Tietoliikenteen perusteet

Verkkokerros

Kurose, Ross: Ch 4.1- 4.42 ja 4.5

Sisältöä

- Verkkokerros
- Reititin
- IP-protokolla
- Reititysalgoritmit

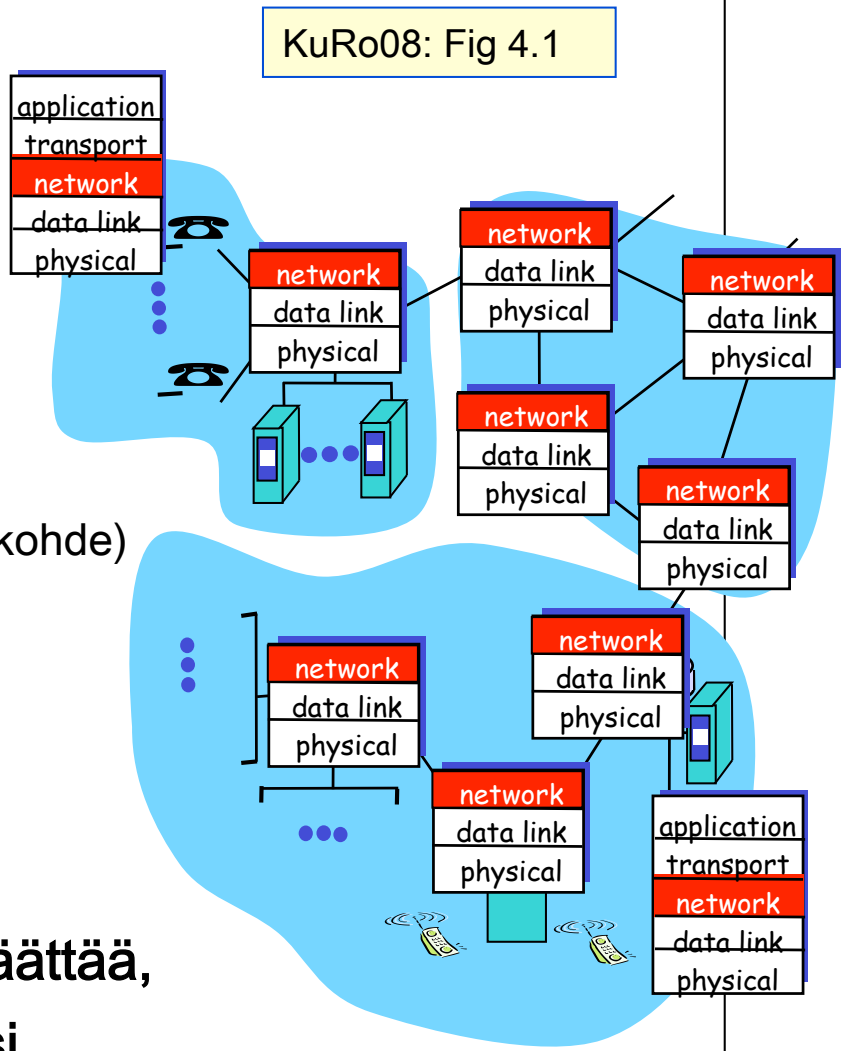
Oppimistavoitteet:
- Osata selittää, kuinka IP-paketteja välitetään verkossa
- Tietää, mitä tietoja sisältyy IP-pakettiin (ja miksi)
- Osata selittää reitittimien rakenne ja toiminta
- Osata kuvailla, kuinka reitittimet kokoavat reititystietonsa
= linkkitila- ja etäisyysvektorialgoritmien toimintaideat

Verkkokerros

Verkkokerroksen tehtävät

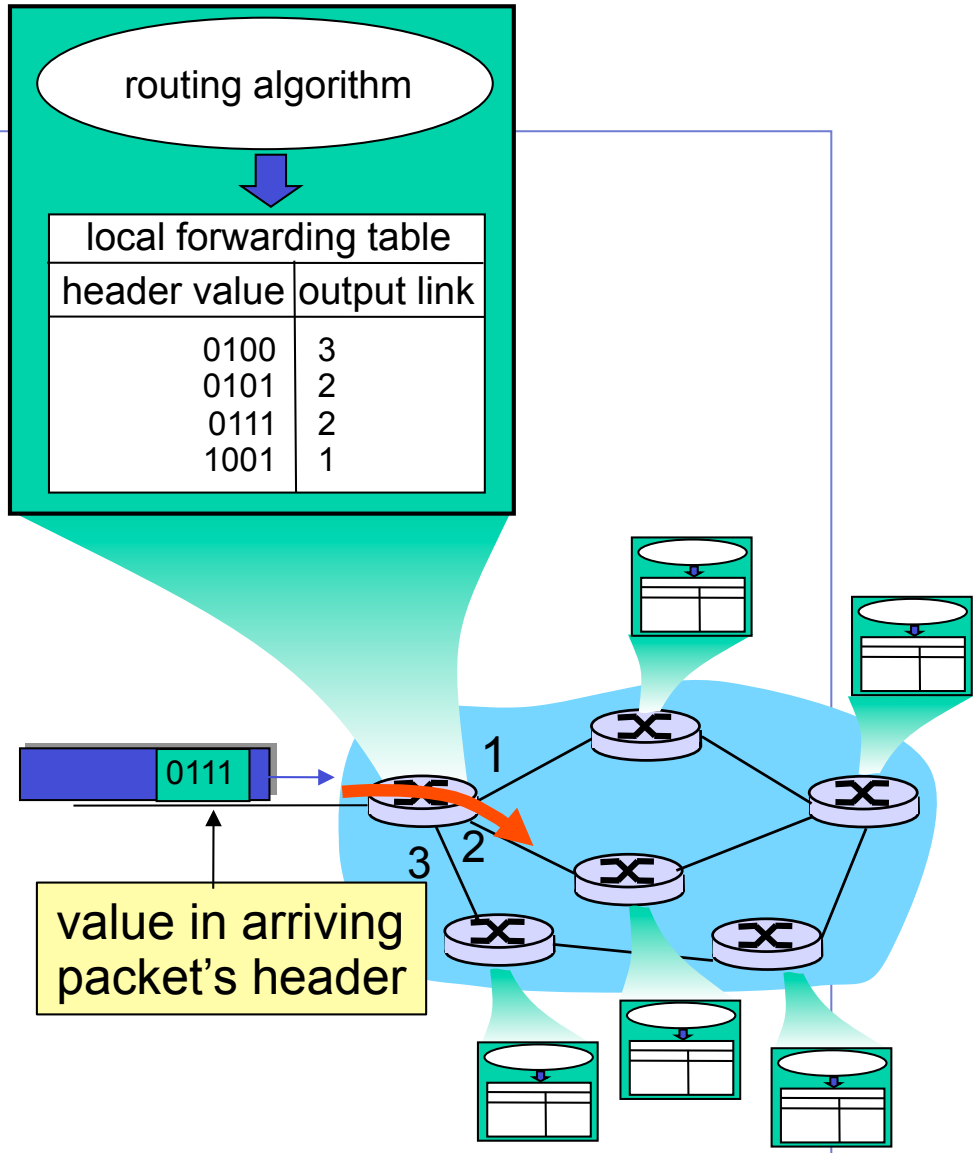
Verkkokerros

- Toimittaa kuljetuskerroksen segmentit vastaanottajalle
  - Lähetys
    - Luo segmenteistä verkkokerroksen IP-paketteja
    - Lisää otsaketietoja: mm. IP-osoitteet
  - Pakettien kulku verkossa
    - Isäntä (lähde) – reititin - ...- reititin – isäntä (kohde)
  - Vastaanotto
    - Poista otsake
    - Anna segmentti kuljetuskerrokselle
- Toimii etenkin reitityksessä
  - Reititin tutkii IP-paketin otsakkeen ja päättää, mihin linkkiin se lähetetään seuraavaksi



Reititin

- Ohjaa paketin reitittimen sisääntulolinkistä johonkin ulosmenolinkkiin (forwarding)
  - Katsoo reititystaulusta minne
- Kertoo muille reitittimille reitittämiseen liittyviä tietoja (routing)
  - Reittien selvittäminen
  - Reititystaulun ylläpito
  - Oma protokolla tätä varten (reititysalgoritmi, reititysprotokolla)
  - Käsin konfigurointi on hankalaa
- Piirikytkentäisessä verkossa myös yhteydenmuodostus ja purku



KuRo05: Fig 4.2

Miksi verkkokerros?

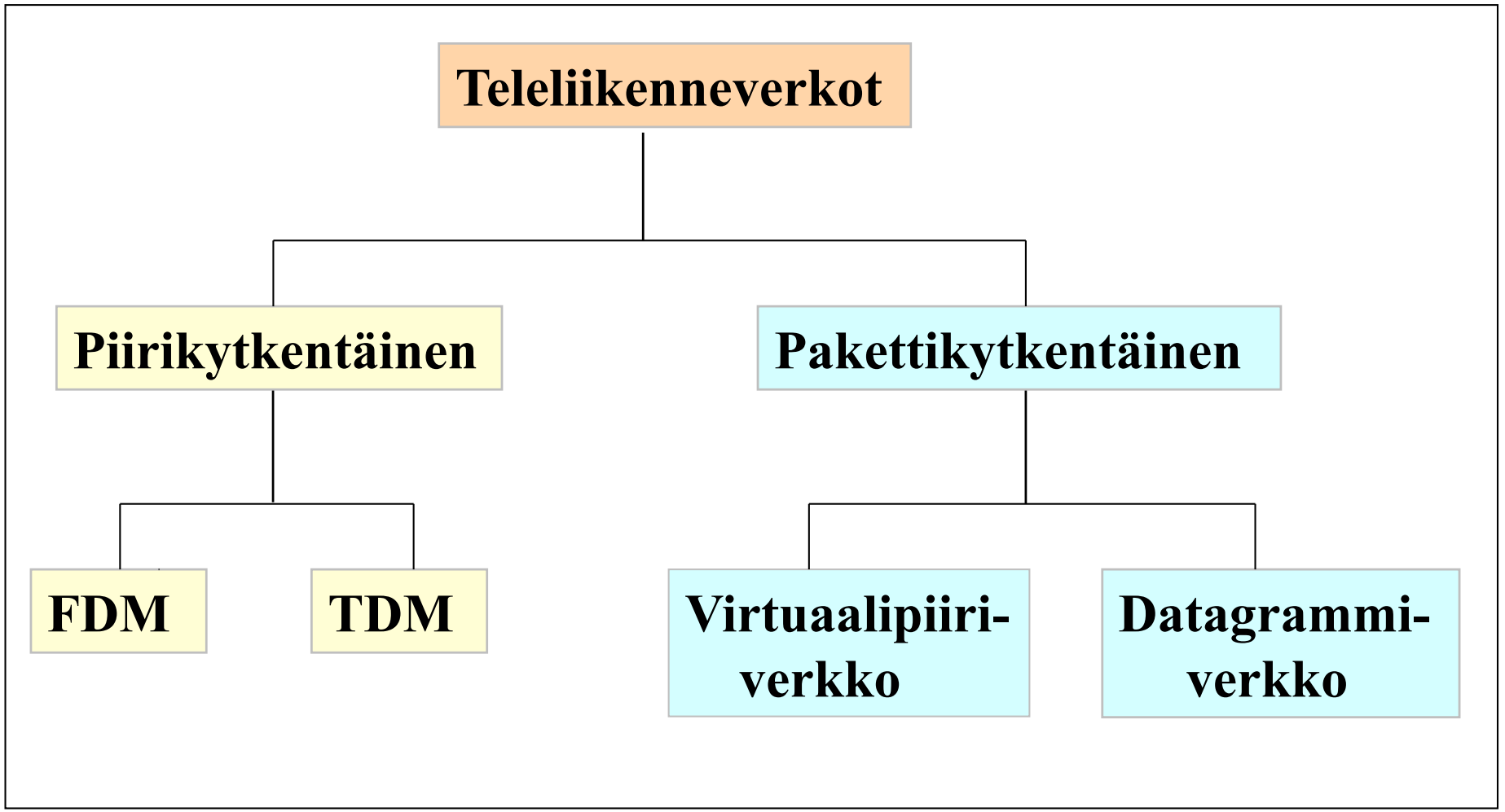
- Internet koostuu hyvin heterogeenisista verkoista
  - Verkot toteutettu eri teknologialla
  - Verkoilla kehyksen (frame) maksimikoko erilainen
  - Palvelu: yhteydellinen / yhteydetön,
  - Erilaisia osoittamistapoja: yksitasoinen/hierarkkinen
  - Monilähetys / yleislähetys
  - Toiminnot: virheenkäsittely, vuonvalvonta, ruuhkanvalvonta, yhteyden laatu / laatutakuu (QoS), turvaus, laskutus, ..
- Verkkoprotokolla IP (Internet Protocol) on verkkokerroksen yhteinen kieli
  - Internetin isäntäkoneiden ja reitittimien kommunikointitapa
    - "Kullakin omat murteensa ja tapansa, mutta kaikki osaavat IP:tä."
- Yhteinen osoitustapa: IP-osoite
  - Yksikäsitteiset osoitteet

Verkon palvelun laatu

Network Archtecture	Service Model	Bandwidth Guarantee	No-loss Guarantee	Ordering	Timing	Congestion Indication
Internet	Best Effort	none	none	any order possible	not maintained	none
ATM	CBR	guaranteed constant	yes	in order	maintained	congestion will not occur
ATM	ABR	guaranteed minimum	none	in order	not maintained	congestion indication provided

KuRo08:Table 4.1

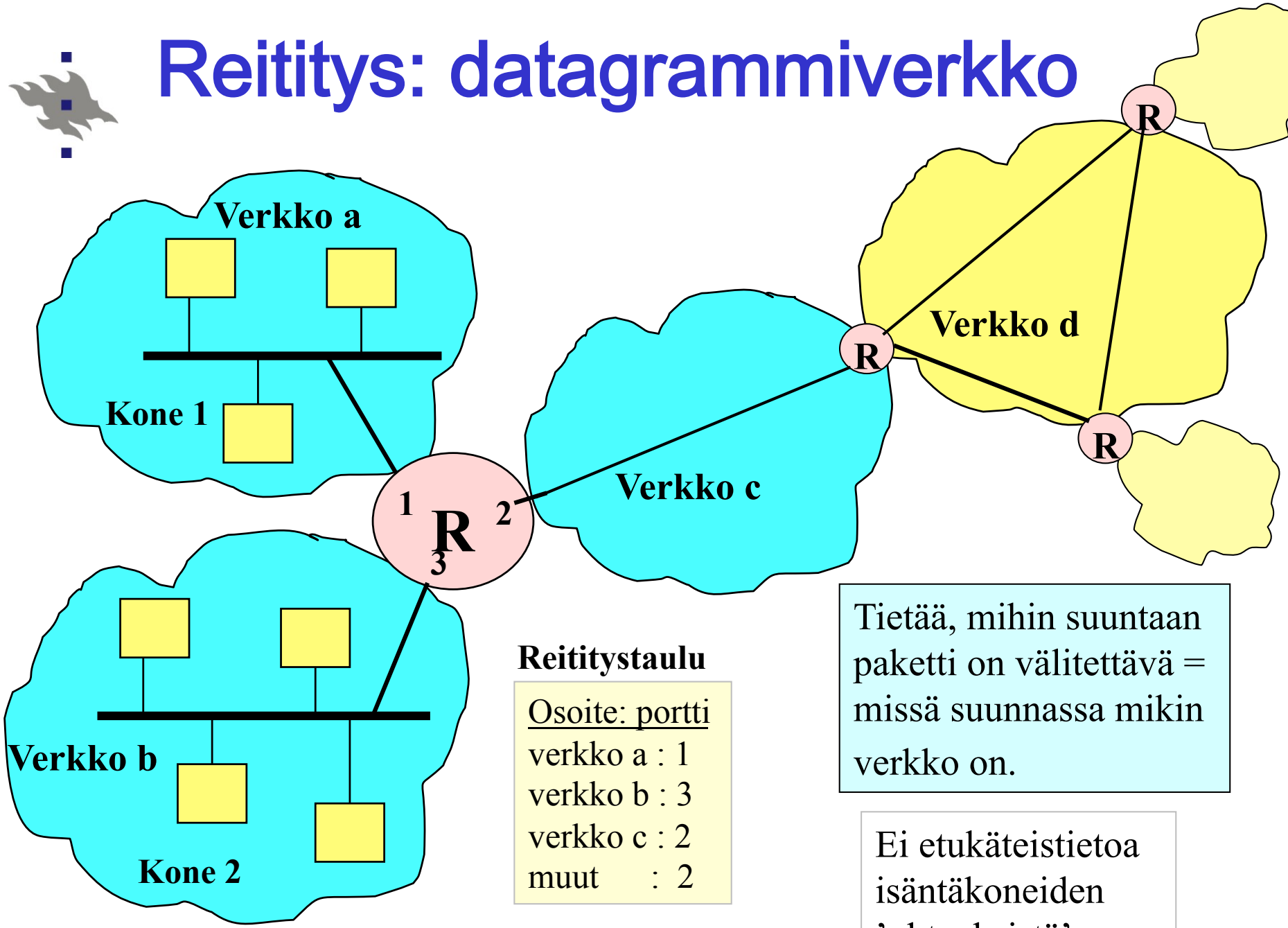
Verkkojen taksonomia



Pakettikytkentäinen verkko

- Joko **datagrammiverkkona** (= **Internet**)
  - Sanoman jokainen paketti reititetään erikseen
    - kohteen IP-osoitteen perusteella
  - “Tyhmä verkkokerros”: vain pakettien välitys koneelta koneelle
  - “Fiksut isäntäkoneet”: virheenvalvonta, vuonvalvonta, järjestys
- tai **virtuaalipiiriverkkona**
  - Sanoman jokainen paketti kulkee samaa reittiä pitkin
    - linkkiin liitetyn virtuaalipiirinumeron perusteella
  - Signallointiprotokolla: yhteydenmuodostus, ylläpito, purku
    - yhteyden tietoja reitittimessä (virtuaalipiirin muunnostaulukko)
    - mahd. myös kaistavarausta
  - Fiksu verkkokerros: vuonvalvonta, virhevalvonta, järjestys
  - tyhmät isäntäkoneet: vrt. Puhelin
  - ATM-verkot (Asynchronous Transfer Mode), X.25-verkot

Reititys: datagrammiverkko



Reititystaulu

Osoite: portti	
verkko a	: 1
verkko b	: 3
verkko c	: 2
muut	: 2

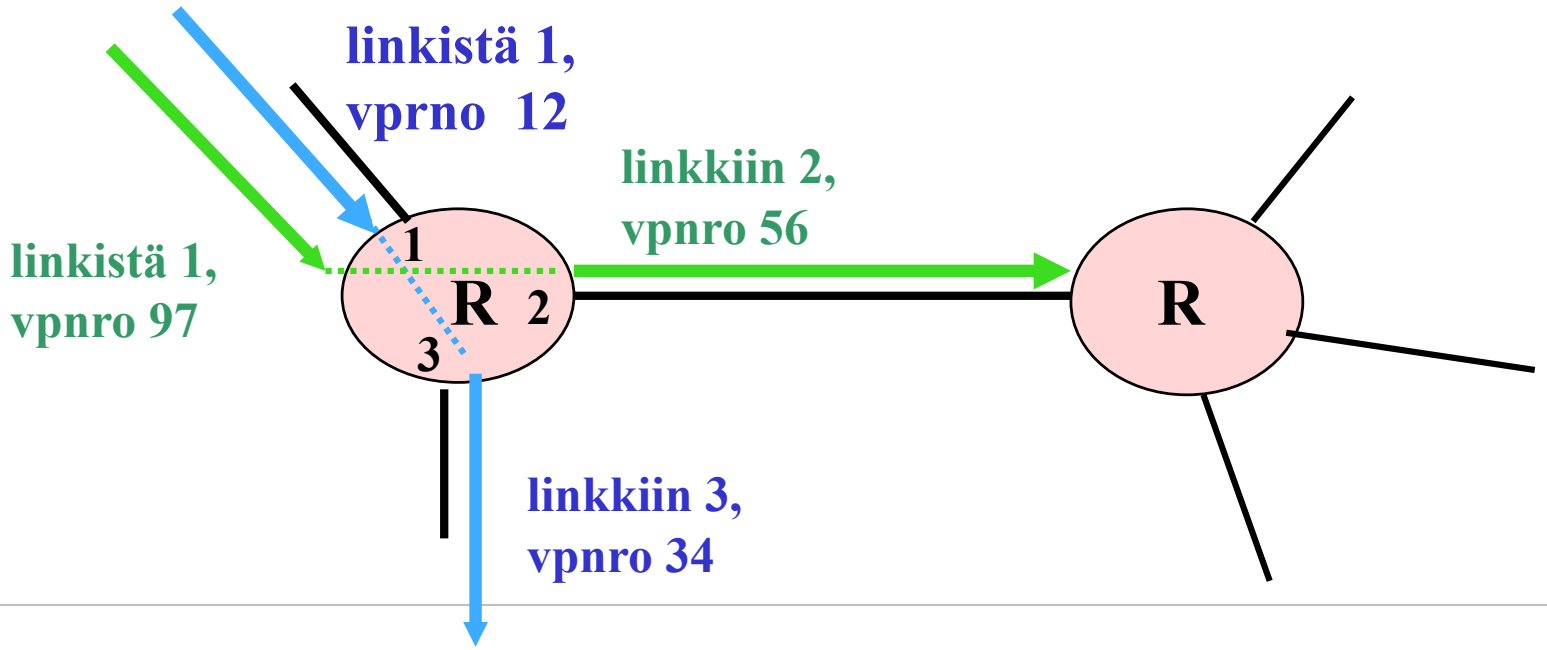
Tietää, mihin suuntaan paketti on välitettävä = missä suunnassa mikin verkko on.

Ei etukäteistietoa isäntäkoneiden ’yhteyksistä’

Reititys: Virtuaalipiiriverkko

- 1. paketti muodostaa reitin, muut paketit kulkevat samaa reittiä
  - otsakkeessa kohdeosoitteen lisäksi virtuaalipiirinumero vpnro
  - reititin ylläpitää tietoa piirinumeroista (’hajujälki’)

  Reititys = selvitä vpnro:a vastaava linkki, välitä paketti linkille



Virtuaalipiirin muunnostaulukko

Sisään: linkki	vpnro	Ulos: vpnro	linkki
1	12	34	3
1	97	56	2
2	42	101	3
2	10	78	1
3	12	65	2

Taulukkoa päivitettävä aina kun uusi yhteys on muodostettu tai vanha purettu!

- **Pakettivälitystä: Ylläpitää kyllä tilatietoja yhteydestä (=vpnro), mutta ei varaa resursseja etukäteen!**

Virtuaalipiirin muunnostaulukko

- **Virtuaaliyhteyden joka linkillä omat VP-numerot**
  - reititin antaa VP-numerot
- **Miksi ei käytetä koko yhteydellä samaa VP-numeroa?**
  - **Tarvittaisiin paljon enemmän numeroita!**

    Nyt riittää pienempi numeroavaruus =>

    tarvitaan pienempi kenttä numeroa varten

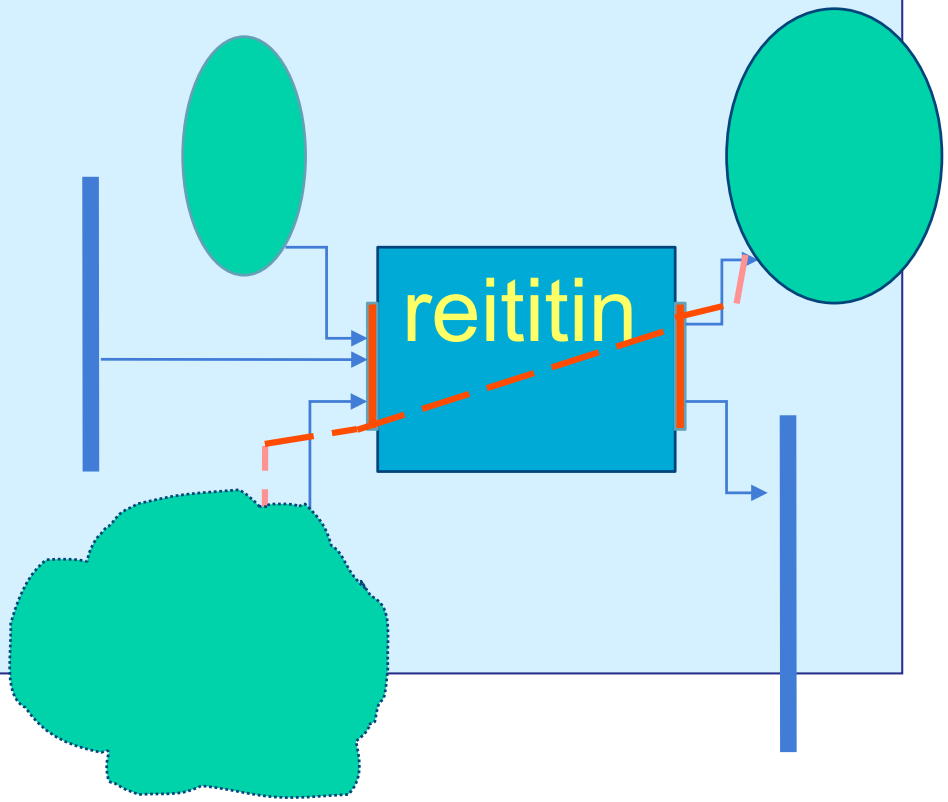
    0-255 => riittää 8 bittiä

    0-4095 => tarvitaan 12 bittiä
  - **Yhteisestä, koko verkon läpikäyvästä numeroinnista sopiminen on isossa verkossa lähes mahdoton tehtävä!**

Verkkokerros

Reititin

Ch. 4.3

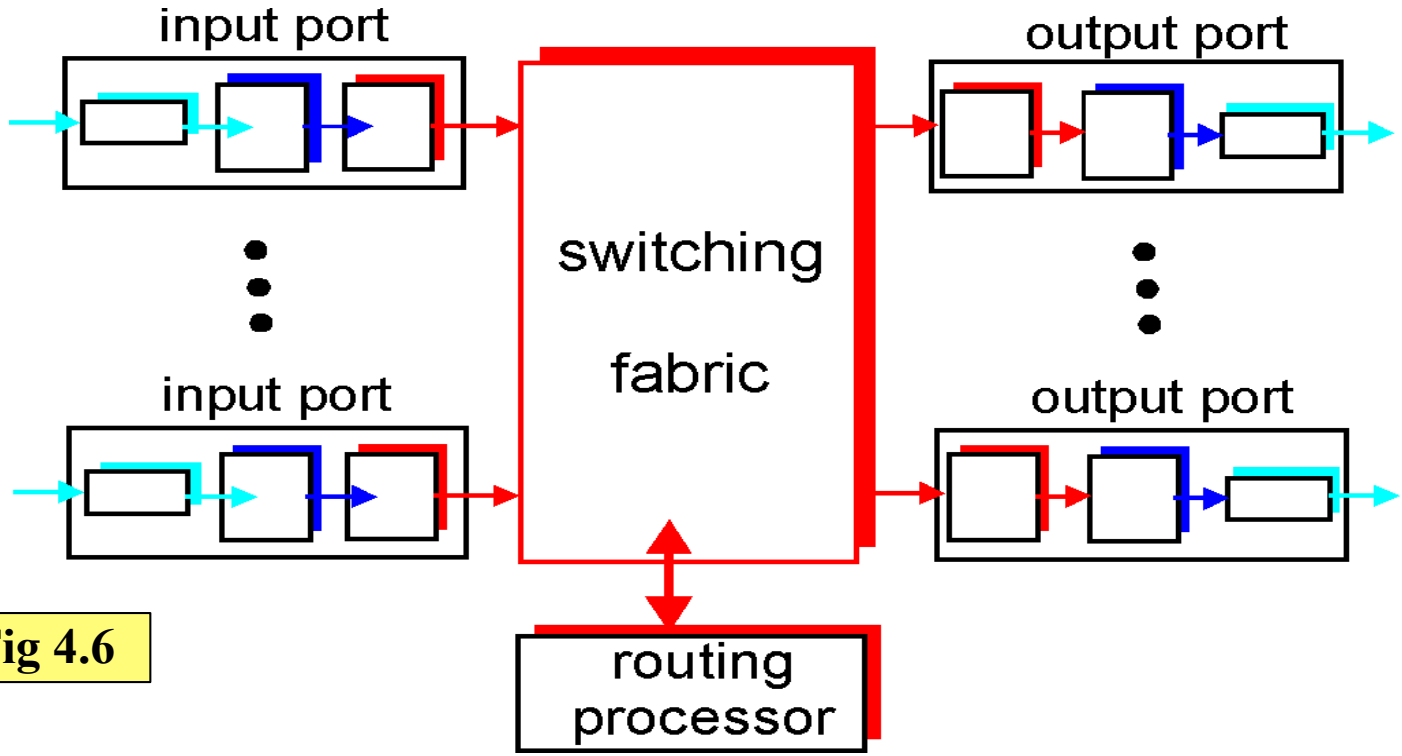


Verkkokerros ~ reitittäminen

- **Reititin** (router)
  - Osaa muunnokset siihen kytkettyjen teknologioiden välillä
  - Sisääntulolinkki ja ulosmenolinkki voivat ollat eri teknologiaa
  - Välittää verkkokerroksen otsakkeen perusteella (IP-osoite)
  - Laitteistotoimintona tai osin ohjelmallisesti
- **Kytkin** (switch)
  - Sekä sisääntulolinkki että ulosmenolinkki ovat samaa teknologiaa
  - Lähiverkon sisällä välitys linkkikerroksen otsakkeen perusteella
  - Poikkeuksetta aina laitetason toimintona

Reitittimen arkkitehtuuri

- Kaksi tehtävää
  - Välitä paketteja tulolinkeistä ulosmenolinkkeihin
  - Suorita reititysalgoritmia / -protokollaa
- Portti ~verkkokortti
  - Useita portteja niputettu yhteen linjakorteiksi (line card)

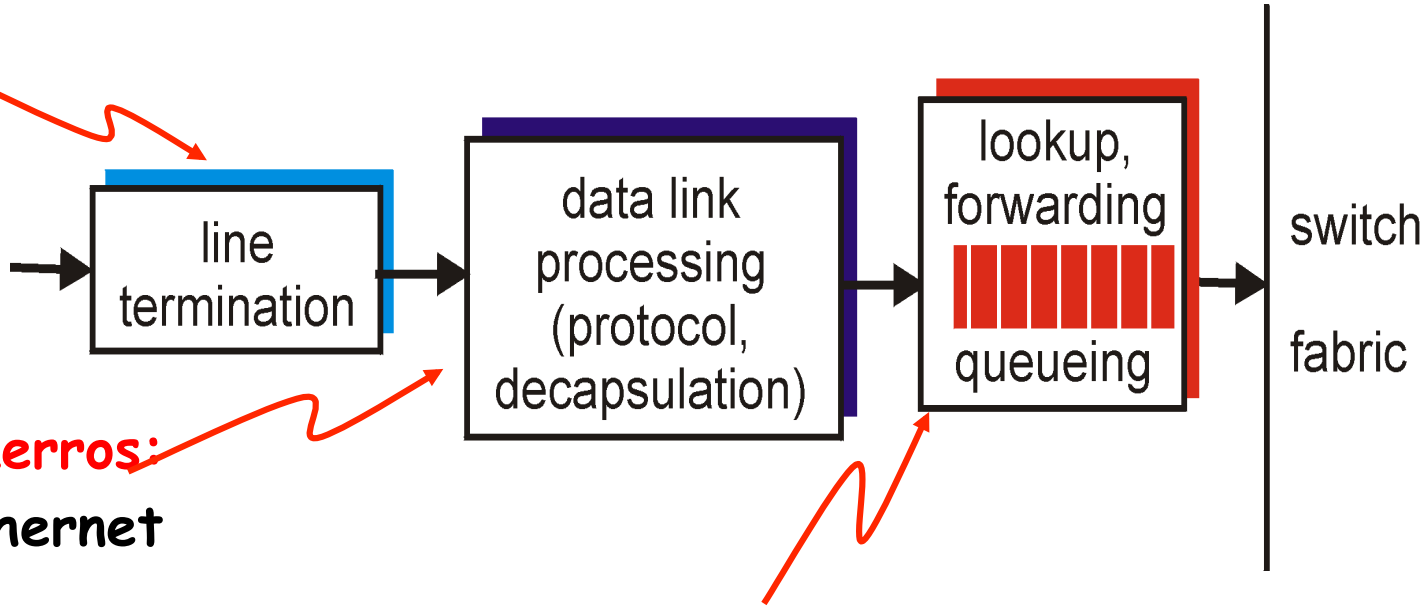


KuRo08:Fig 4.6

Sisääntuloportti (input port)

KuRo08: Fig 4.7

Fyysinen kerros
bittitason esitys



Tavoite: paketti ulos sisääntulon nopeudella

Jonotus: jos ulosmeno hitaampi kuin sisääntulo tai joku muu siirtää samaan ulostuloon

myös HOL (head-of-line blocking)

Jos linjanopeus 2.5 Gbps ja paketin koko 256 tavua => 1.2 miljoonaa pakettia sekunnissa!

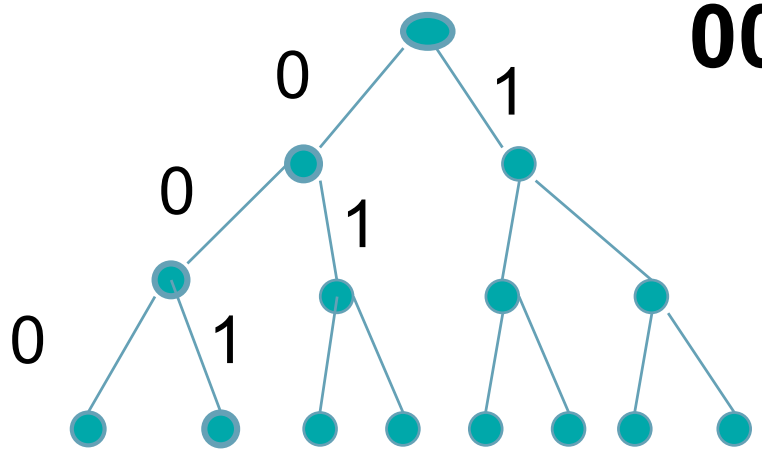
Osoitteen

1. bitti

2. bitti

3. bitti

jne



Kun n = 32 (IP-osoite), ei ole tarpeeksi nopea nykyisiin runkoreitittimiin!

- content addressable memory (CAM)
- välimuistin käyttö

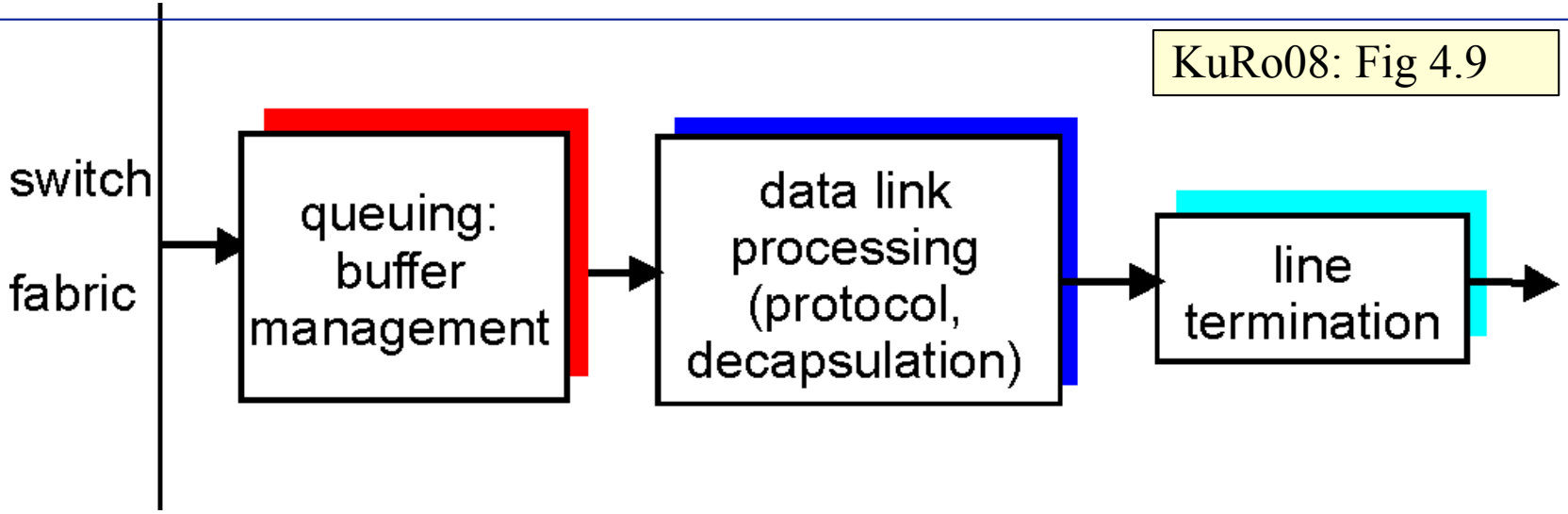
Hajautettu kytkentä

Esim. kullakin portilla on kopio reititystaulusta, voi tutkia itse

Content Addressable Memory (CAM), assosiatiivinen haku,

longest prefix match **(Cisco 8500: 64 K CAM)**

Ulosmenoportti (output port)



Puskuroi, jos paketteja tulee nopeammin kuin ulosmenon siirtonopeus salli

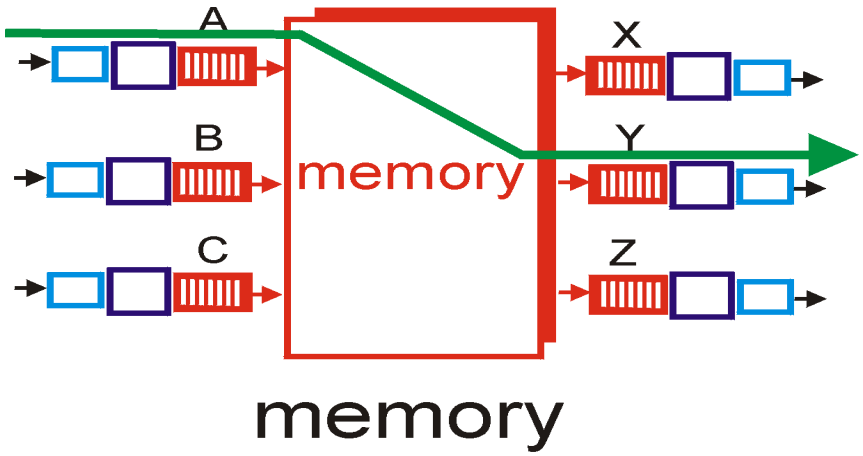
- Sisääntulo nopeampi tai monesta samaan kohteeseen

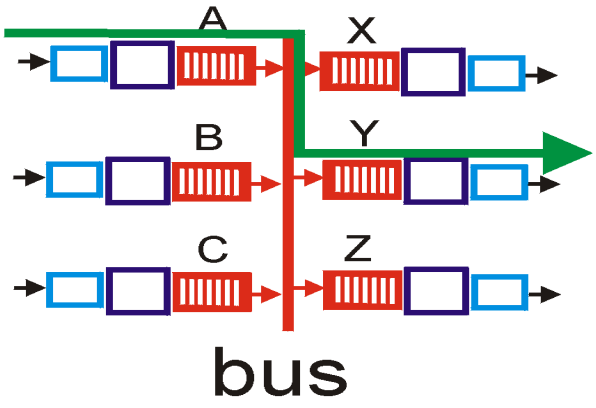
Voi käyttää priorisointia (packet scheduling)

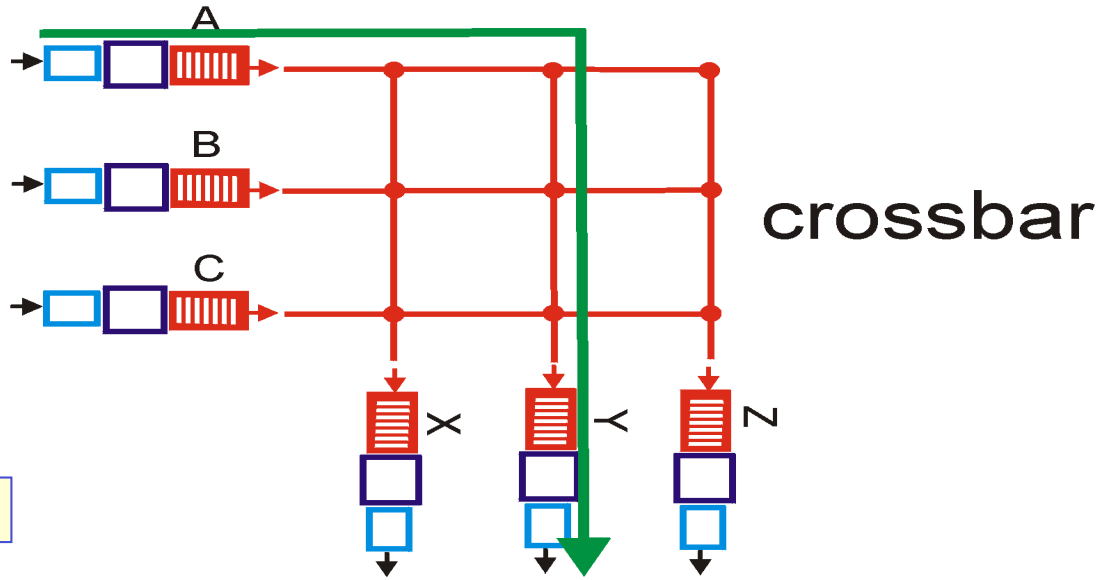
- FCFS (First Come First Serviced), WFQ (Weighted Fair Queuieng),...
- QoS (Quality of Service) **(ei käsitellä tällä kurssilla!)**

Suorittaa linkki- ja fyysisen kerroksen operaatiot

Kolme erilaista kytkentätapaa:







KuRo08: Fig. 4.8

Kytkentä muistin kautta

- “Tavallinen” tietokone reitittimenä
  - Sisääntulo: keskeytys, CPU kopioi paketin muistiin, tutkii minne on menossa
  - Ulosmeno: CPU kopioi paketin muistista
  - Väylä pullonkaula: 2 kopiointia per paketti
- Linkkikerros ja fyysinen kerros laitetoimintoja
- Jonot keskusmuistissa

Kytkentä väylän kautta

- Sisääntulo siirtää paketin väylän kautta suoraan ulosmenoporttiin
- Vain yksi kytkentä aktiivinen kerrallaan
  - Väylä edelleen pullonkaula
- Väylänopeus rajoittaa kytkentänopeutta
  - Gbps nopeudet, riittävä LAN- ja yritysverkoilla

Kytkentä kytkentäverkon kautta

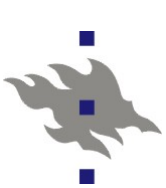
- Ristikytkentä (crossbar switch)
  - 2*N väylää yhdistää N sisääntloa ja N ulosmenoa
  - Valitse vaaka- ja pystylinja
- Jos sama ulosmeno/sisääntulo, odotus sisääntuloportissa
  - Sisääntulo voi pilkkoa paketin pienemmiksi soluiksi (cell) ja välittää yksi kerrallaan
  - Ulostulo kokoaa solut taas paketeiksi
- Suuri siirtonopeus
  - Esim. Cisco 12000: 64 Gbps

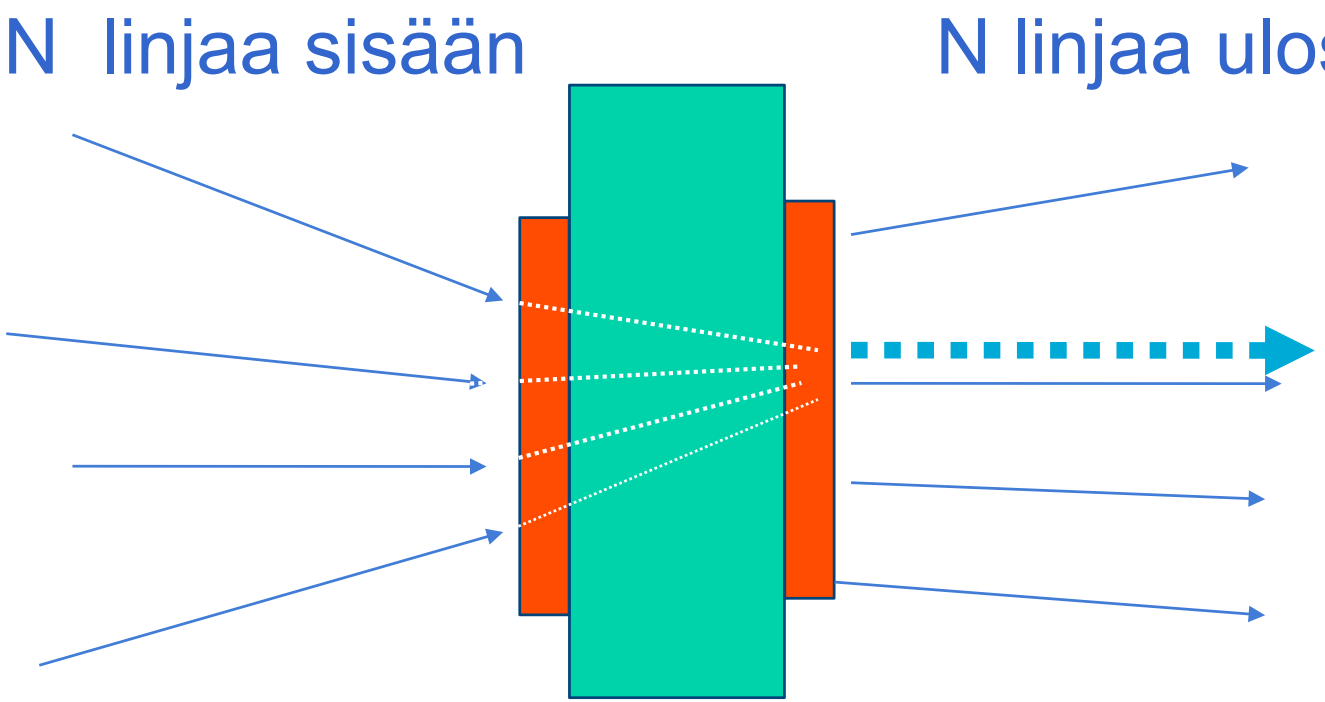
Reititysprosessori

- Suorittaa reititysprotokollaa
  - Reitityksinformaation välitystä reitittimeltä toiselle
  - RIP, OSPF, BGP,...
  - Esim. 5 minuutin välein
- Sisääntulot toimittavat reititysprotokollien paketit prosessorille
- Ylläpitää porttien reititystauluja
  - Kun muuttuu, uusi kopio kullekin portille
- Hallinta- ja ylläpitotoimintoja
  - Reitittimelläkin voi olla suoritettavana sovelluksia

Pakettien hylkäys

- Kun puskuritila ei riitä
  - Hylkää saapuva paketti (drop-tail) tai joku muu ..
  - Se kummassa jonossa paketit hylätään, riippuu kytkennän ja linjan nopeuden suhteista
  - RED (Random Early Detection): hylkää jo ennenkuin puskuri täyttyy
- Siirtovirhe
  - Linkkikerros saa hylätä virheellisen
  - Verkkokerros saa hylätä virheellisen (ICMP-protokolla)
- Paketin elinaika (Time-to-live , TTL)



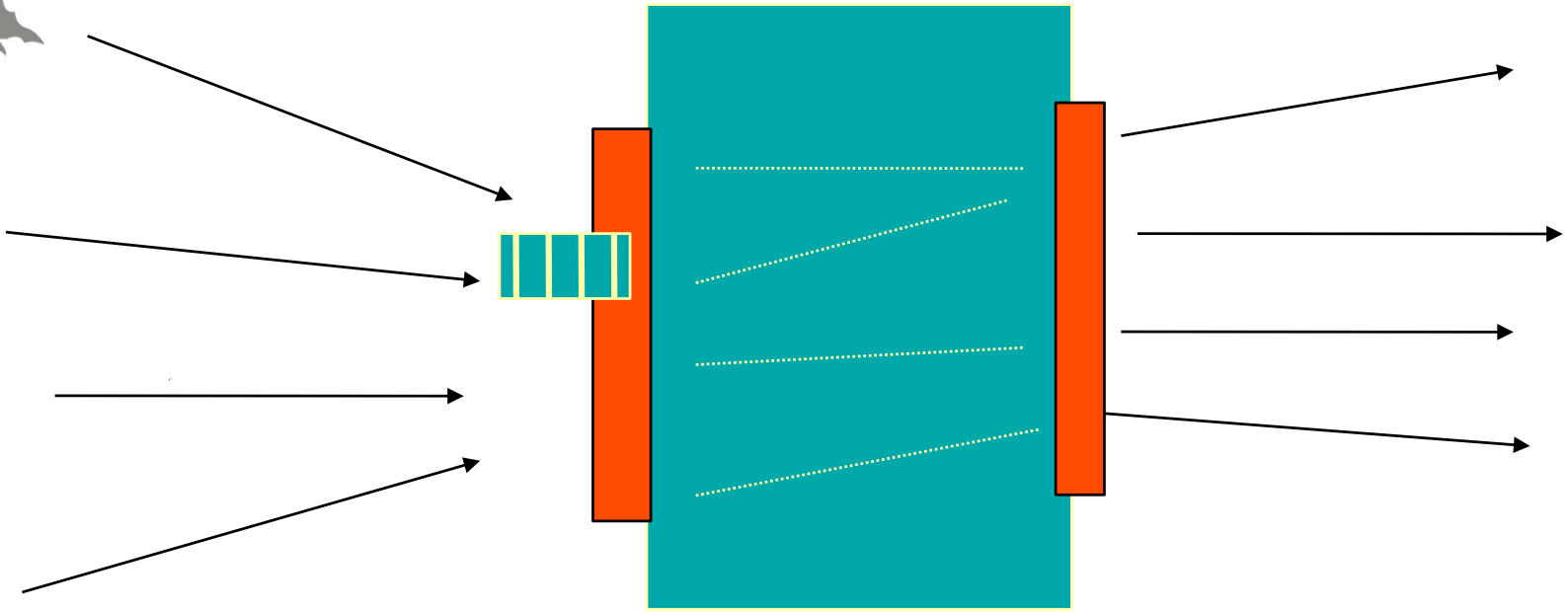


Kytkin toimii riittävällä nopeudella, joten sisääntulossa ei tarvitse jonottaa.

Yhdelle linjalle liian paljon liikennettä => ulosmenoportin puskuritila täyttyy ja paketteja katoaa!

N linjaa sisään

N linjaa ulos



Jos kytkin ei toimi tarpeeksi nopeasti, sisääntuloportteihin syntyy jonoja.

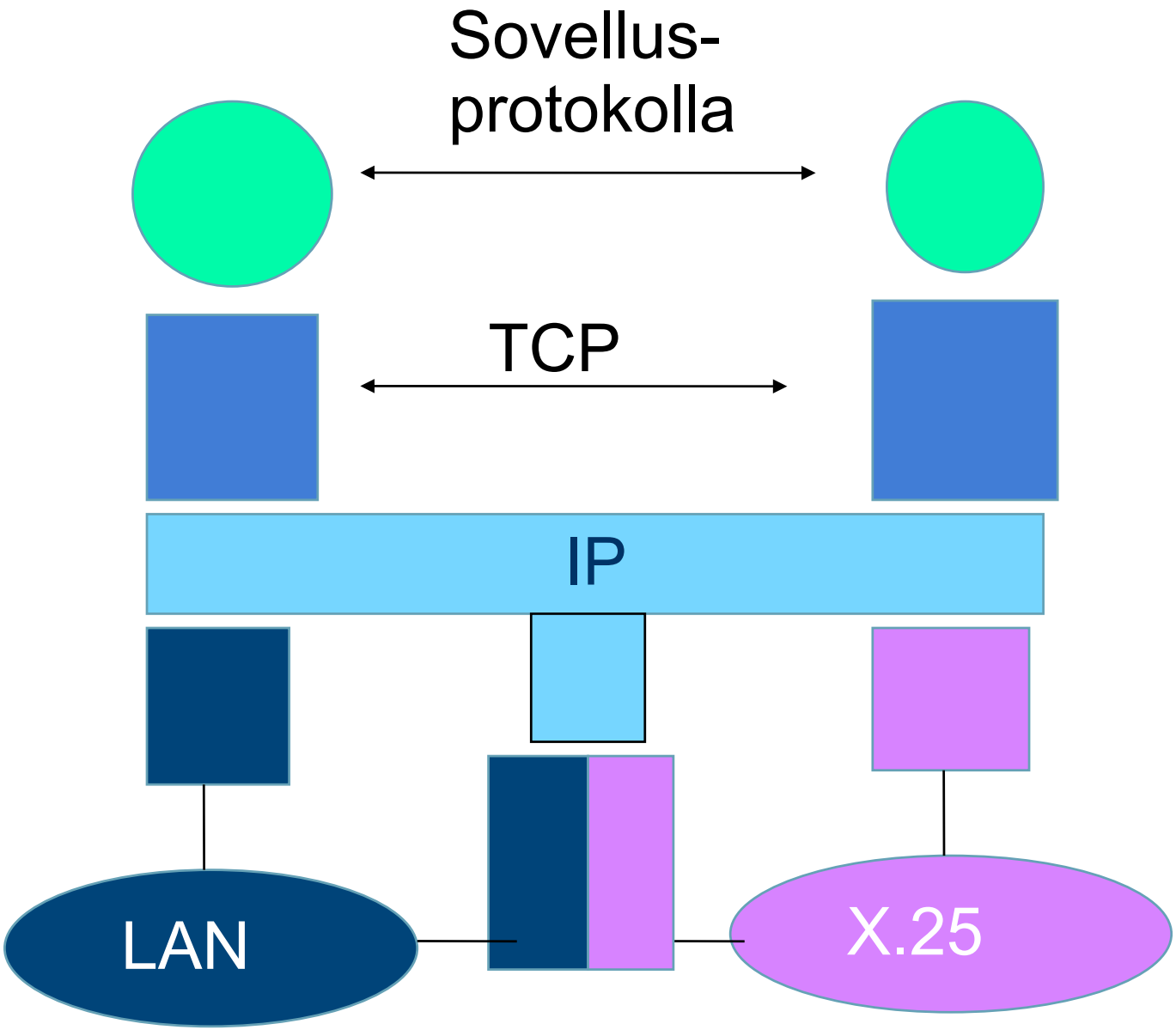
Esim. Ristikkäinkytkimessä paketti joutuu odottamaan, jos samaan kohteeseen on menossa useita paketteja. Jonottava paketti voi tukkia tien myös muilta saman portin paketeilta, jotka muuten voisivat edetä kytkimessä.

(head-of-the-line-blocking)

IP-protokolla

Ch 4.4

RFC 791


Sovellus-
protokolla
TCP
IP
LAN
X.25

Internetin verkkokerros

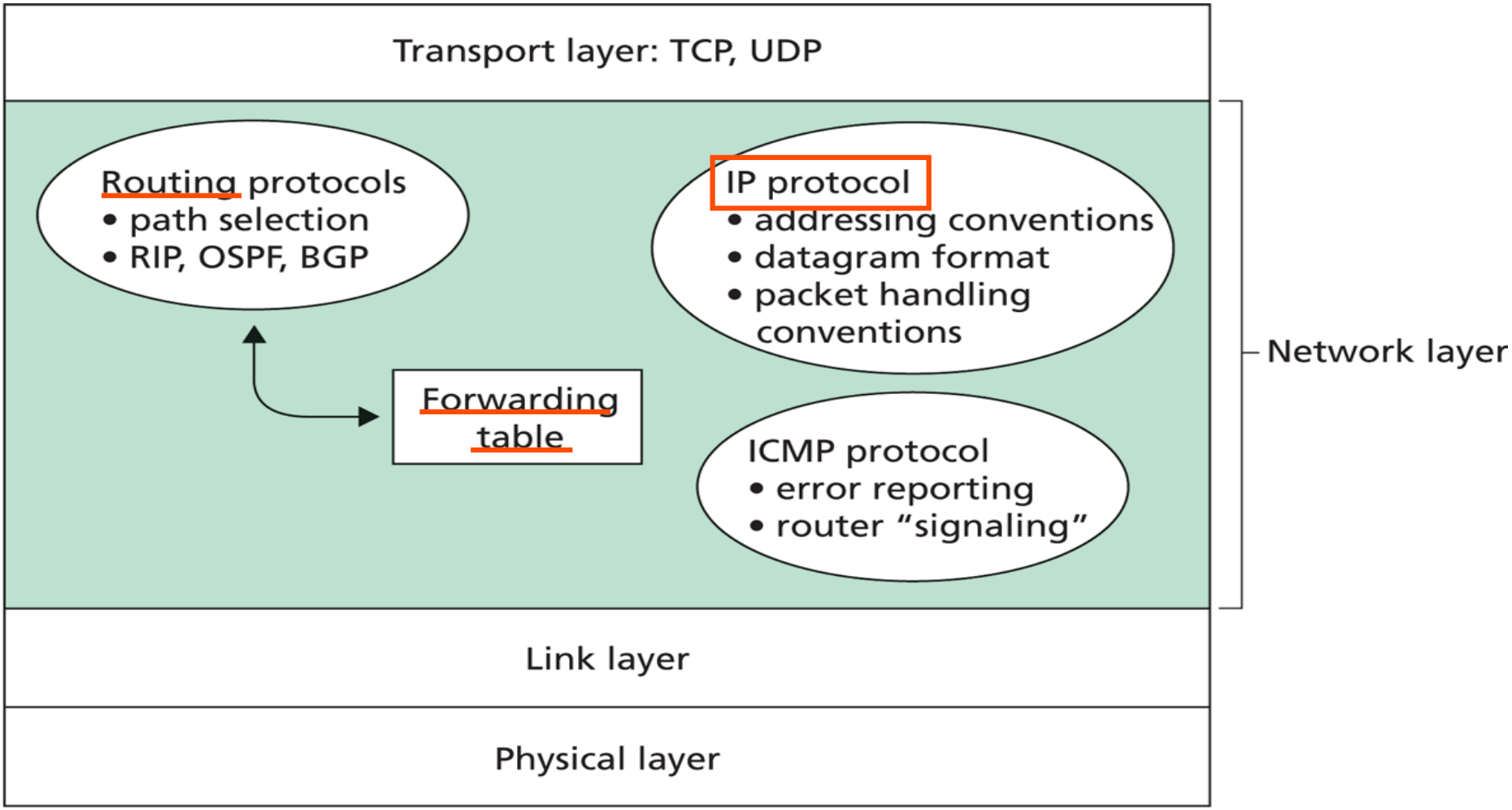


Figure 4.12 ♦ A look inside the Internet's network layer

Internetin verkkokerros

- Tällä kurssilla:
  - IPv4 ja reitityksen periaatteet
    - Etäisyysvektorireititys
    - Linkkitilareititys

Internet-protokollat kurssilla:

- IPv6, ICMP
- Reititysprotokollat
  - Reititystaulujen (forwarding table) ylläpitämistä varten
    Erillään tavallisten pakettien lähetyksestä
  - RIP (Routing Information Protocol): etäisyysvektorialgoritmi
  - OSPF (Open Shortest Path First): linkkitila-algoritmi
  - BGP (Border Gateway Protocol): hierarkkinen, autonomisten alueiden välinen algoritmi

Internetin verkkokerros

- ICMP (Internet Control Message Protocol)
  - Protokolla, jolla isännät ja reitittimet vaihtavat verkkokerroksen kuulumisia
  - Tavallaan verkkokerroksen päällä: IP-paketissa kuljetuskerroksen tietojen sijasta ICMP-dataa
  - Virheraportointi: unreachable host/network/port/protocol
    Reititin ei tiedä, minne toimittaisi ...
  - Kaiutus: echo request / reply
    tätä ping ja traceroute käyttävät RTT:n mittaamisessa
- IPv6
  - Uudistettu versio IP-protokollasta, 128 bitin IP-osoite
  - mm. kiinteänkokoinen otsake, ei tarkistussummaa,
  - pakettien paloittelu jo lähettäjän koneessa

IP-protokolla

- Verkkokerros siirtää kuljetuskerroksen segmentit lähdekoneelta kohdekoneelle
- Tehtävässä tarvitaan
  - Osoitteet (lähettäjä, vastaanottaja)
  - Tieto ylemmän kerroksen protokollasta (UDP, TCP tai joku muu), jotta osaa antaa oikealle rutiinille
  - Liian ison IP-paketin paloittelu tarvittaessa pienemmiksi IP-paketeiksi
  - 'Harhautuneiden' pakettien hävittäminen (time-to-live)
  - Tarkistukset (checksum)
- Hyviä ominaisuuksia (?)
  - Siirtopalvelun eriyttäminen erityyppisille sovelluksille
  - Lähdereititys (source routing): lähettäjä määrää reitin, paketissa tieto siirtopolusta

IP-paketin rakenne (IPv4)

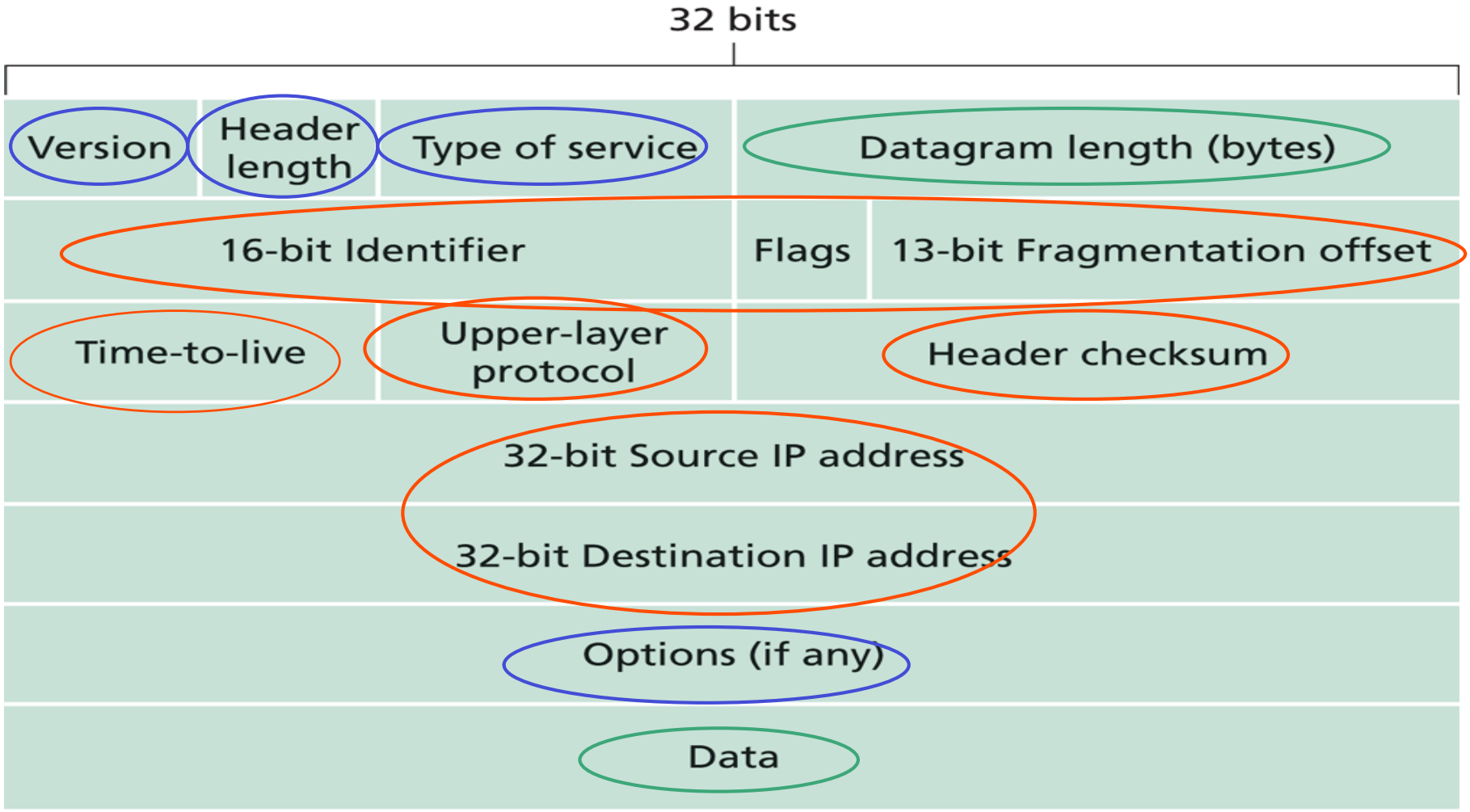


Figure 4.13 ◆ IPv4 datagram format

IP-otsake

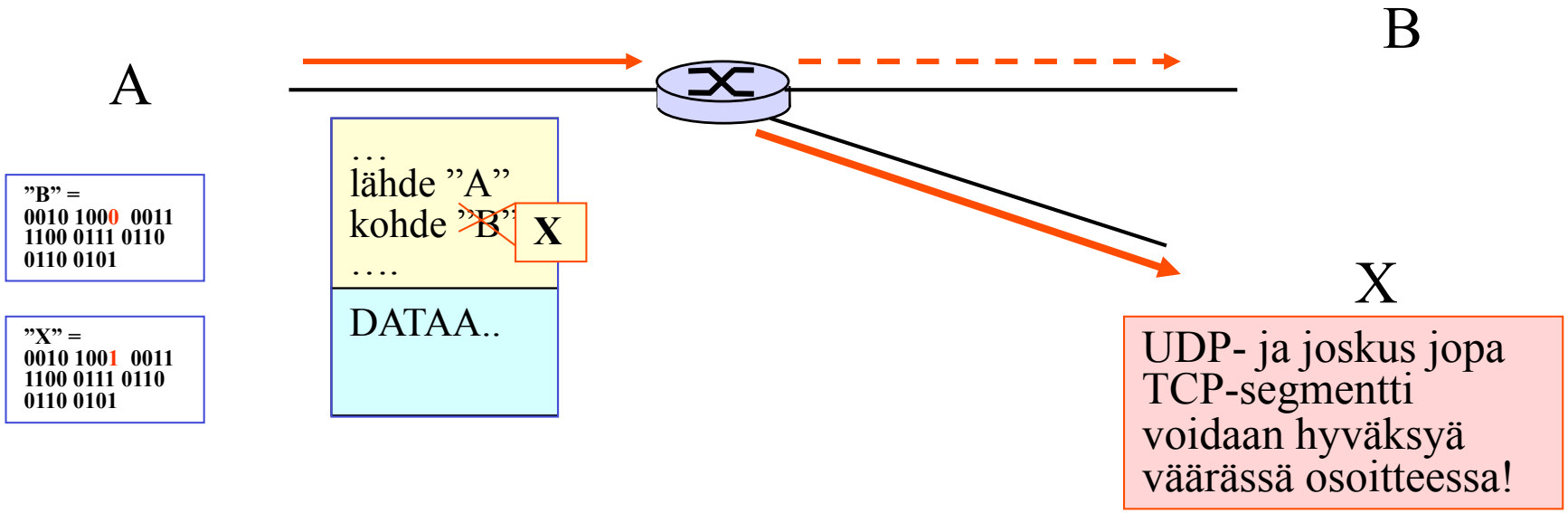
- Versionumero
  - IPv4 vai IPv6, kummallakin erilainen otsake
- Otsakkeen pituus (header length)
  - Vaihtelevan pituinen optiokenttä, **mininimi on 20 B**
- **TOS-kenttä (Type of Service)**
  - **Varattu halutun palvelun kertomiseen:**
    - **Nopeus, luotettavuus, kapasiteetti; ääni vs. tiedosto**
  - **Yleensä ei ole käytössä (osa käytössä uusissa reitittimissä)**
- **IP-paketin pituus (Datagram length)**
  - **Koko IP-paketin pituus tavuina, maksimi 65535 B**
  - **Tavallisimmin 576-1500 B**
- **Paketin tunniste (16-bit identifier), lippuja (flags), palan paikka (fragmentation offset)**
  - **Paketin pilkkomiseen pienemmiksi ja kokoamiseen takaisin isoksi**

IP-otsake (jatkuu)

- **Elinaika** (time-to-live, TTL)
  - Rajoittaa paketin elinaikaa, maksimi 255
  - Vähenee joka hypyllä reitittimestä toiseen, kun TTL=0, hylätään
- **Kuljetettu protokolla** (Upper-layer protocol)
  - Kumpi kuljetuskerroksen protokolla (TCP=6, UDP=17) vai kenties verkkokerroksen sisäistä dataa (ICMP, reititysprotokolla)
- **Otsikon tarkistussumma** (Header checksum)
  - Vain otsakkeelle (Internet checksum)
  - Tarkista ja laske uusi joka reitittimessä (TTL, Options)
  - Hylkää virheellinen paketti
- **Osoitteet** (Source IP Address, Destination IP Address)
  - Lähteen ja kohteen IP-osoitteet
- **Optiot** (Options)
  - Laajennuksia: mm. lähdereititys, harvoin käytetty

IP-otsakkeen tarkistussumma

- Miksi tarkistussumma IP-otsakkeelle?



- Miksi vain otsakkeelle?
  TCP:llä ja UDP:llä omat tarkistussummat.

IP-pakettien paloittelu (fragmentointi)

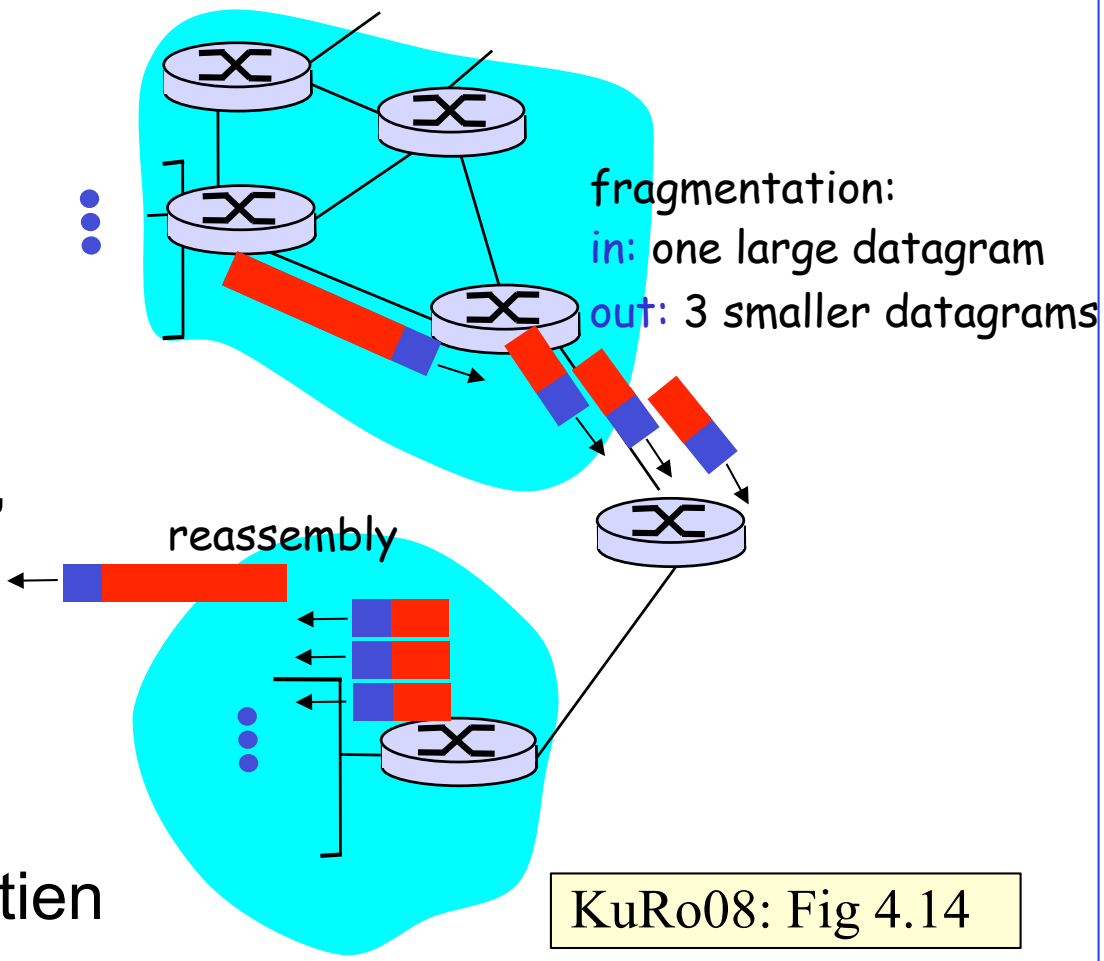
Maximum transfer Unit (MTU)

- suurin mahdollinen IP-paketti
- eri linkeillä eri koko
- Esim. Ethernet 1500 B

Liian iso paketti pilkottava reitittimessä pienemmiksi paketeiksi (fragmenteiksi), jotka kohdekone kokoaa

- voivat kukin kulkea eri reittiä

IP-otsakkeessa kentät yhteenkuuluvien fragmenttien tunnistamiseksi



KuRo08: Fig 4.14

Pakettien palastelusta

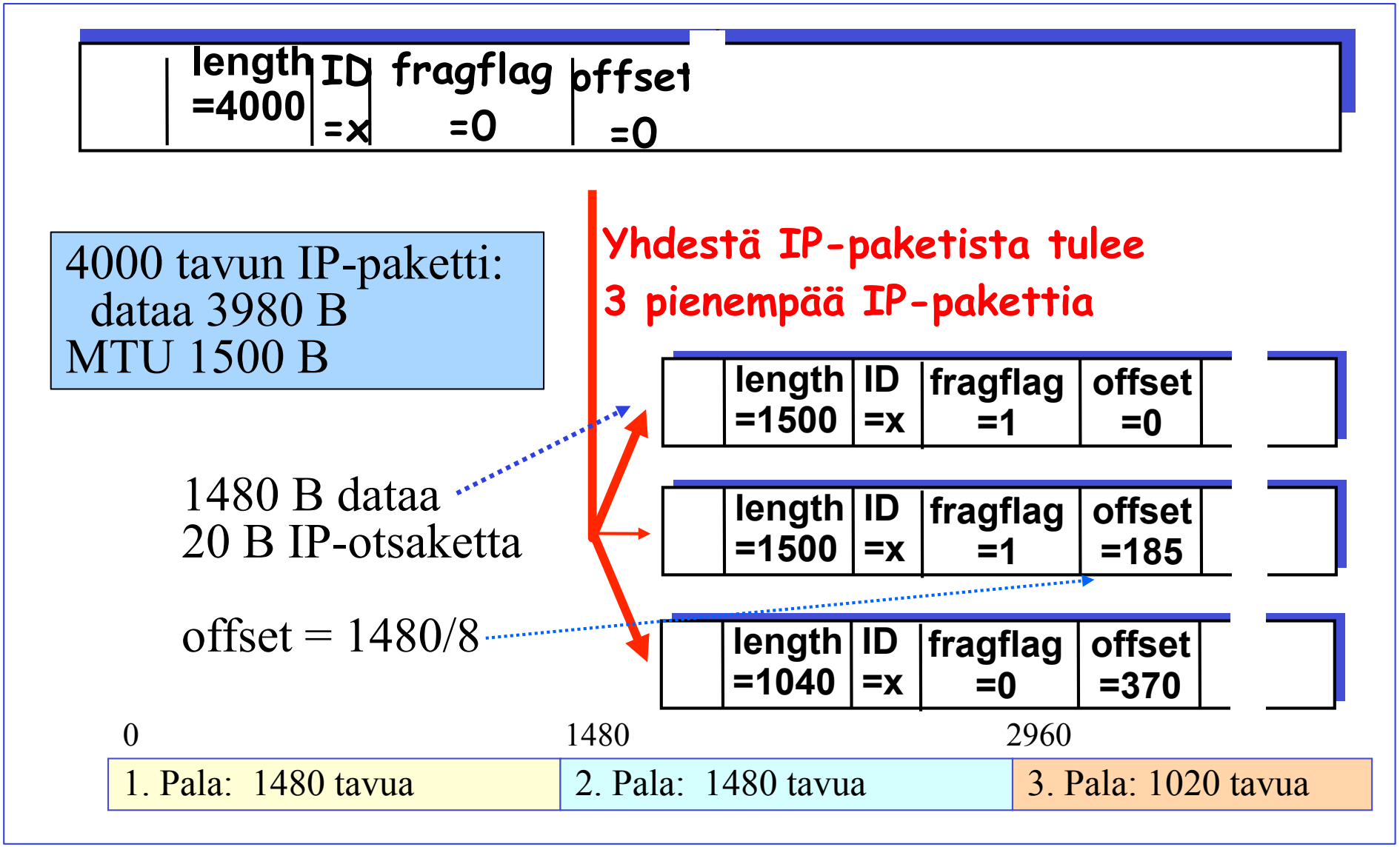
- Path MTU Discovery (PMTUD)
  - Käytetään don’t fragment lippua (DF) ja eri kokoja
  - Kokeillaan lähetystä kunnes fragmentti menee perille
  - Reitittimet lähettävät ICMP viestin “fragmentation needed” jos eivät voi välittää pakettia eteenpäin
  - Saadaan MTU (Maximum Transfer Unit)
  - TCP käyttää tätä tietoa (MSS, käyttöjärjestelmä asettaa)
- IPv4 reitittimet
- IPv6 isäntäkoneet

IP-pakettien fragmentointikentät

- **Paketin tunniste** (16-bit identifier)
  - Sama kaikissa IP-paketin fragmenteissa
- **Lippuja** (flags)
  - DF-bitti (Don't fragment) kieltää paloittelun, esim. jos vastaanottaja ei kykene kokoamaan
  - MF-bitti (More fragments)

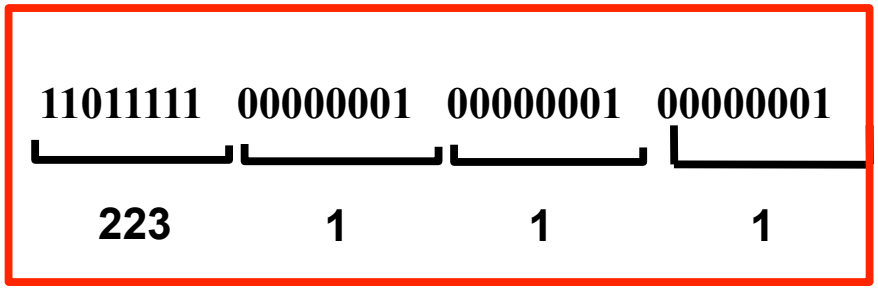
    0= paketin viimeinen fragmentti, 1= ei vielä viimeinen
- **Fragmentin sijaintipaikka** (13-bit Fragmentation Offset)
  - paikka alkuperäisessä IP-paketissa siirtymänä paketin alusta
  - 13 bittiä => 8192 eri arvoa; Jotta pystyisi käsittelemään täysimittaiset segmentit (= max 65535 tavua) => siirtymä esitetään 8 tavun monikertoina => fragmenttien myös oltava 8 tavun monikertoja (paitsi viimeinen)
- IPv6 ei käytä fragmentointia

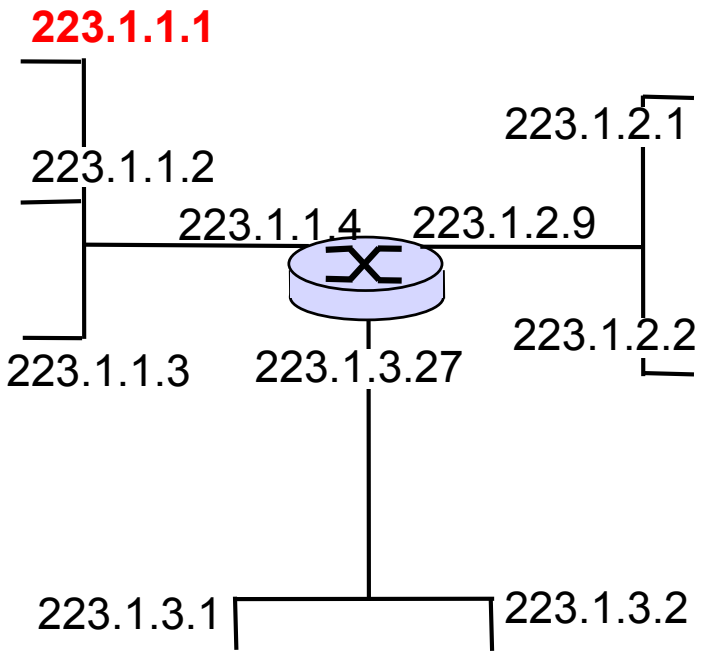
Esimerkki


length =4000
ID =x
fragflag =0
offset =0
4000 tavun IP-paketti:
dataa 3980 B
MTU 1500 B
Yhdestä IP-paketista tulee
3 pienempää IP-pakettia
1480 B dataa
20 B IP-otsaketta
offset = 1480/8
length =1500
ID =x
fragflag =1
offset =0
length =1500
ID =x
fragflag =1
offset =185
length =1040
ID =x
fragflag =0
offset =370
0
1480
2960
1. Pala: 1480 tavua
2. Pala: 1480 tavua
3. Pala: 1020 tavua

IP-osoitteet

- 32 bittinen tunniste isäntäkoneille ja reitittimien linkeille
  - verkkoliittymän tunniste
- Reitittimellä useita liittymiä
  - kullakin oma IP-osoite
- Myös isäntäkone voi olla liitettynä useaan verkkoon



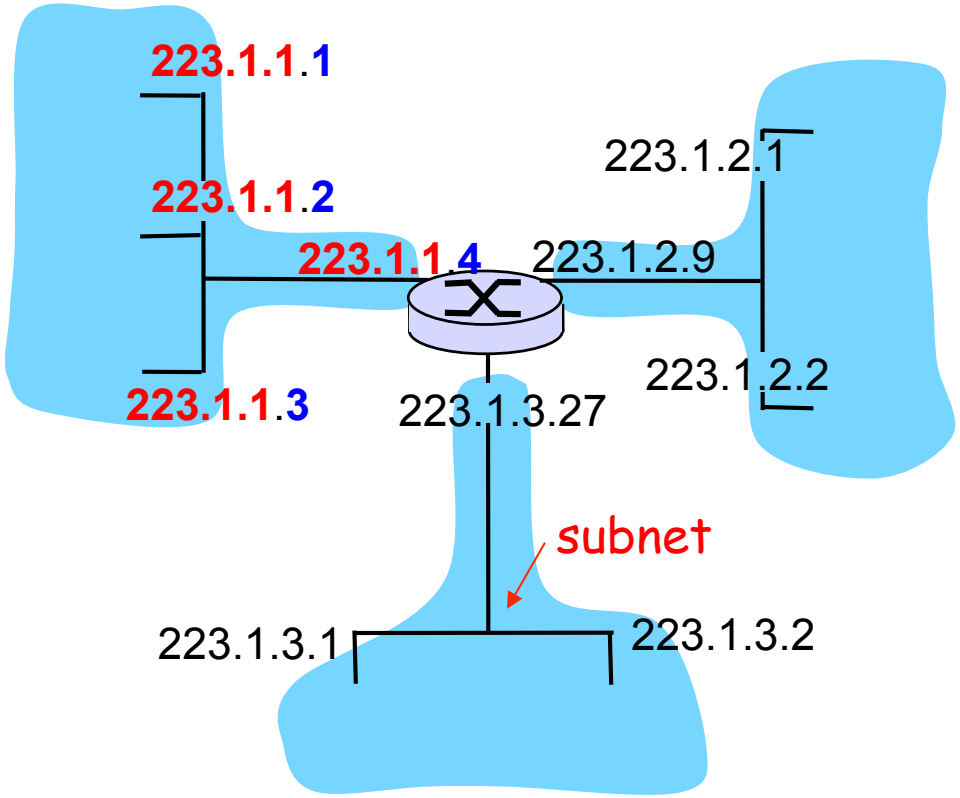


KuRo08:Fig 4.15

ICANN Internet Corporation for Assigned names and Numbers verkkonumerot palvelun tarjoajille, 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

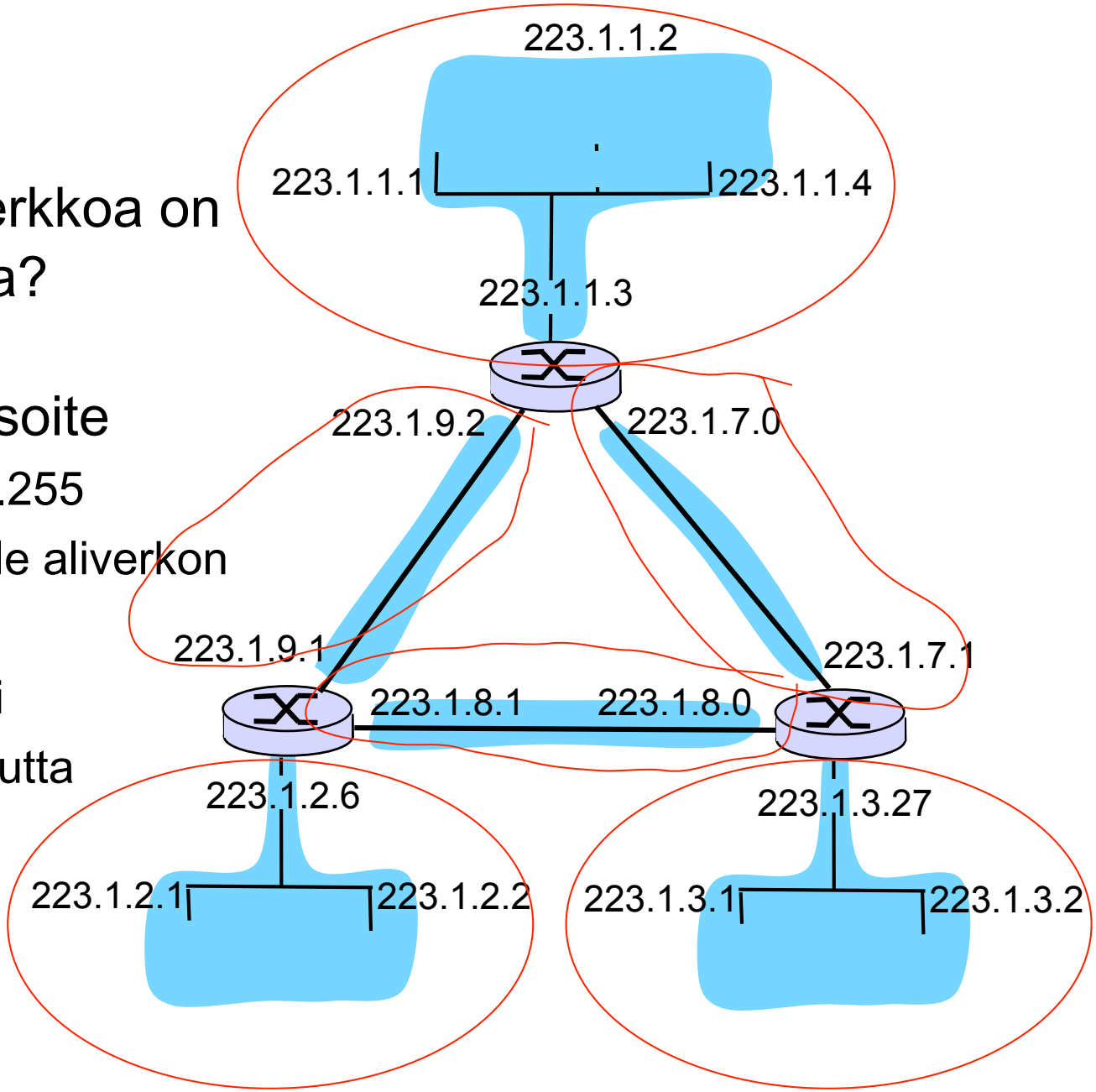


network consisting of 3 subnets

KuRo08: Fig 4.15

Aliverkot

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



CIDR: Classless InterDomain Routing

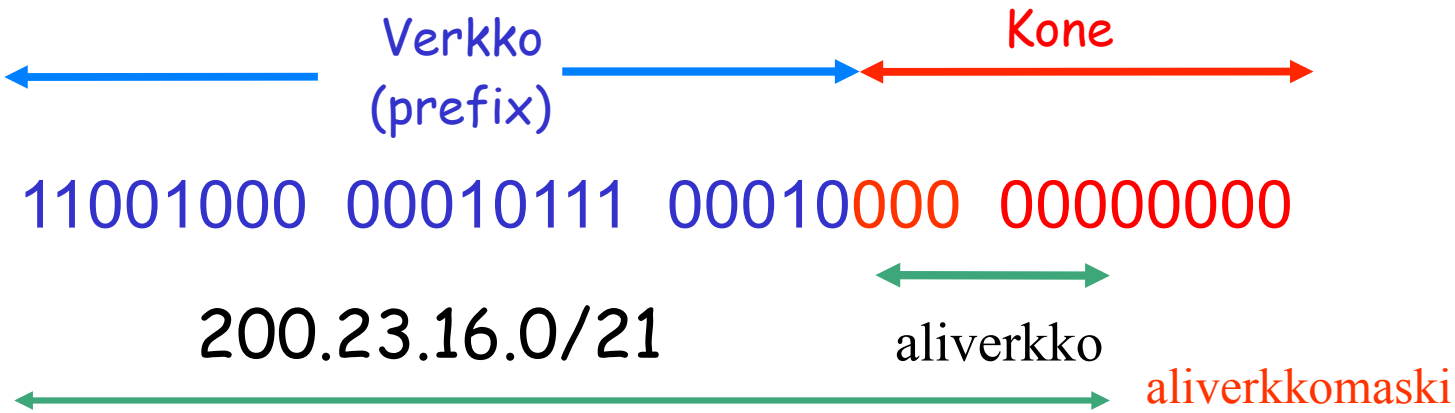
- Verkko-osa voi olla minkä tahansa kokoinen

  Vanha luokallinen osoite: A-luokka 8 b, B-luokka 16 b,C-luokka 24 b

- Formaatti: a.b.c.d/x

  x ilmoittaa verkko-osan bittien lukumäärän (prefix)

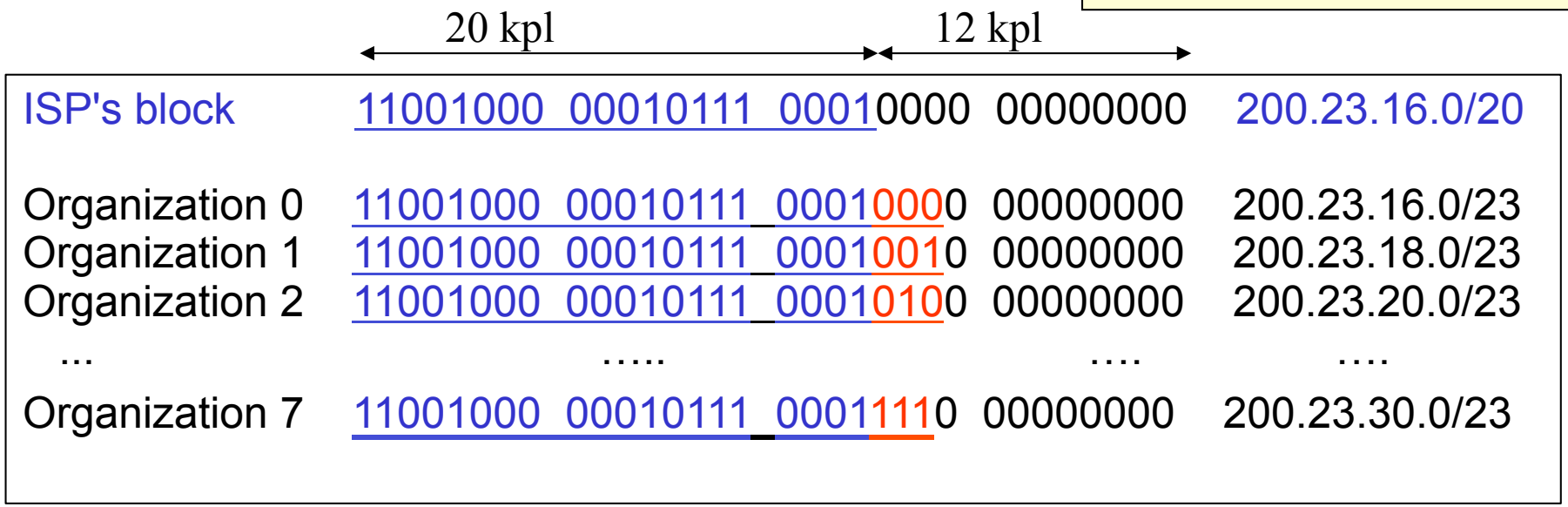
  Esim. Organisaatio, jolla 2000 konetta varaa 2048 = 2^11 konenumeroa, jolloin verkko-osaa varten jää 21 bittiä
  Yritys voi vielä itse jakaa viimeiset 11 bittiä aliverkko-osoitteeksi ja koneosoitteeksi. Tämä jako ei näy ulkopuolelle.



Koneen IP-osoite

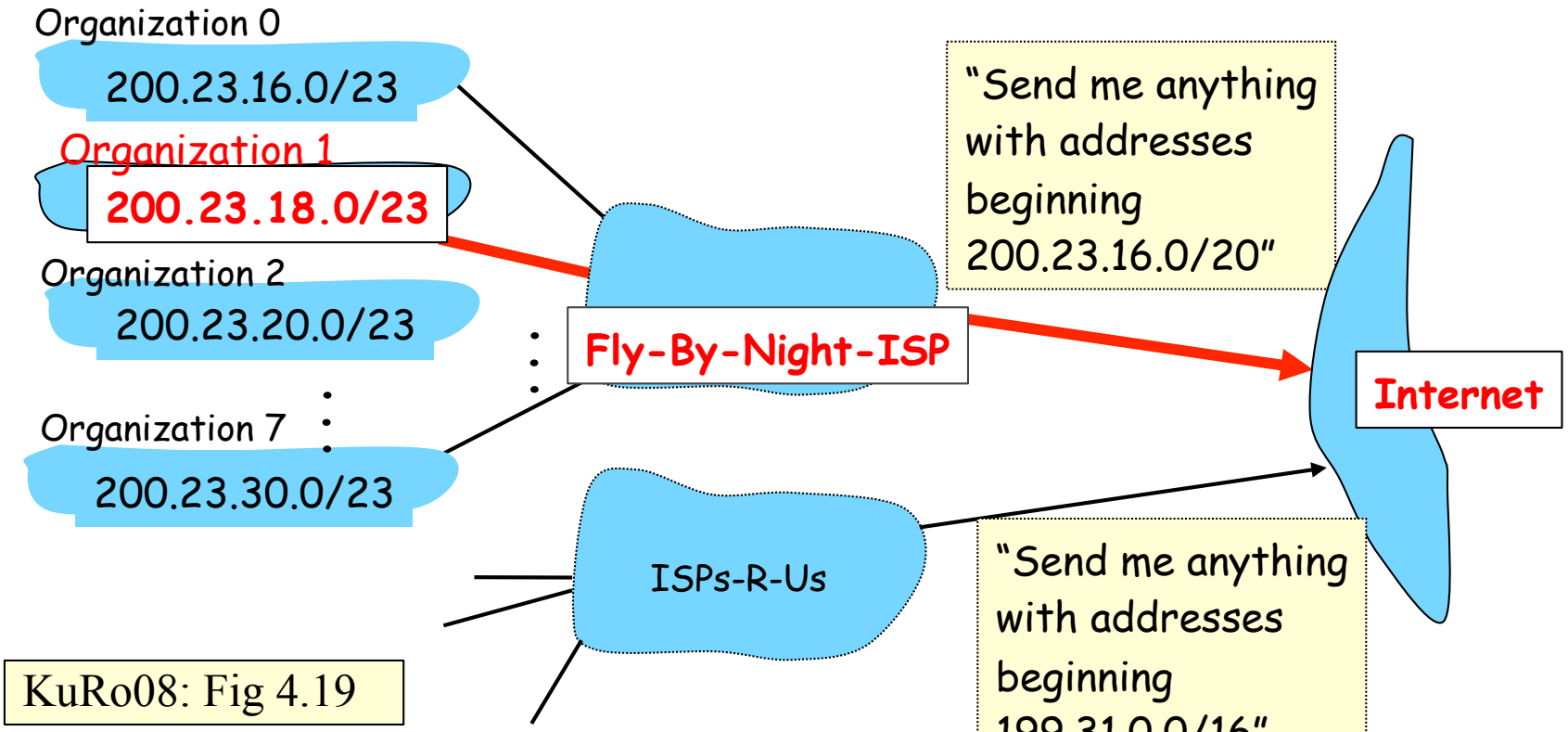
- Palveluntarjoaja saa verkkonumeronsa ICANN:lta isona lohkona
- voi jakaa saamansa osoiteavaruuden (osoitelohkon) edelleen aliverkkoihin
  - esim. Kukin organisaatio saa aliverkon, jossa on numerot 512 koneelle

2**12 = 4096 = 8 * 512



Hierarkkinen osoite

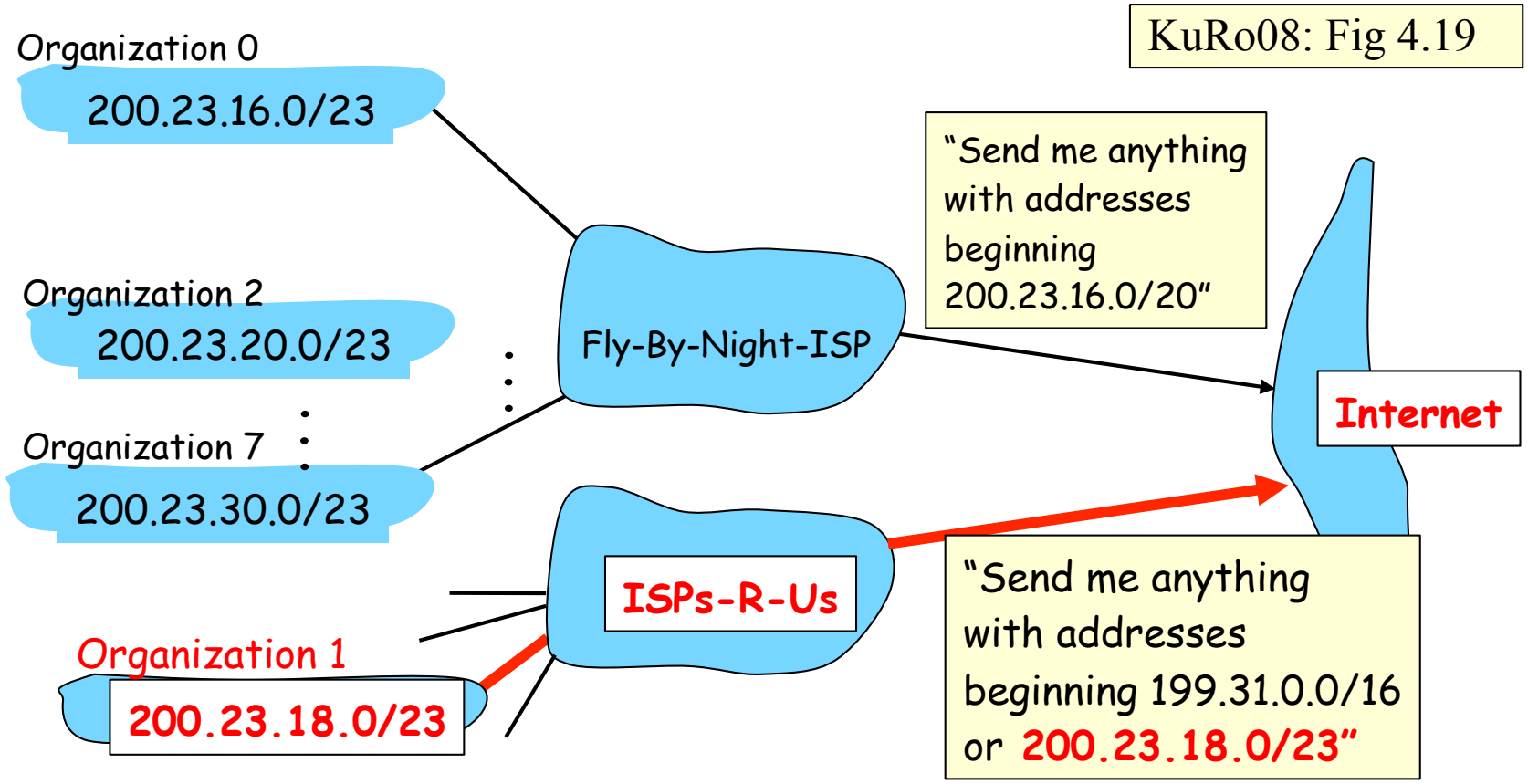
- CIDR luo reititystä helpottavan hierarkian
  - Aggregointi (yhdistäminen): yhteinen alkuosa => samaan suuntaan



KuRo08: Fig 4.19

Jos palveluntarjoaja (ISP) vaihtuu?

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

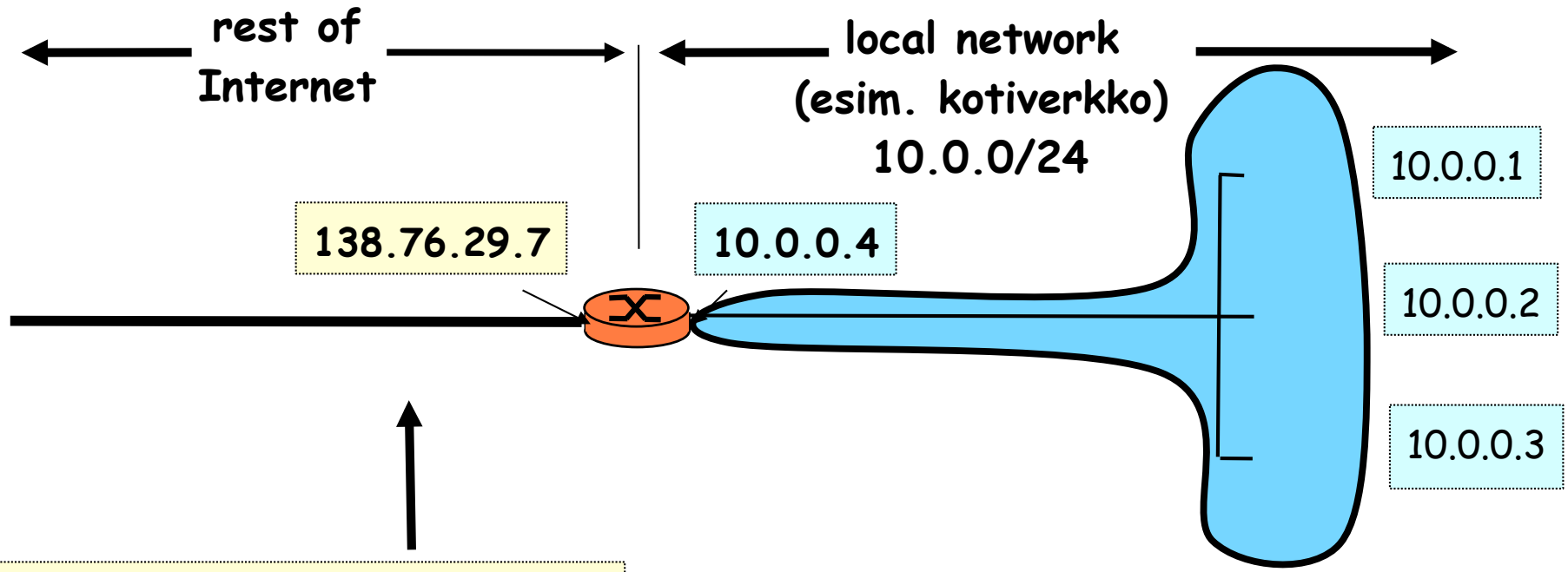


Koneen IP-osoite

- Koneen IP-osoite konfiguroidaan (usein) käsin koneelle
- Tai yhä useammin saadaan automaattiseti 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
  - WLAN
    - “wash-and-go”, “plug-and-play”

NAT: Network Address Translation

Vain ~ 4 miljardia osoitetta!



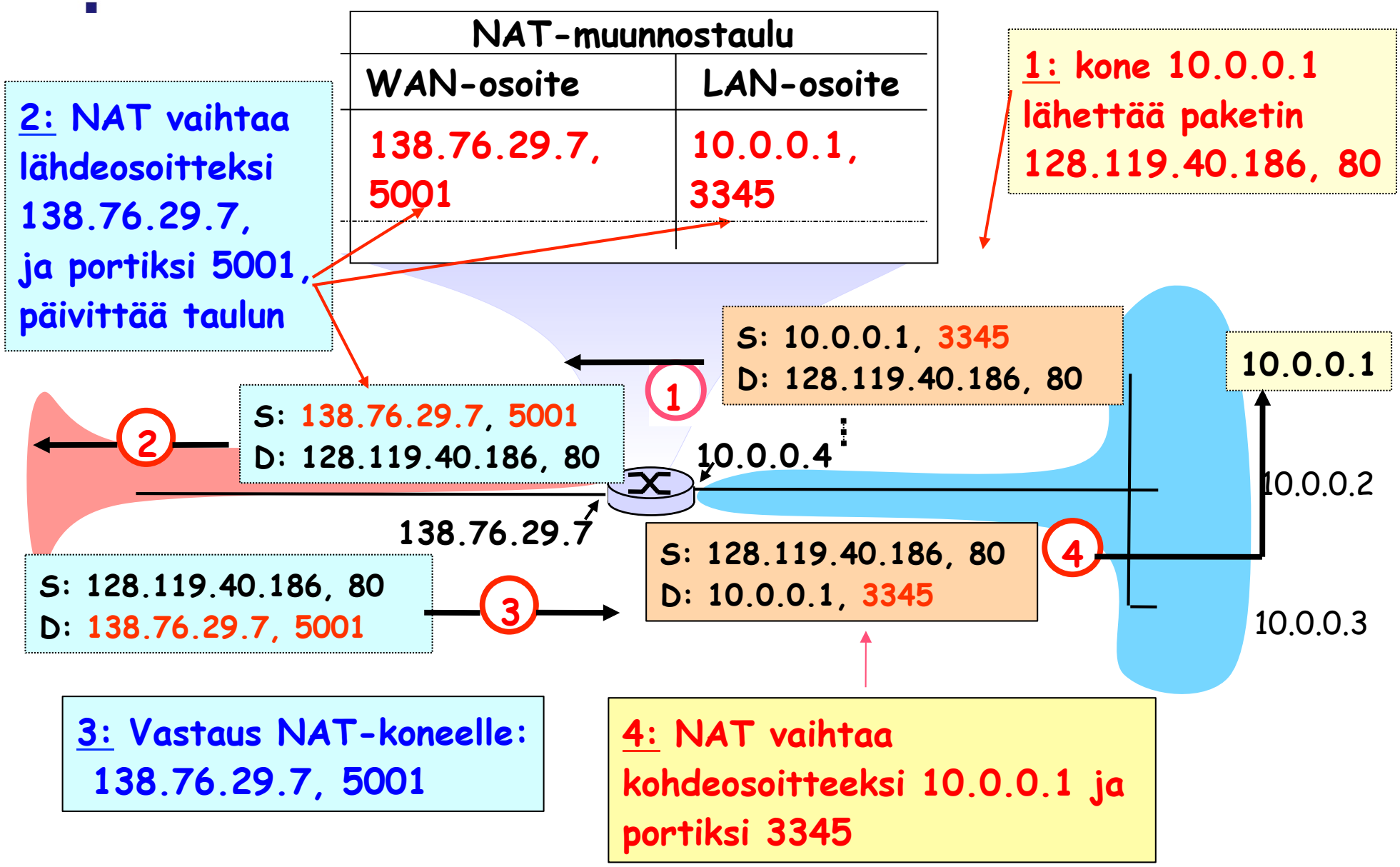
Kaikilla ulosmenevillä ja sisääntulevilla paketeilla sama IP-osoite
138.76.29.7
mutta eri porttinumeroita.

Kotiverkossa käytössä sisäiset IP-osoitteet
10.0.0/24
(esim. DHCP:llä)

NAT-reititin

- Ulosmenevät paketit
  - Korvaa lähdekoneen IP-osoite ja porttinumero NAT-koneen IP-osoitteella ja NAT-koneen valitsemalla 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: 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)
  - Onko ohjelmoijaan 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

- **Etsii edullisimmat reitit lähdekoneelta kohdekoneille**
  - **Käytetään reititystaulun muodostamiseen**
    - Mille linkille paketti seuraavaksi siirretään tältä reitittimeltä
- **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 vs. ei
  - Linkin ruuhkautuneisuus voi vaikuttaa sen kustannukseen
  - Nykyalgoritmit eivät ota kuormitusta huomioon
    - Tosin kyllä epäsuorasti linkin hitautena ('kustannuksena')

Reititystiedon kerääminen

- Reitin saa tietoja seuraavasti:
  - Linkkikerros tarjoaa yhteyden naapureihin
    - MAC-osoite → IP-osoite
  - Naapurit havaitaan saapuvista kehyksistä ja paketeista (broadcast, unicast)
- Naapurit kertovat omasta reititystaulustaan
  - Linkkitila: kaikki tiedot
  - Etäisyysvektori: lyhimmät etäisyydet kohteisiin naapurien kautta
- Tietojen päivitys: reaktiivinen tai ajastettu

Verkko graafina (graph)

Verkko G= (N,E)

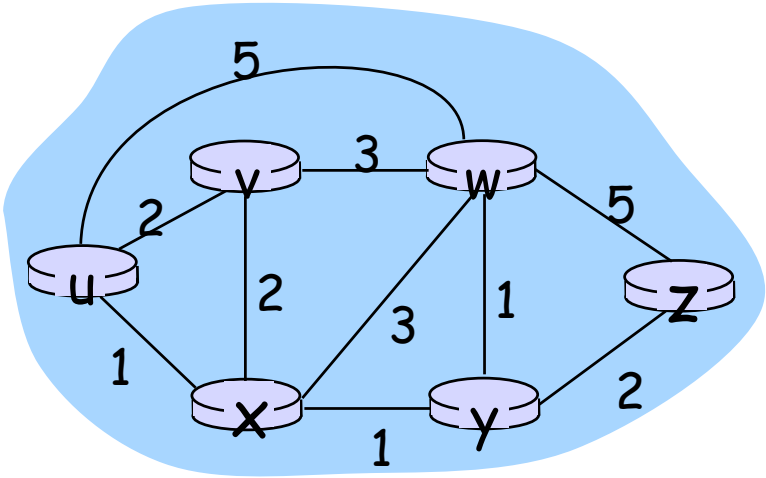
N = solmujen (nodes) joukko

E = linkkien (edges) joukko

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

c(x,y) = linkin kustannus

kaistanleveys, ruuhkaisuus, raha, ..



KuRo08; Fig. 4.27

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

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

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
- **Merkinnät**

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

D(v) toistaiseksi edullisin kustannus solmuun v

p(v) solmun v edeltäjä reitillä

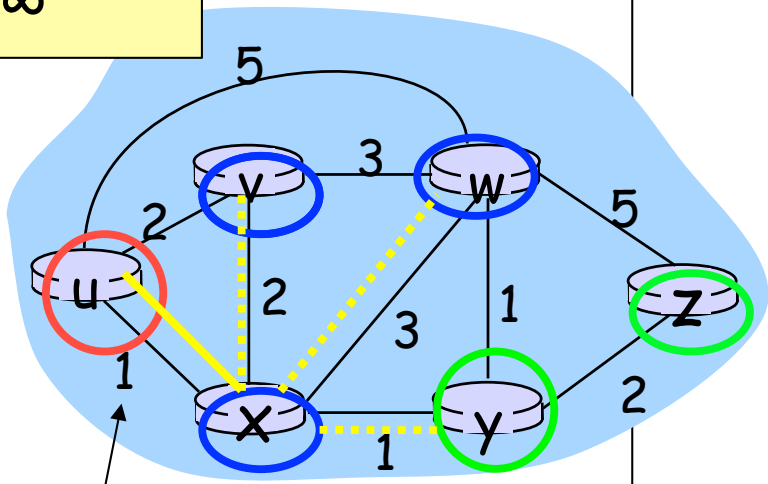
N = solmujen joukko, N' = jo käsiteltyjen solmujen joukko

Dijkstran algoritmi

D(v)=2, D(w) = 5, **D(x)=1**
D(y) = ∞ , D(z)= ∞

1. Eli jos u:n vieressä

```
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'
```



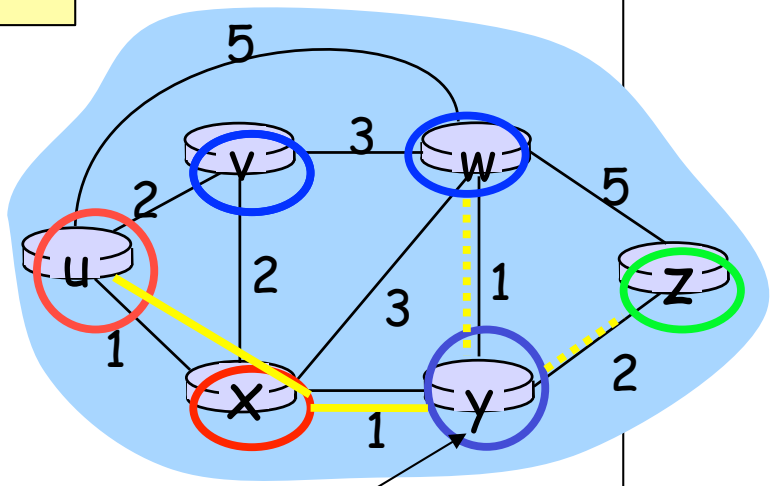
2. Aina valitaan käsittelemätön jonka etäisyys u:sta on pienin

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

Dijkstran algoritmi 2

D(x)=1, D(v)=2, D(w)=4, D(y) = 2 , D(z)= ∞

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



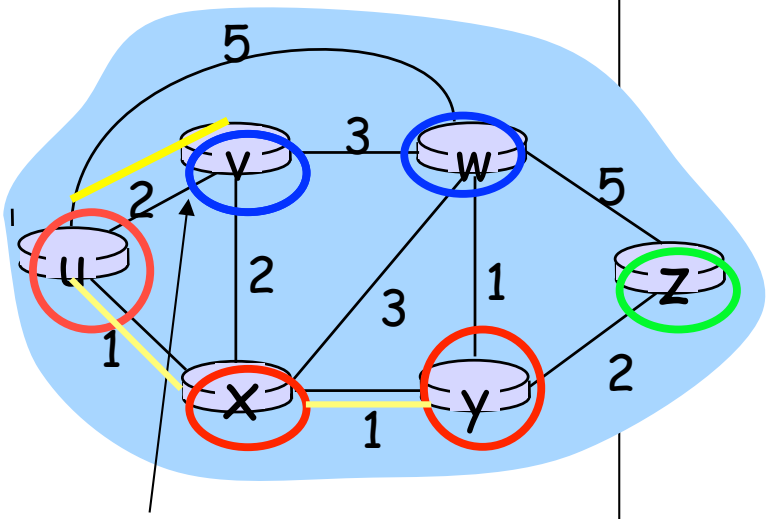

```
8   Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15  until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

Dijkstran algoritmi 3

D(x)=1, D(v)=2, D(w)=4, D(y) = 2 , D(z)= ∞

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

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



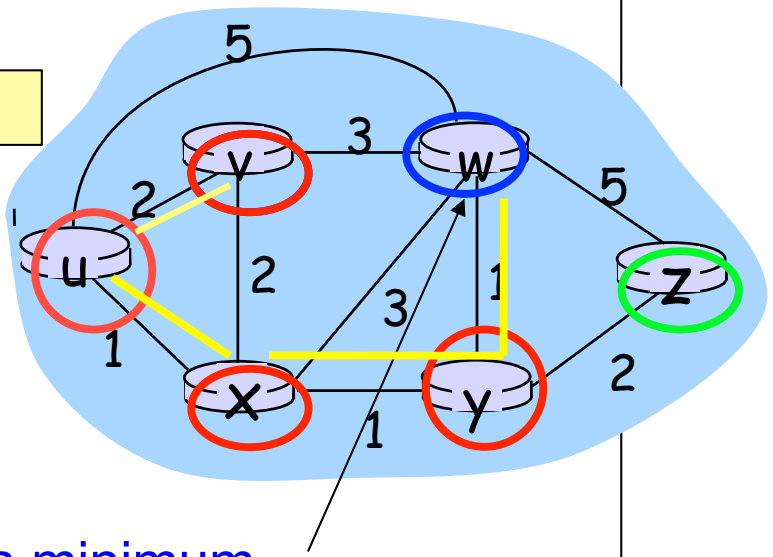

```
8   Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15  until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

Dijkstran algoritmi 4

D(x)=1, D(v)=2, D(w)=4, D(y) = 2 , D(z)= ∞

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

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



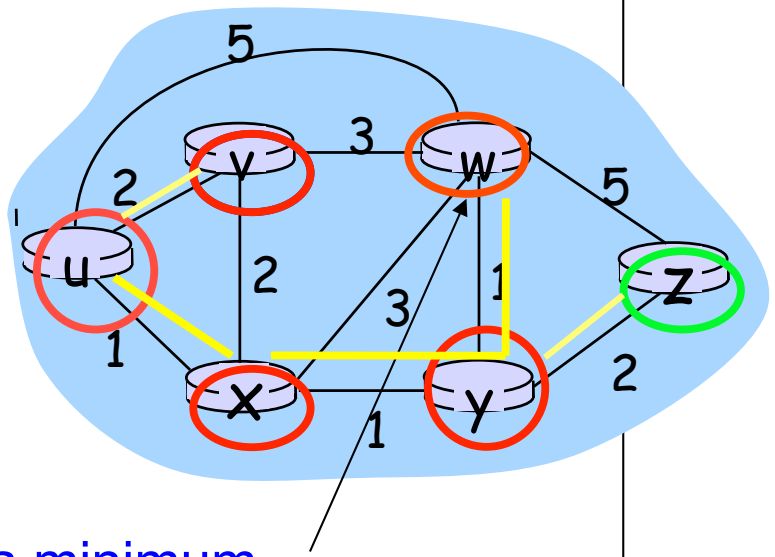

```
8   Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15  until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

Dijkstran algoritmi 4

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

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

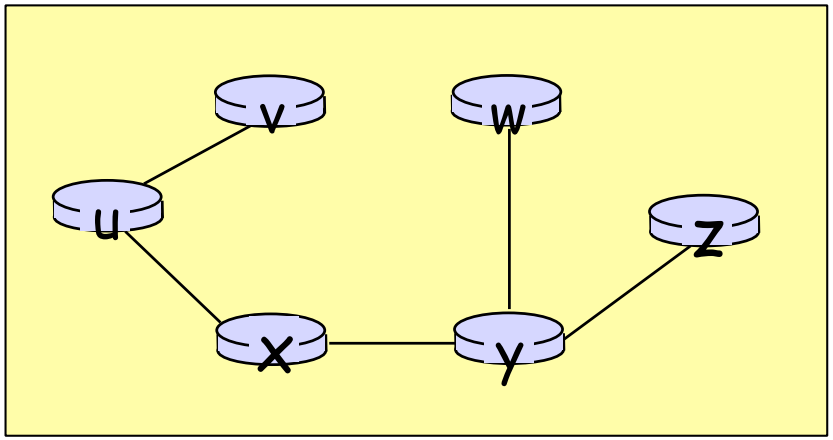
D(x)=1, D(y) = 2, D(v)=2, D(w)=3, D(z)=4




```
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

Resulting shortest-path tree from u:



KuRo08: Fig. 4.28

Resulting forwarding table in u:

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

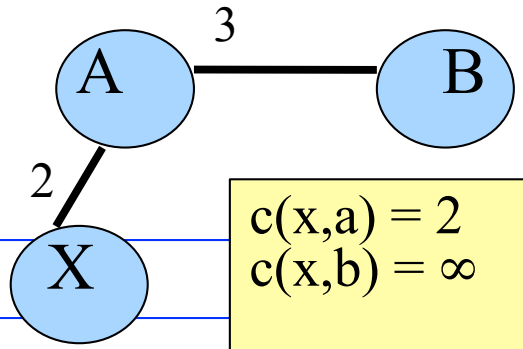
2) Etäisyysvektorireititys (distance vector)

- **Arpanet-verkon alkuperäinen reititysalgoritmi**
  - Käytössä useissa Internetin reititysprotokollissa 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

Etäisyysvektorireititys (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 kustanuksiltaan edullisin reitti

Etäisyysvektorireititys



c(x,a) = 2
c(x,b) = ∞

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

Merkinnät

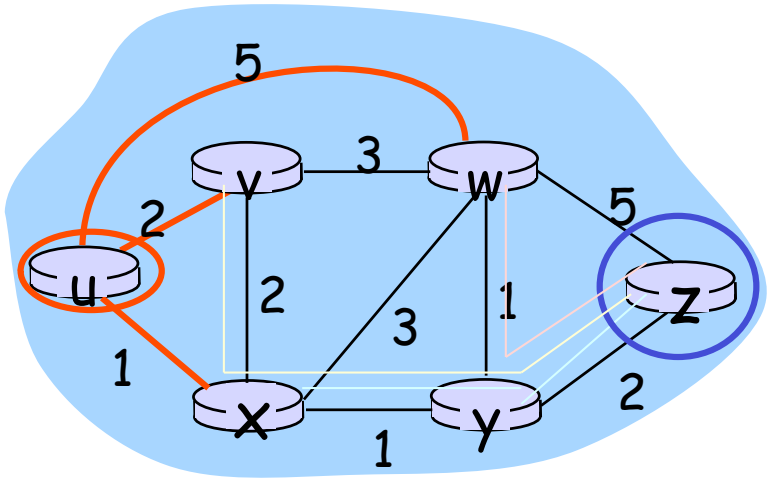
$c(x,v)$ kustannus solmusta x naapuriin v,
jos v ei ole x:n naapuri, $c(x,v) = \infty$

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

- **Kukin solmu ylläpitää omaa etäisyysvektoria kaikkiin tuntemiinsa kohteisiin** $D_x = [D_x(y): y \epsilon N]$
  - **edullisin tiedetty kustannus solmusta x kuhunkin solmuun y**
- **Sekä saa naapureiltaan niiden etäisyysvektorit**

  $D_v(y) = [D_v(y): y \epsilon N]$ **= 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 solmuun y**
  - **Reittejä useita (eri naapureiden kautta); valitaan edullisin eli pienin kustannus**

Esimerkki 1



Kohde	kust. linkki
Z	4 X:ään

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

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

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:

Node x table

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

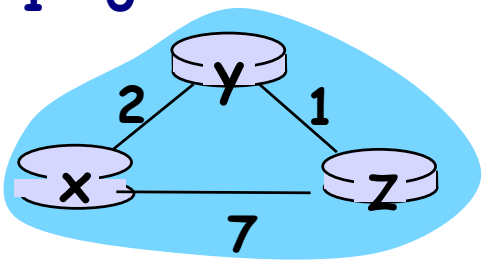
Node y table

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

Node z table

from \ cost to	x	y	z
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:

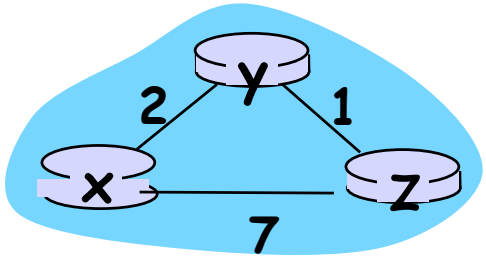
Samalla tavalla toimivat solmut y ja z:

y:

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

z:

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0



Solmut lähettävät taas tietonsa toisilleen ja laskevat uudet uudet lyhimmät reitit.

X:

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

Y:

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

Z:

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

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

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

node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	**3**
y	2	0	1
z	7	1	0

cost to

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

node y table

cost to

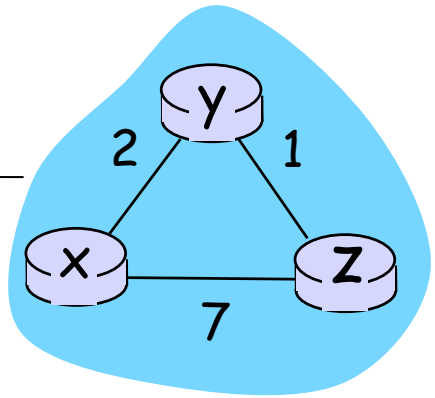
from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	7
y	**2**	**0**	**1**
z	7	1	0

cost to

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



node z table

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

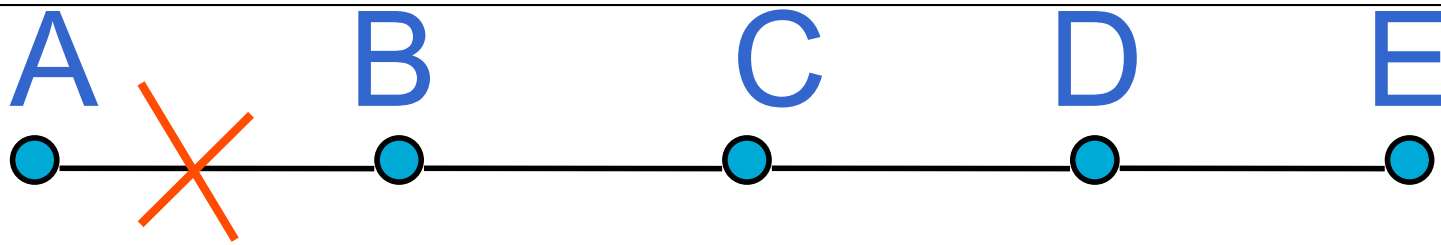
cost to

from	x	y	z
x	0	2	7
y	2	0	1
z	**3**	1	0

cost to

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

Hyvä uutinen etenee nopeasti



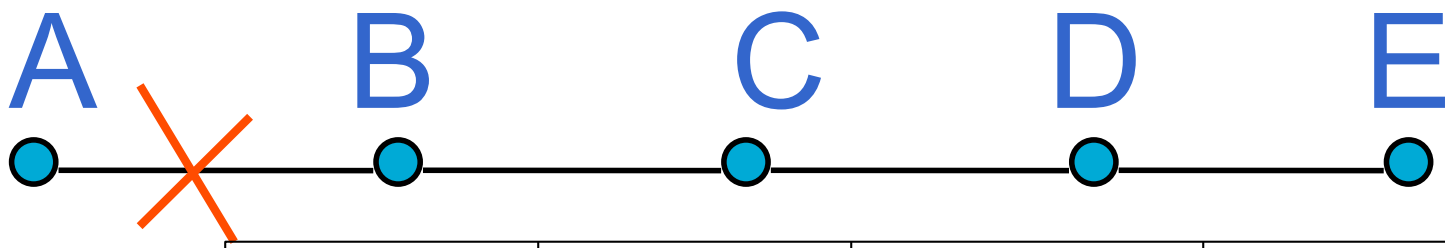
Aluksi yhteys A:han on poikki ja sitten linkki AB toimii taas

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

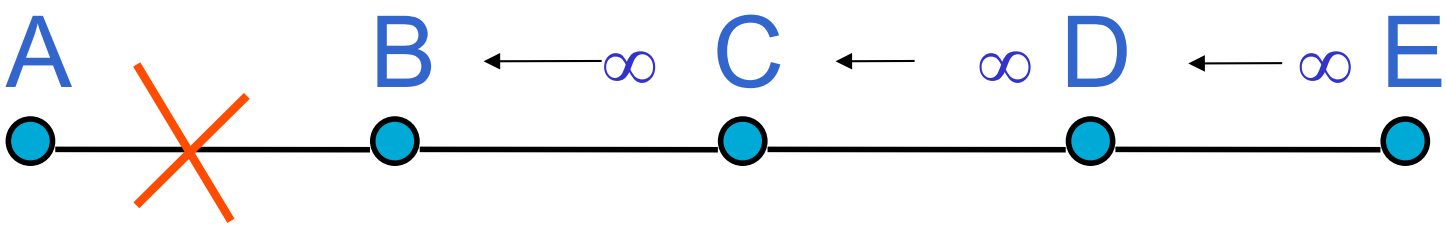
Count-to-infinity -ongelma

$D_B(A)$	$D_c(A)$	$D_D(A)$	$D_E(A)$
∞	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!

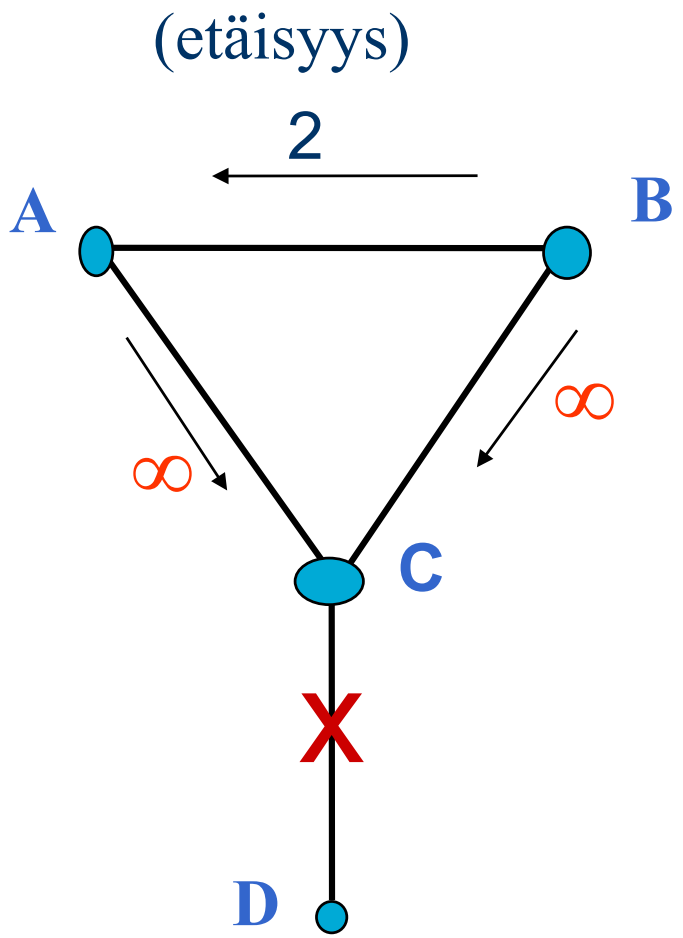
Ilmoita etäisyys äärettömäksi naapurille, jonka kautta linkki kulkee. Kerro muille oikea etäisyys.

Tieto etenee joka vaihdossa yhden linkin yli

Etäisyys A:han

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

Ratkaisu ei toimi aina!



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) = ∞

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

C kertoo B:lle, ...

Algoritmien aikavaativuus

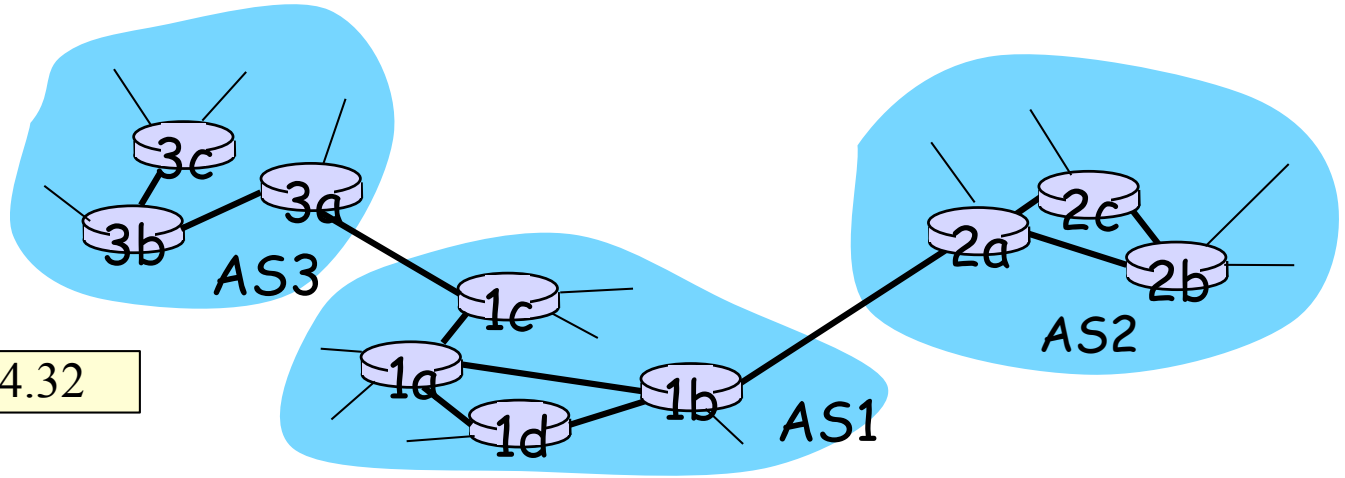
- Dijkstra
  - Naive: $O(V^2)$ Efficient with binary heap:O(E+log V), V on solmujen lukumäärä ja E kaarien lukumäärä
- Distance vector (Bellman-Ford)
  - O(E*V)

3) Hierarkkinen reititys

- Reitityksen skaalautuus?
  - Isossa verkossa runsaasti reitittimiä
    - 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 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



KuRo08: Fig 4.32

Kertauskysymyksiä

- Keskeisimmät IP-osakkeen tiedot?
- Paketin paloittelu
- Millainen on IP-osoite?
- Reitittimen arkkitehtuuri?
- Longest prefix match?
- CIDR
- NAT:n toiminta
- Miten reititin saa reititystiedot?
- Linkkitila-algoritmi, Dijkstran algoritmi
- Etäisyysvektorialgoritmi, count-to-infinity-ongelma

ks. kurssikirja s. 417