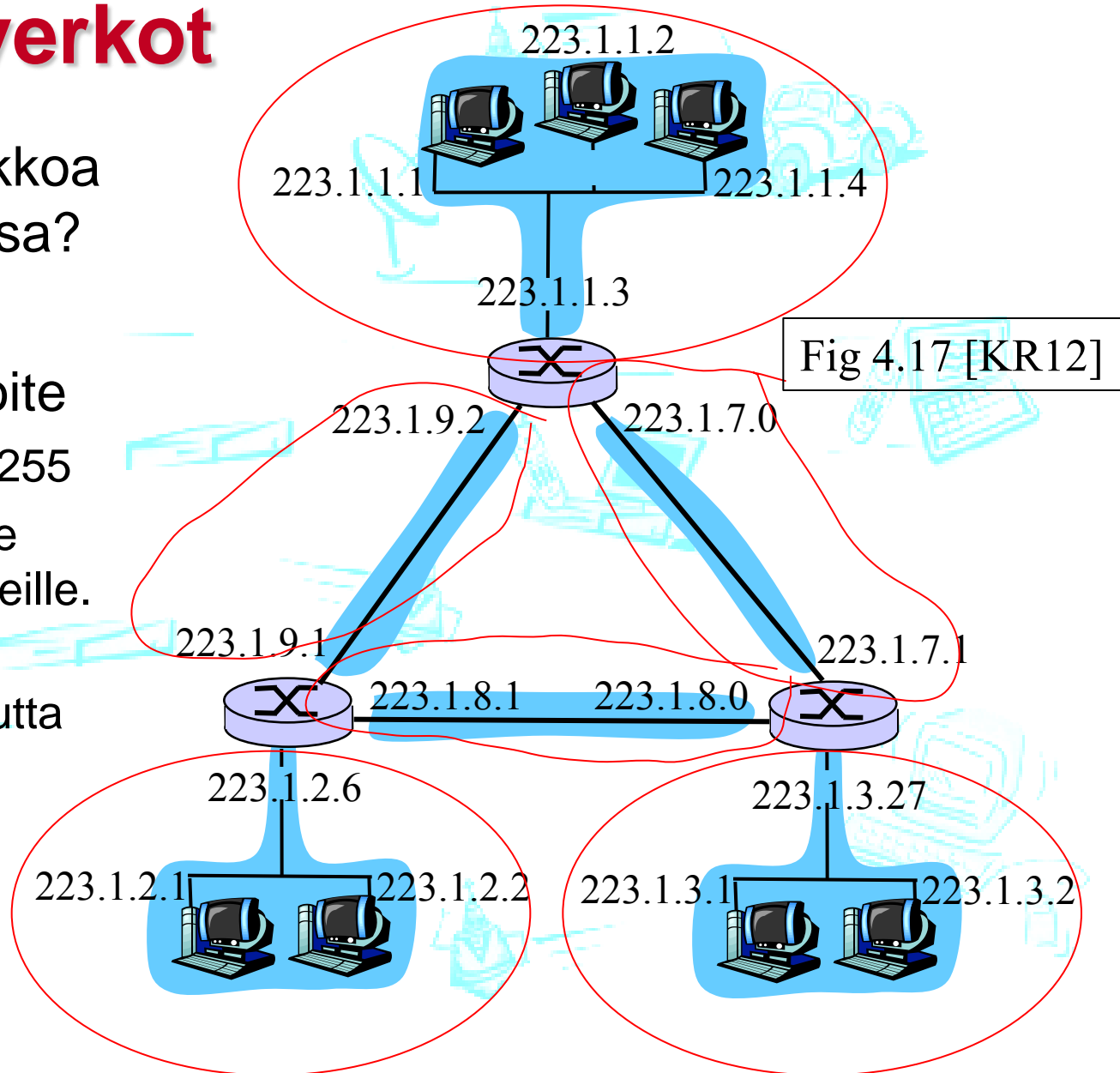


network consisting of 3 subnets

Fig 4.16 [KR12]

Aliverkot

- Montako aliverkkoa on tässä kuvassa?
- Yleislähetysosoite
 - 255.255.255.255
 - Paketti kaikille aliverkon koneille.
 - Mahdollisesti reitittimen kautta muillekin

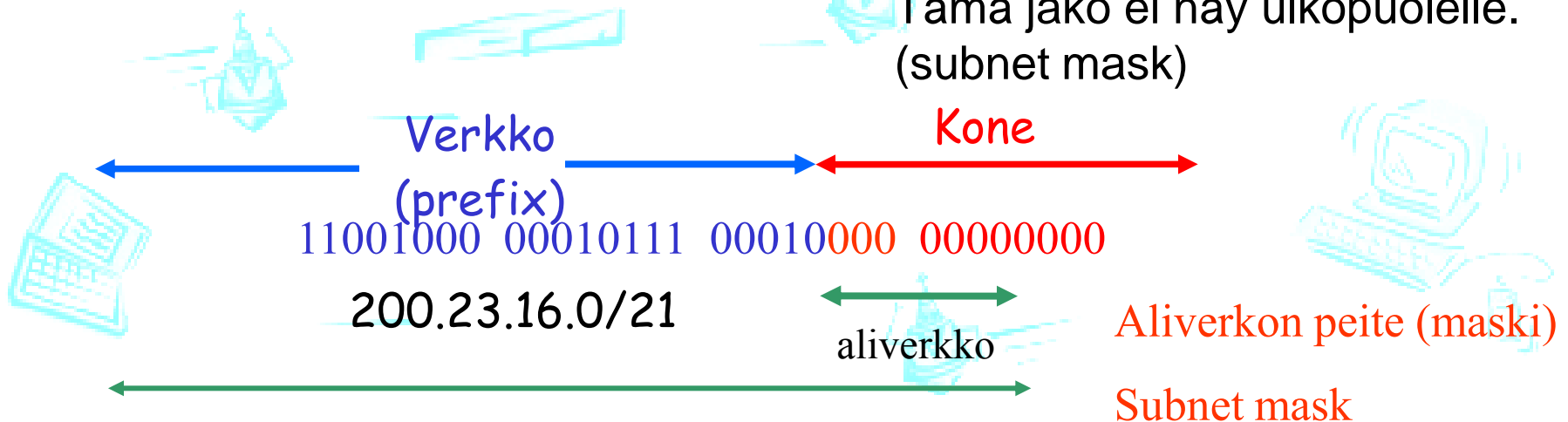


Aliverkkojen osoitteet:

- 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

$$2^{**12} = 4096 = 8 * 512$$

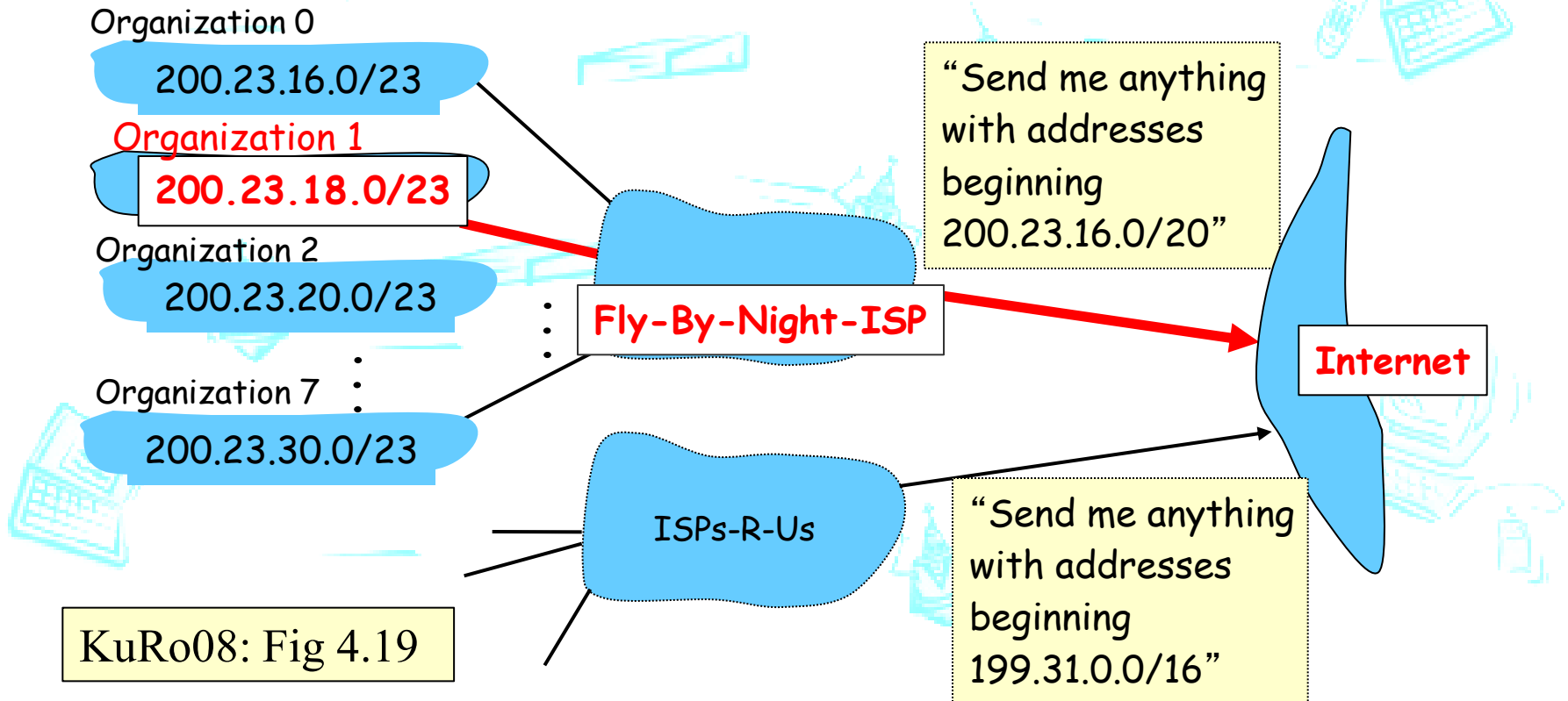


| | | |
|----------------|--|----------------|
| ISP's block | <u>11001000</u> <u>00010111</u> <u>00010000</u> 00000000 | 200.23.16.0/20 |
| Organization 0 | <u>11001000</u> <u>00010111</u> <u>00010000</u> 00000000 | 200.23.16.0/23 |
| Organization 1 | <u>11001000</u> <u>00010111</u> <u>00010010</u> 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

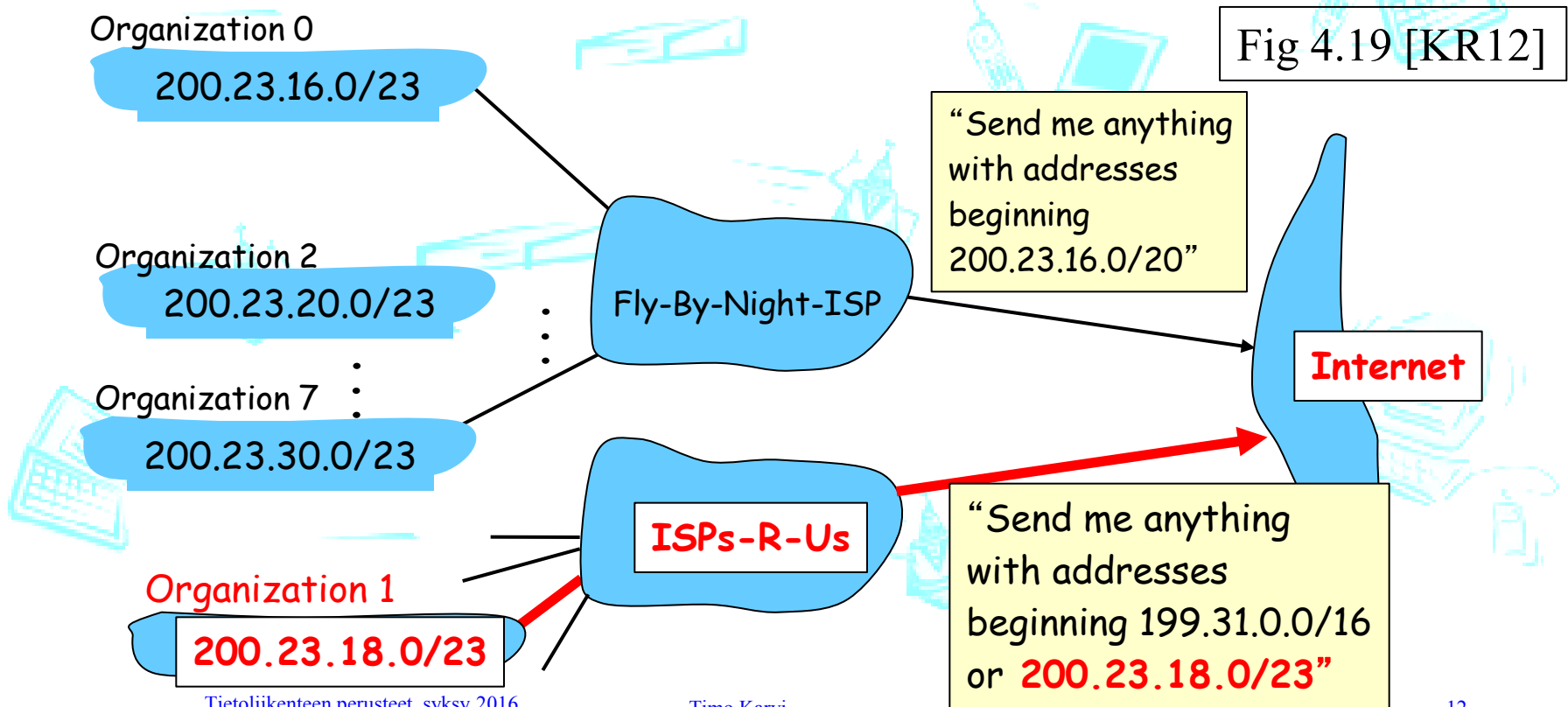
Fig 4.18 [KR12]

- CIDR luo reititystä helpottavan hierarkian
 - Aggregointi (yhdistäminen): yhteinen alkuosa => samaan suuntaan



Jos palveluntarjoaja (ISP) vaihtuu?

- IP-osoitteet voi säilyttää
- Uudelta ISP:ltä tarkempi reititysohje
 - **Pisin sopiva alkuosa määrää reitityksen (longest prefix match)**



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)
 - “plug-and-play”

DHCP: Osoitekysely

Fig 4.21 [KR12]

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

DHCP ACK

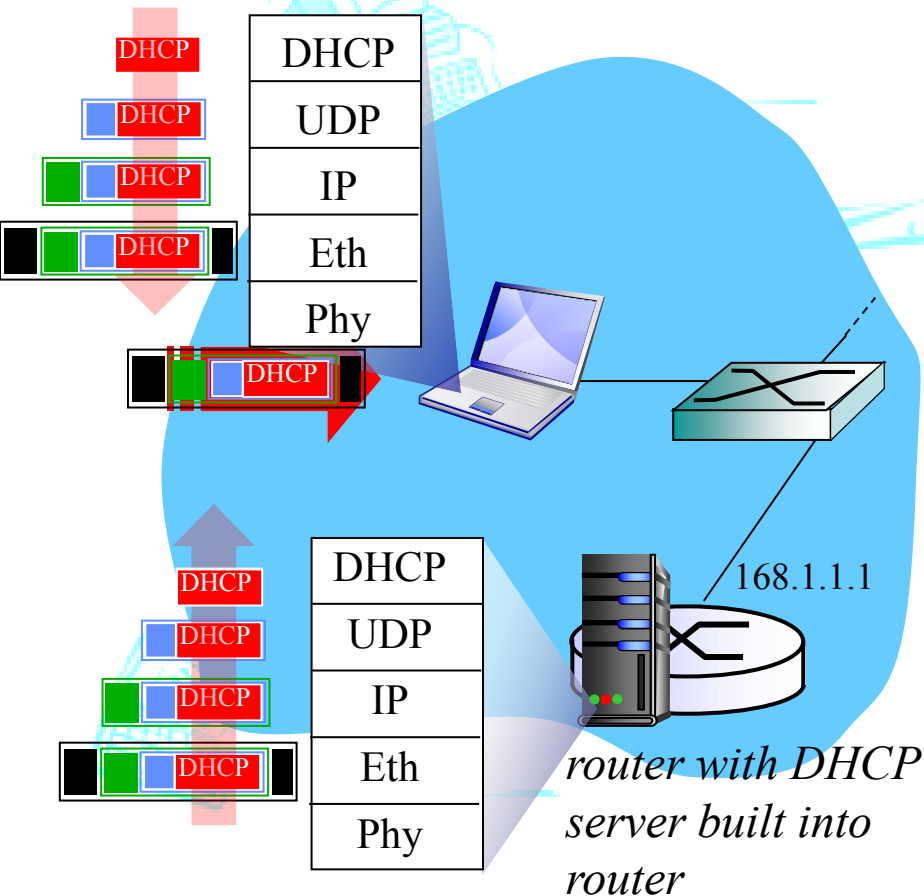
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

DHCP:

Muutakin tietoa kuin IP-osoite

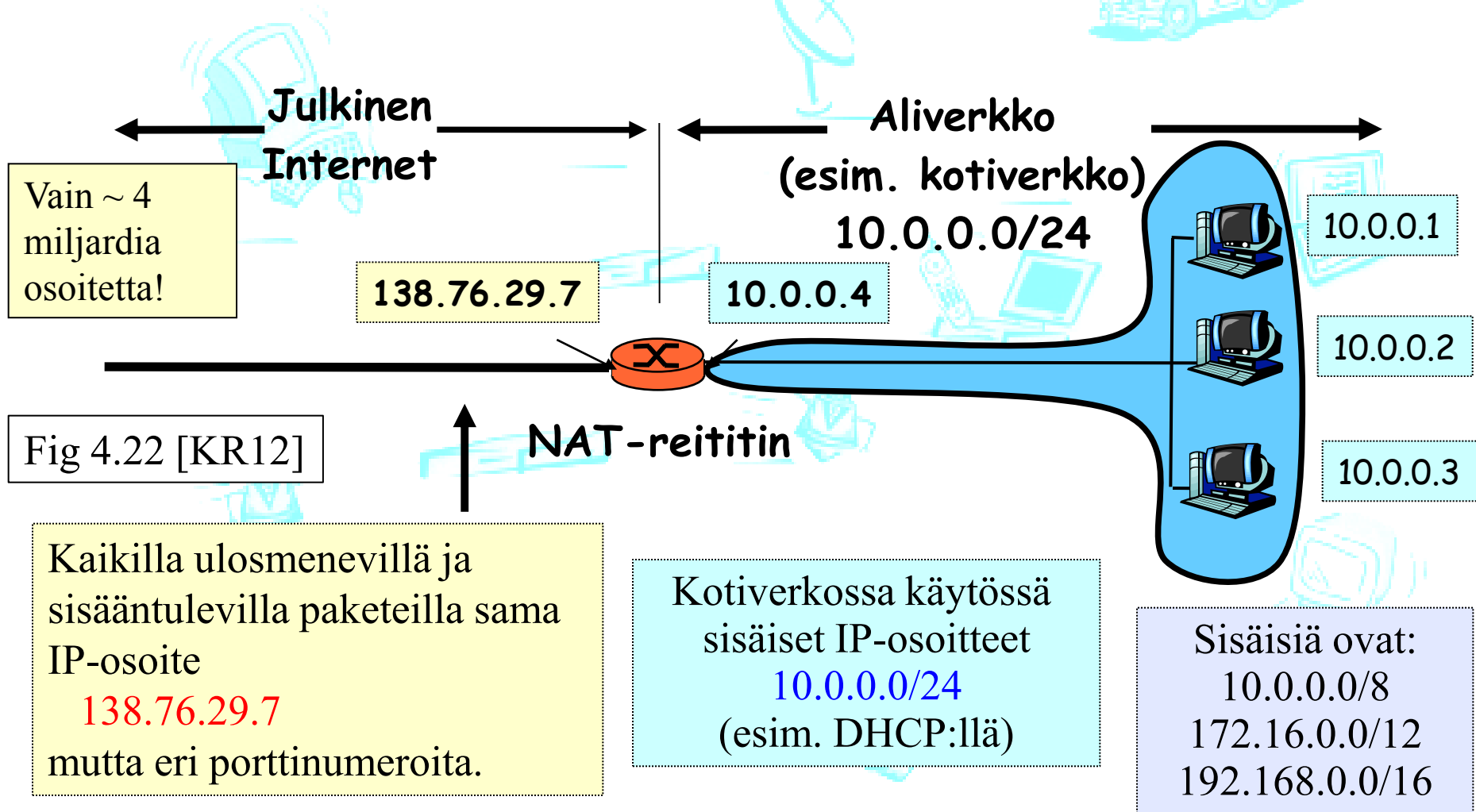
- DHCP antaa osoitteen lisäksi yleensä muutakin tärkeää tietoa:
 - Reitittimen osoite
 - Yhteyspiste ulos tästä aliverkosta
 - Nimipalvelimen (DNS) nimi ja IP-osoite
 - Täältä voi tiedustella muiden osoitteita
 - Aliverkon peite (subnet mask)
 - Kertoo mitkä osoitteen bitit ovat verkon osia ja mitkä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

Sanomanvälitys - Esimerkkinä DHCP request



- Kannettavan pyyntö: DHCP request
 - sovelluskerros: DHCP request,
 - kuljetuskerros paketoit UDP-segmentiksi,
 - verkkokerros paketoit IP-datagrammiksi,
 - linkkikerros paketoit 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)

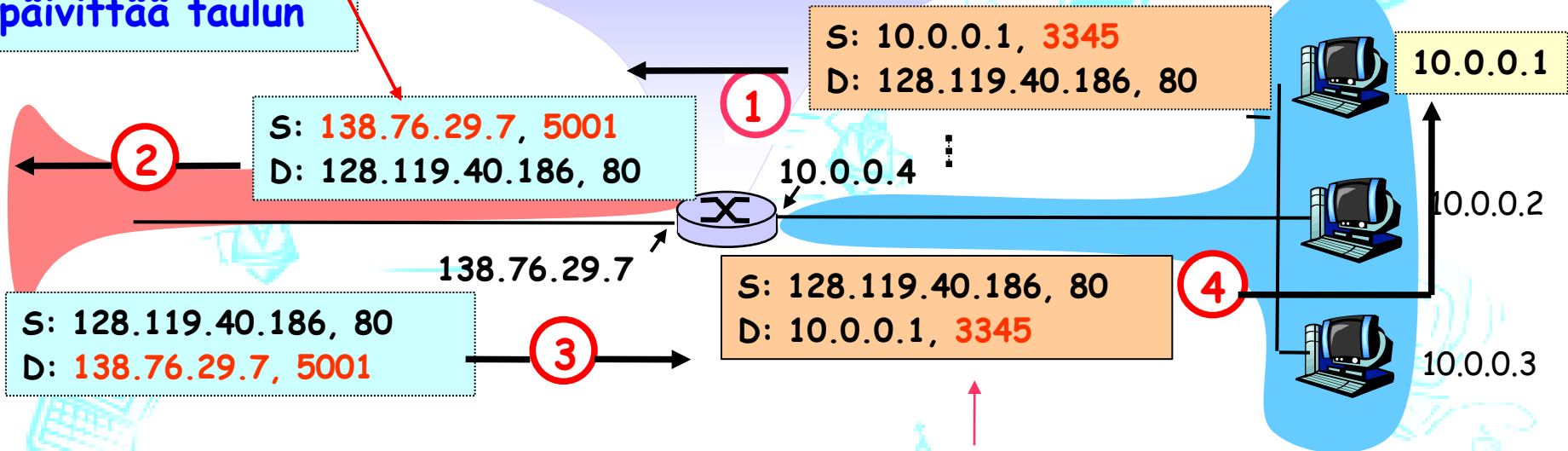
- Ulosmenevät paketit
 - Korvaa lähdekoneen IP-osoite ja porttinumero NAT-koneen IP-osoitteella ja NAT-koneen valitseamalla porttinumerolla
 - Päivitä NAT-muunnostaulu
- Sisääntulevat paketit
 - NAT-koneelle NAT:n antamaan porttiin
 - Korvaa NAT:n muunnostaulun avulla paketissa oleva IP-osoite ja portti
 - Välitä paketti perille
- NAT-muunnostaulu
 - (IP-osoite, portti) (NAT-koneen osoite, NAT:n portti)

NAT: Osoitteenmuunnos esimerkillä

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

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

1: kone 10.0.0.1 lähettää paketin 128.119.40.186, 80



3: Vastaus NAT-laitteelle: 138.76.29.7, 5001

4: NAT vaihtaa kohdeosoitteeksi 10.0.0.1 ja portiksi 3345

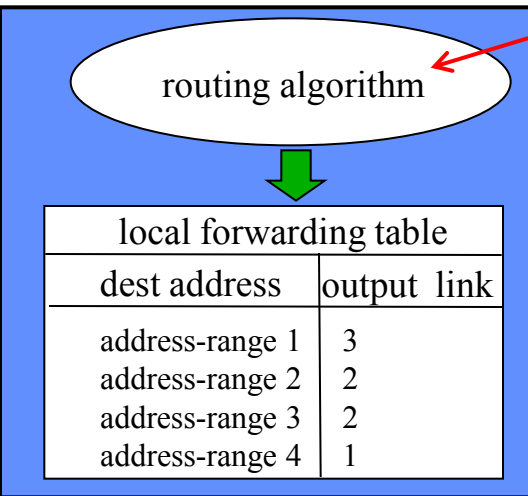
NAT: Kommentteja / kritiikkiä

- Hyödyt
 - Kotiverkko tarvitsee ISP:ltä vain yhden IP-osoitteen
 - Voi muutella 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)
 - Salauksessa ja todennuksessa voi tulla ongelmia (IPsec)
 - 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



REITITYSALGORITMIT

Reititysalgoritmi



IP destination address in arriving packet's header

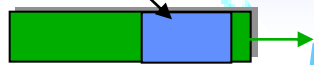
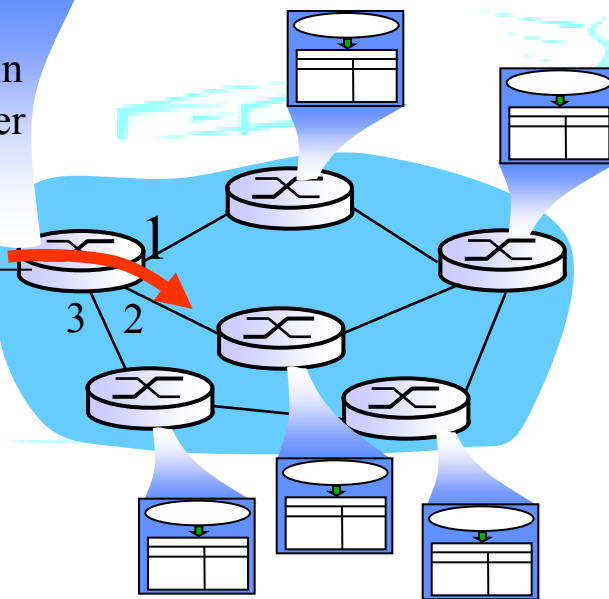


Fig 4.2 [KR12]



- 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')

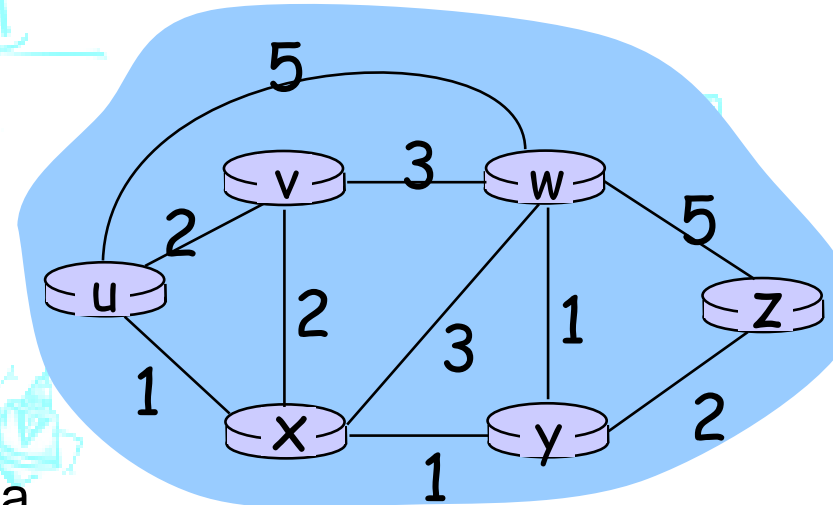
Reititystiedon kerääminen

- 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: lyhimmat 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 linkkikustannuksista naapureihin (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' :

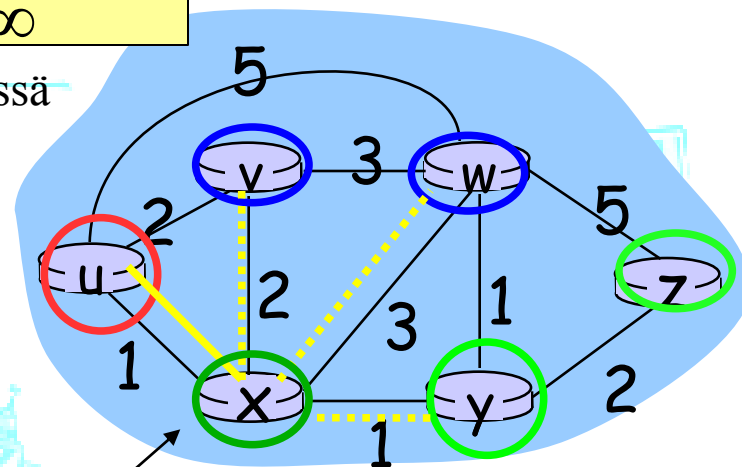
12 $D(k) = \min(D(k), D(b) + c(b,k))$

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

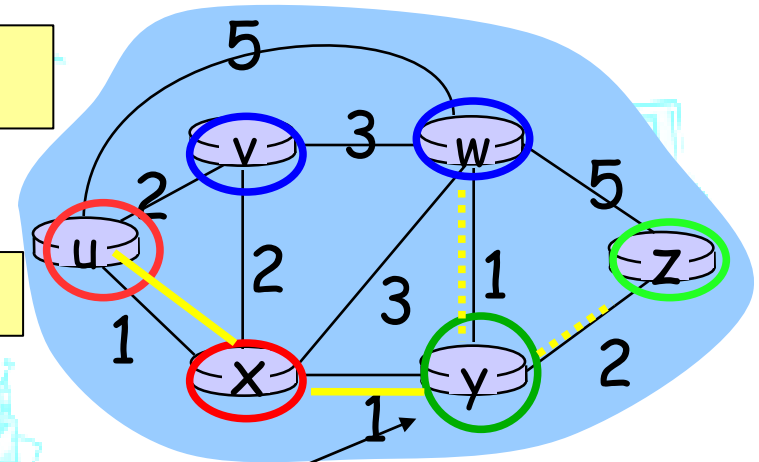
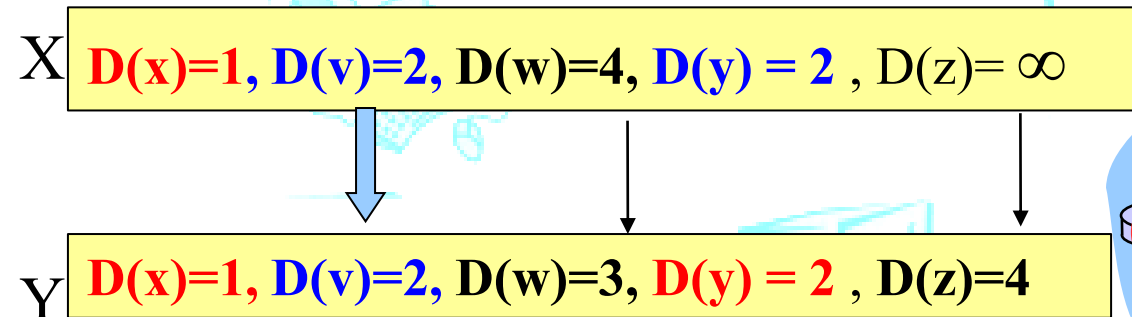


2. Aina valitaan käsittelemätön, jonka etäisyys u:sta on pienin

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

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 */

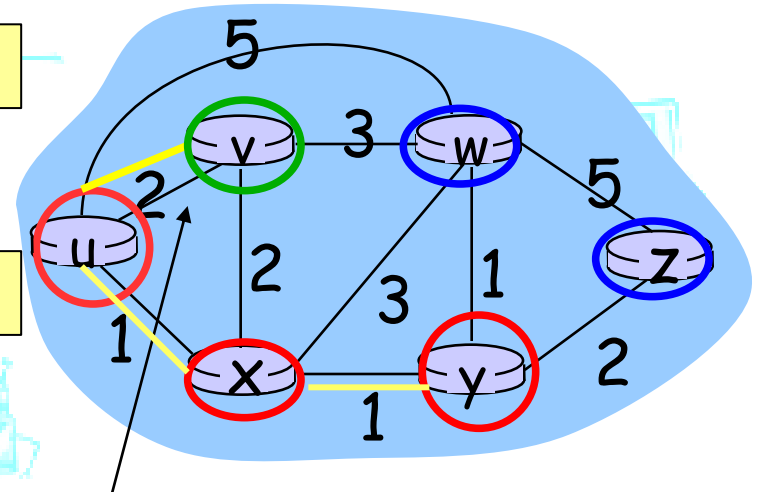
15 **until all nodes in N'**

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

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 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

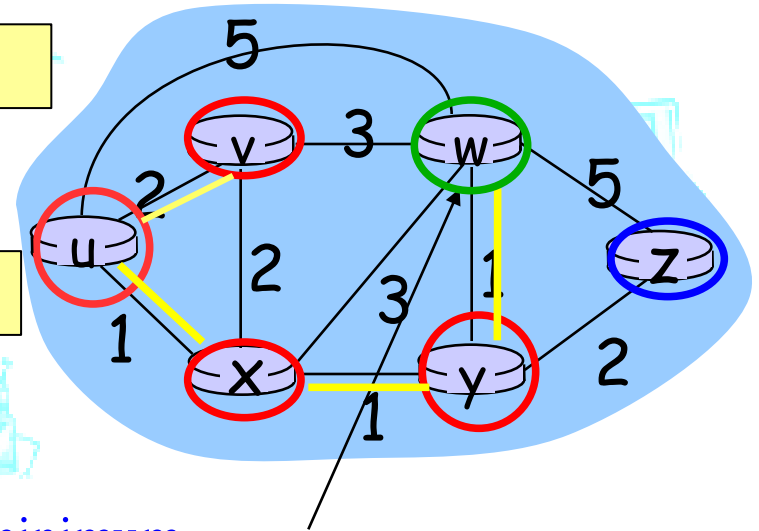
14 shortest path cost to w plus cost from w to v */

15 **until all nodes in N'**

Dijkstran algoritmi (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 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

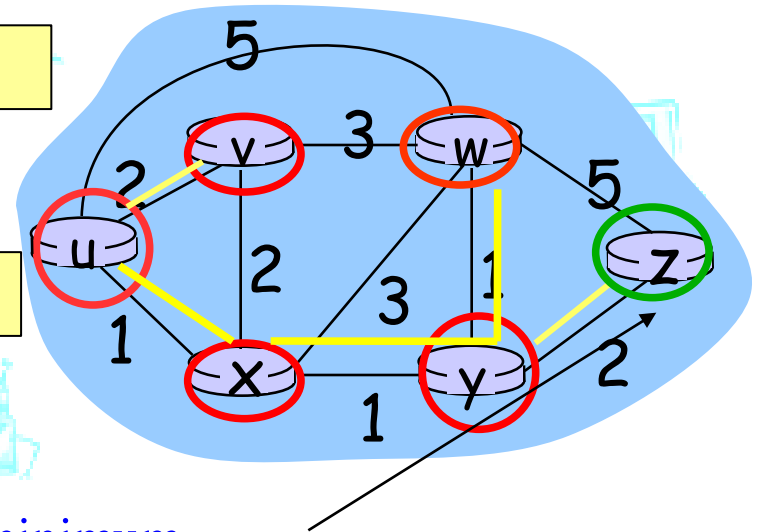
14 shortest path cost to w plus cost from w to v */

15 **until all nodes in N'**

Dijkstran algoritmi (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 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

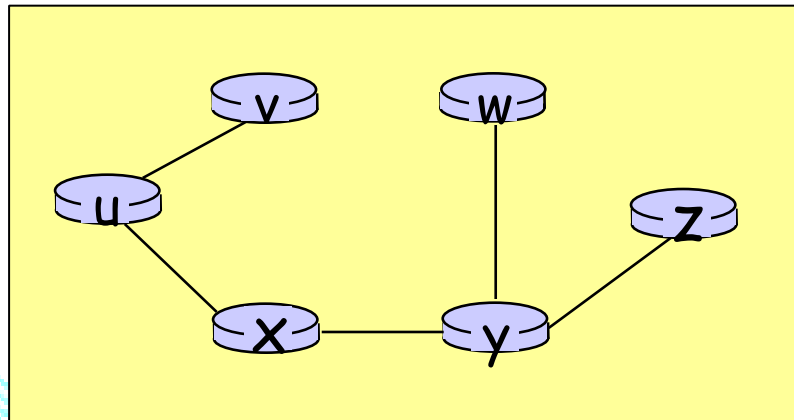
14 shortest path cost to w plus cost from w to v */

15 **until all nodes in N'**

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

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

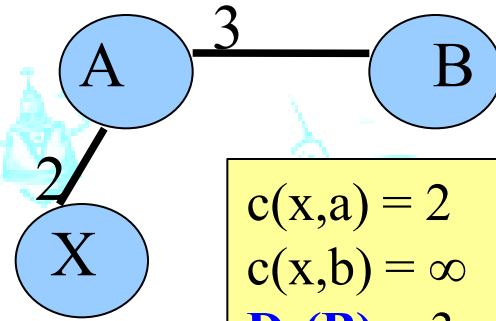
2) Etäisyysvektoreititys (distance vector)

- Arpanet-verkon alkuperäinen reititysalgoritmi
 - Käytössä useissa Internetin reititysprotokollissa, kuten RIP, BGP, Novell IPX, ISO IDRP
- Interaktiivinen, hajautettu ja asykroninen
- 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

Etäisyysvektoreireitys (jatkuu)

- Kullakin reitittimellä on oma 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 oman 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



$$c(x,a) = 2$$

$$c(x,b) = \infty$$

$$D_a(B) = 3$$

$$D_x(B) = 2 + 3 = 5$$

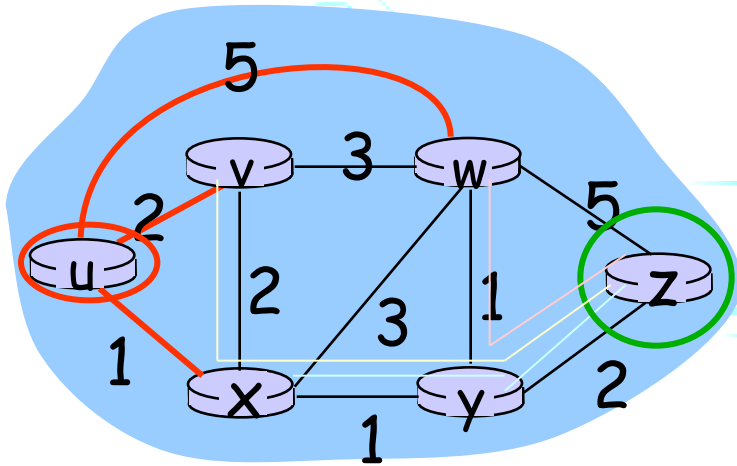
- Merkinnät

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



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

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

$$\begin{aligned} D_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\quad c(u,x) + d_x(z), \\ &\quad c(u,w) + d_w(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 1 + 3, \\ &\quad 5 + 3 \} = 4 \end{aligned}$$

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.

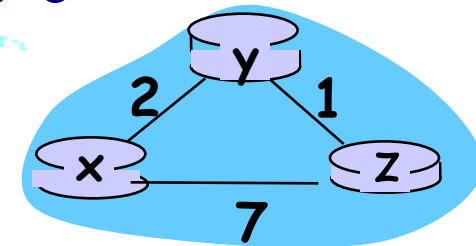
Alussa kukin solmu tuntee vain etäisyydet naapureihinsa itsensä kautta:

| | | 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 |

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

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

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

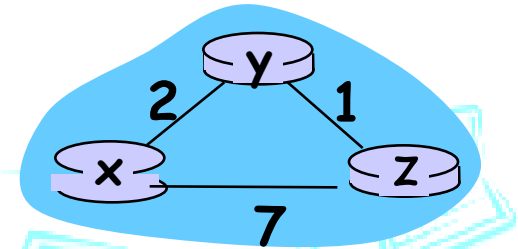
$$= \min\{2+1, 7+0\} = 3$$

Esimerkki 2 jatkuu:

Samalla tavalla toimivat solmut y ja z:

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

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



Solmut lähettävät taas tietonsa toisilleen ja laskevat uudet lyhimmät reitit.

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

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

| | | cost to | | | |
|------|----|---------|---|---|---|
| | | x | y | z | |
| from | Z: | x | 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 |

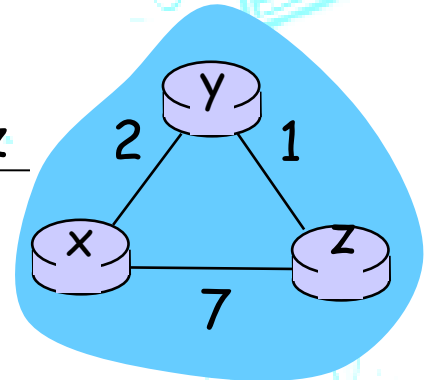
| | | cost to | | |
|------|---|---------|---|---|
| | | x | y | z |
| from | x | 0 | 2 | 3 |
| | y | 2 | 0 | 1 |
| | z | 3 | 1 | 0 |

z:

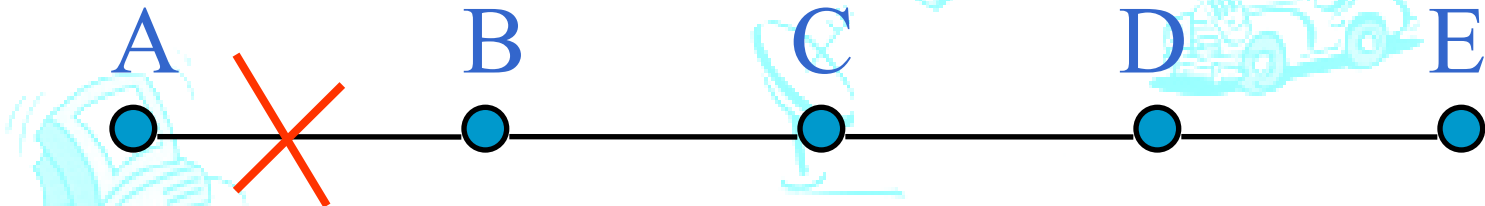
| | | cost to | | |
|------|---|----------|----------|----------|
| | | x | y | z |
| from | x | ∞ | ∞ | ∞ |
| | y | ∞ | ∞ | ∞ |
| | z | 7 | 1 | 0 |

| | | 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

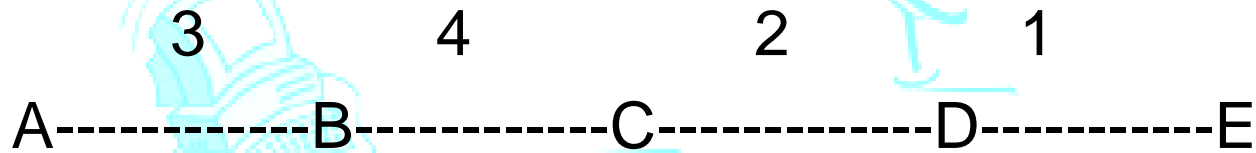
on poikki
ja sitten
linkki AB
toimii taas

Tieto etenee joka
vaihdossa
yhden linkin yli

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 |

Hyvä uutinen etenee nopeasti



D(B,A) D(C,A) D(D,A) D(E,A)

3 7 9 10

1 (pos. muutos)

1 5 9 10

1 5 7 10

1 5 7 8

Eli hyvä tieto leviää joka kierroksella kauemmaksi.

Huono uutinen leviää hitaasti

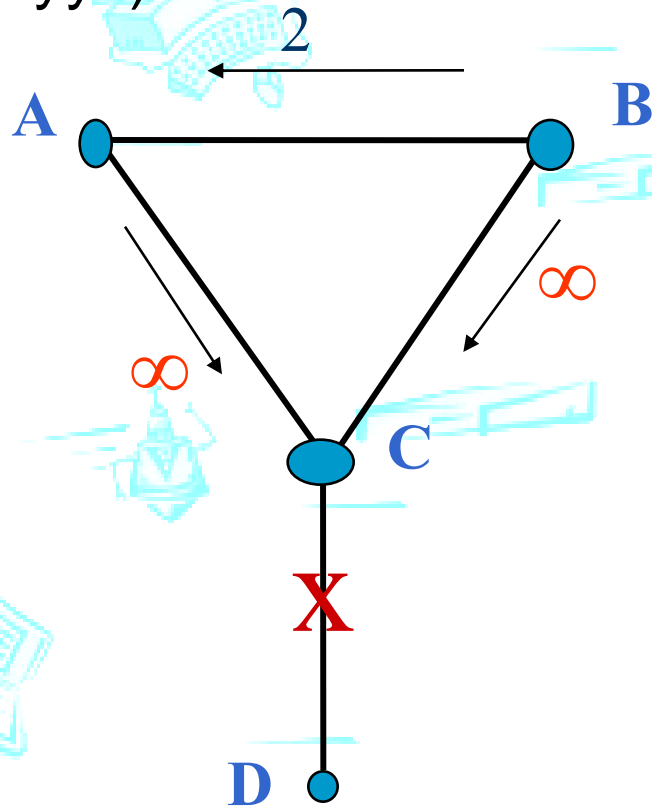
- Tarkastellaan kolmen solmun x, y, z verkkoa, jossa $C(x,y)=4$, $C(y,z)=1$, $C(x,z)=50$. Solmut ovat naapureita keskenään.
- Näillä arvoilla $D(y,x)=4$, $D(y,z)=1$ ja $D(z,x)=5$.
- Oletetaan nyt, että $C(x,y)$ huononee arvoon 60. y huomaa muutoksen ja päivittää oman $D(y,x)$ -arvon.
- Y käyttää tähän z :n aikaisemmin lähettämää arvoa 5 ja saa $D(y,x) = 6$. Jos nyt y lähettää paketin x :lle, paketti jää ikuiseseen silmukkaan x :n ja z :n välille.
- Seuraavalla kierroksella z päivittää $D(z,x)$:n arvoksi 7, mutta edelleen paketti jää silmukkaan.
- Tätä jatkuu monta kierrosta ennen kuin oikea arvo saapuu y :lle ja z :lle.

Poisoned reverse

- Ratkaisu ikuiseen silmukkaan: Jos y reitittää paketin x :n kautta, y ilmoittaa x :lle linkin $C(y,x)$ kustannuksen äärettömäksi.
- Muille ilmoitetaan oikea kustannus.
- Nyt x ei lähetä paketteja y :n kautta.
- Myös huono uutinen etenee nopeasti.

Ratkaisu ei toimi aina!

- (etäisyys)



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

C päättelee (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 reitittimiä

- Kaikki eivät voi tuntea kaikkia muita

- Reititystaulut suuria, reittien laskeminen raskasta

- Reititystietojen vaihtaminen kuluttaa linjakapasiteettia

- 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 keskustelee naapuriverkon (-verkkojen) kanssa
 - Ulkoatuleva / ulosmenevä paketti reitittyy yhdyskäytävään
 - AS:n sisäinen reititys huolehtii paketin AS:n koneelle tai AS:n läpi toiselle AS:lle

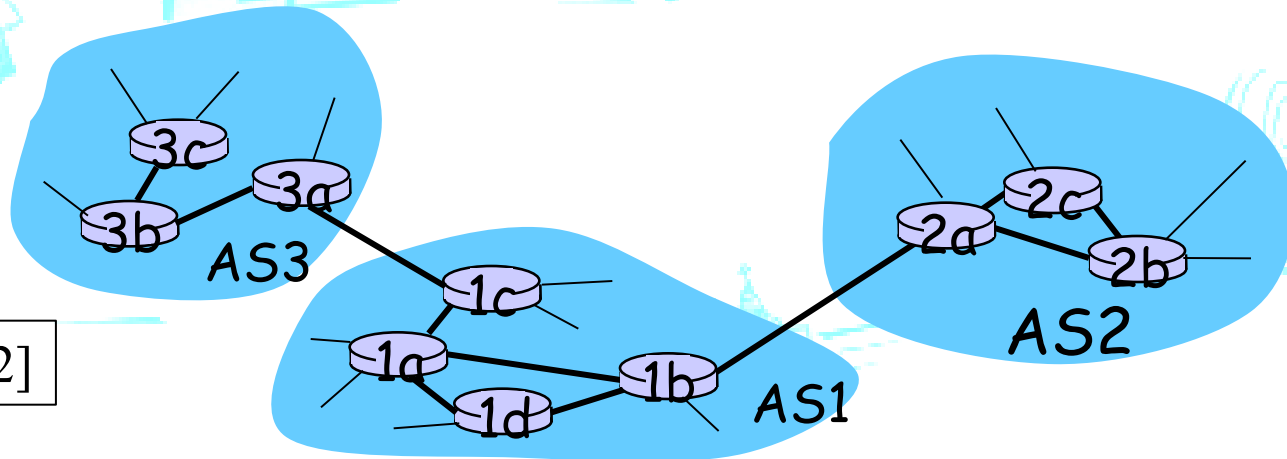


Fig 4.32 [KR12]

RIP

- Autonomisen verkon reititystandardi, alkuperäinen.
- Käyttää etäisyysvektorialgoritmia.
- Reitittää reitittimestä aliverkkoon.
- Kustannus on hyppyjen lkm, maksimi 15.
- Vektoreita vaihdetaan joka 30. sekunti.
- RIP on ohjelmoitu sovellustason prosessiksi, joka kuitenkin voi modifioida reititystauluja Unixin ytimessä.
- RFC 2453.

OSPF

- Toinen autonomisen verkon reititysstandardi.
- RFC 2328.
- Perustuu Dijkstran algoritmiin.
- Linkin kustannuksia ei määritelty: hyyt, kapasiteetti mahdollisia.
- Reititin yleislähettää reititystiedot verkkoon jokaisen muutoksen jälkeen tai vähintään puolen tunnin välein.
- Monia edistyksellisiä piirteitä verrattuna RIP:iin:

OSPF (2)

- i. Reititysanomat voidaan todentaa (vain luotetut reitittimet). Kevyt todennus: kaikilla sama salasana, paketissa salasana selväkielisenä. Vahvempi: tiiviste salaisen avaimen kera.
- ii. Reititin voi käyttää useita samanarvoisia polkuja.
- iii. Multicast routing.
- iv. Mahdollisuus jakaa autonominen verkko hierarkkisesti osiin.

BGP

- Autonomisten verkkojen välinen reititysprotokolla.
- Äärimmäisen monimutkainen.
- Siksi sivuutetaan tällä kurssilla.