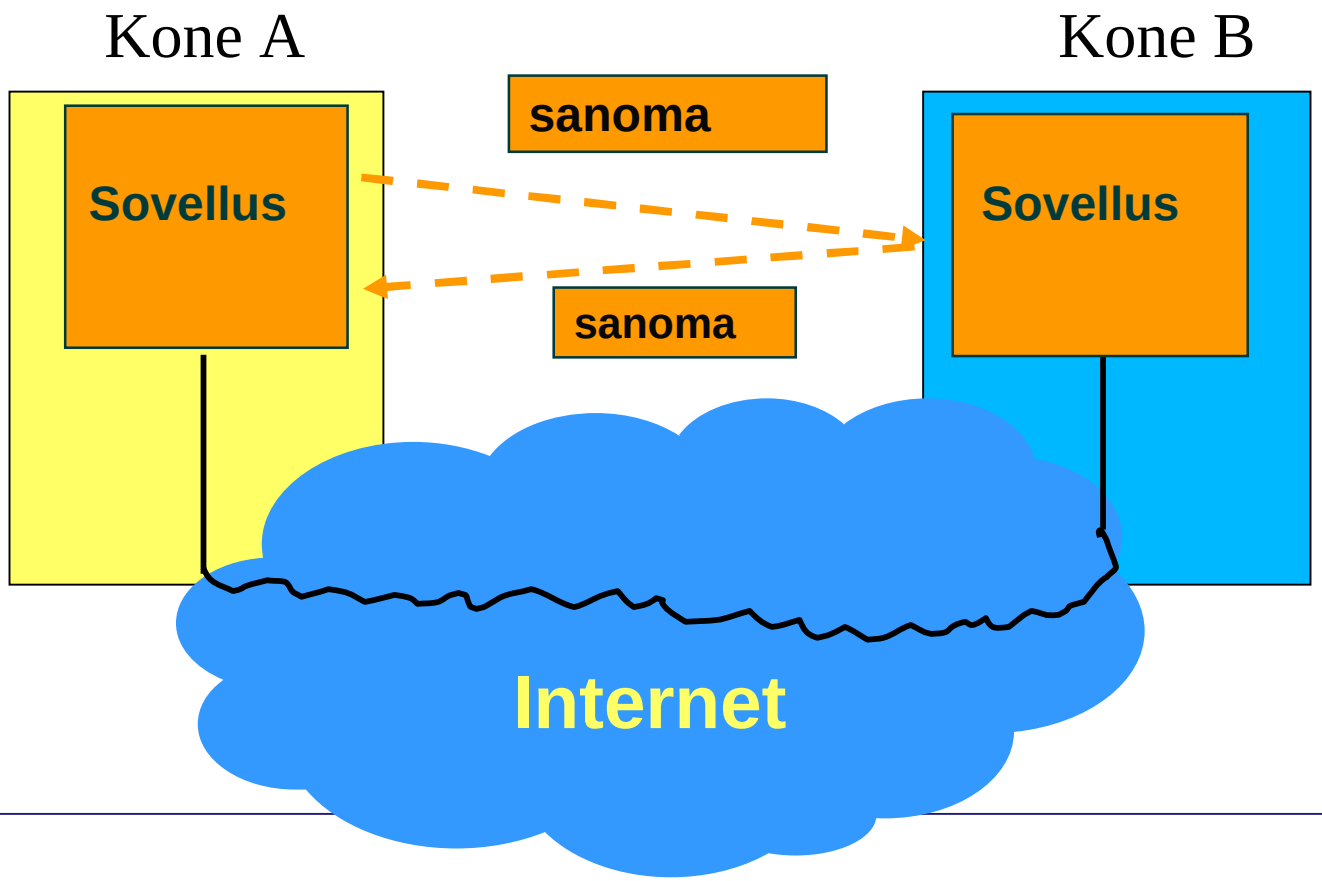
TIETOLIIKENTEEN PERUSTEET kevät 2009

Tässä on koottuna kalvot, joita käytettiin apuna kerrattaessa luentokerran alussa edellisen luentokerran pääkohtia.



Kertausta: termejä ja käsitteitä

internet – intranet - extranet
palvelu - protokolla
yhteydetön – yhteydellinen
luotettava - epäluotettava
isäntäkone - reititin - linkki
asiakas-palvelin-malli - vertaistoimijamalli
piirikytkentä - pakettikytkentä
kanavointi: taajuusjako - aikajako

siirtonopeus – siirtoaika
etenemisviive - jonotusviive

Kertaus jatkuu:

pakettivälityksen tehokkuus

sopii purskeiseen liikenteeseen

etappivälitys – sanoma paketeiksi

paketin otsake – yleisrasite

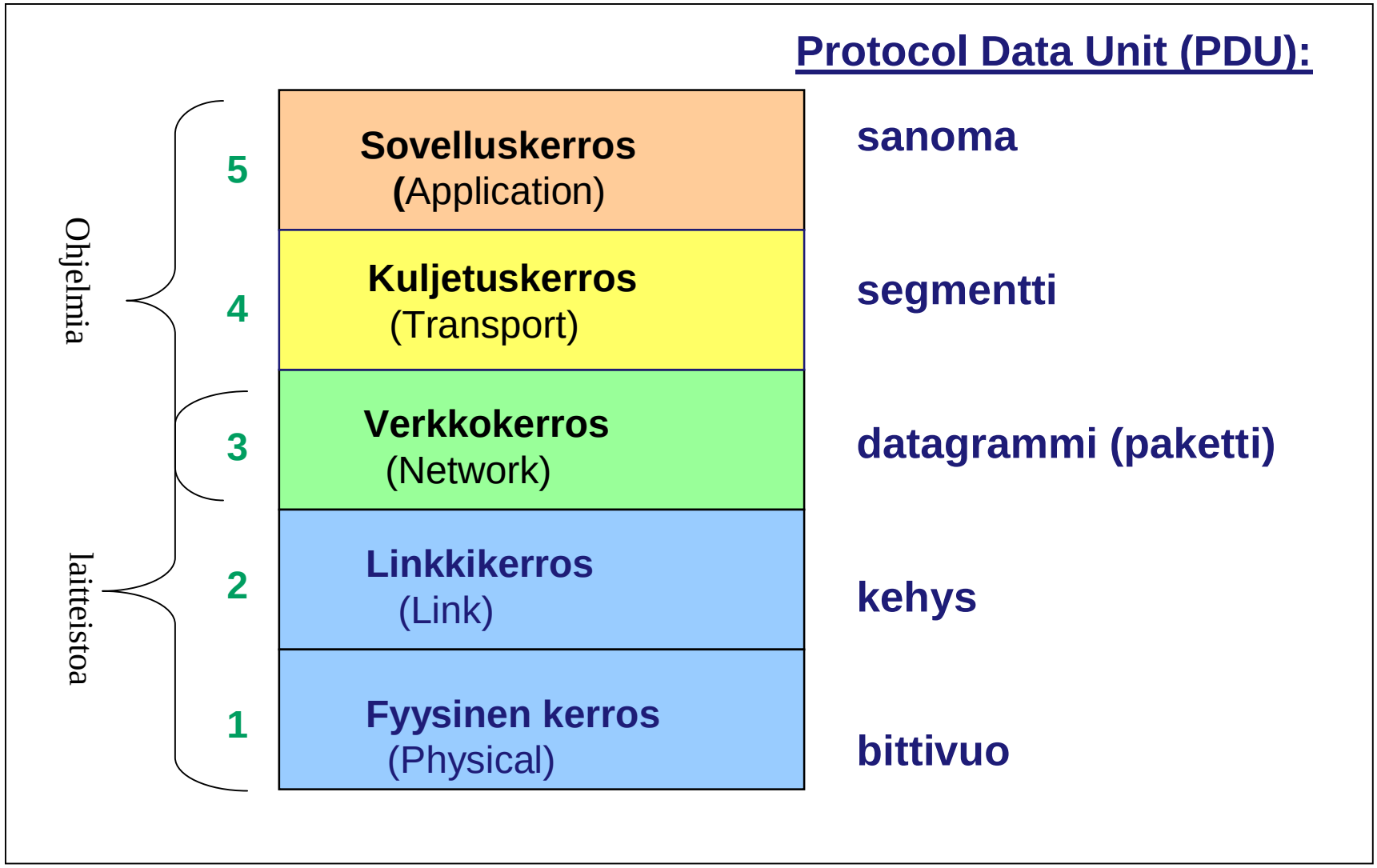
pääsy Internettiin: kaapelimodeemi, ADSL, lähiverkko, langaton yhteys

fyysinen siirtomedia: kierretty pari, koaksiaalikaapeli, valokaapeli, sähkömagneettinen aaltoliike

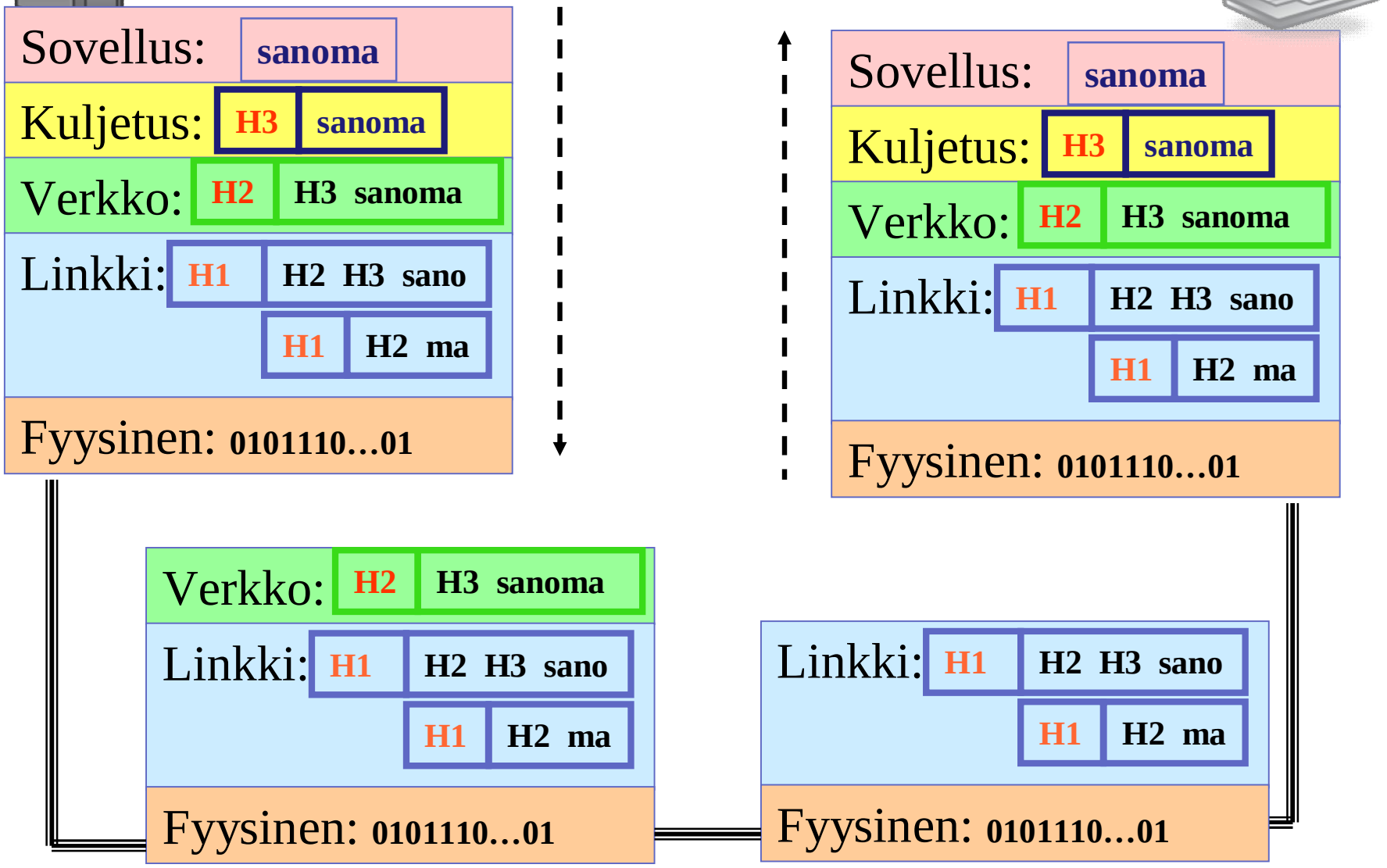
siirtovirhe: signaalin vaimeneminen ja vääristyminen, häiriöt => bittivirheitä

analoginen signaali – digitaalinen signaali

Internet-protokollapino

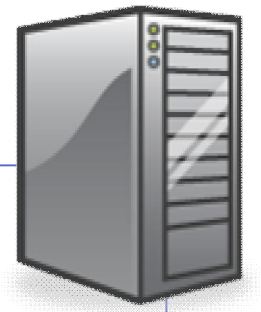


Kapselointi



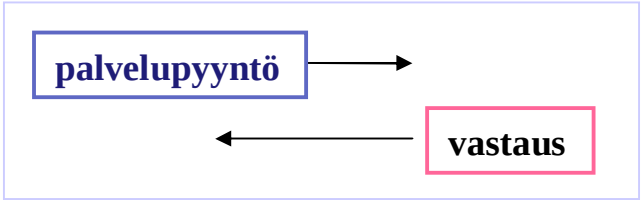
Reititin **Linkkitason kytkin**

Sovellusarkkitehtuuri



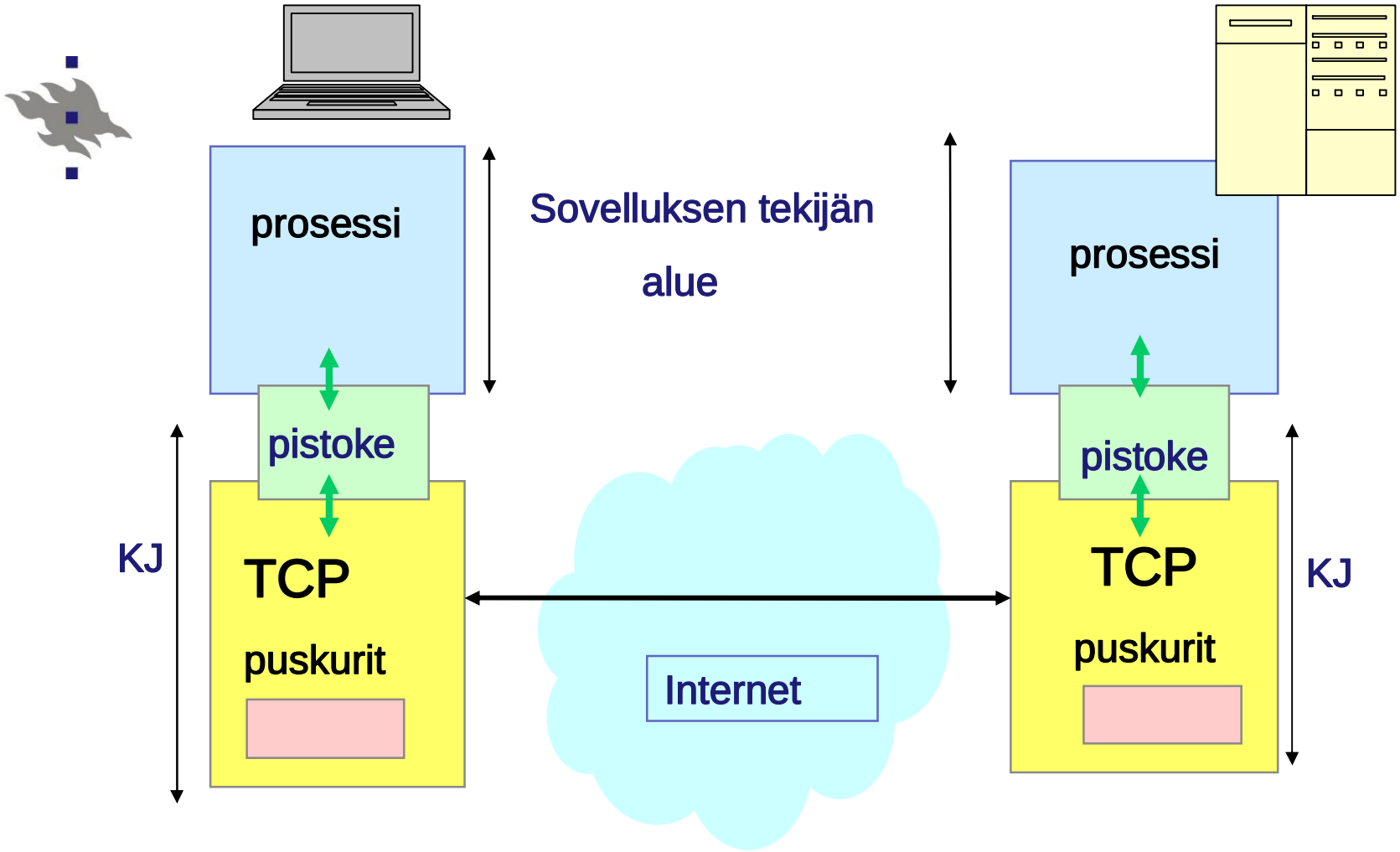
- **Asiakas-palvelija-malli** (esim. selain ja www-palvelin)
  - **Aina toiminnassa oleva palvelinohjelma, jolla kiinteä, tunnettu IP-osoite**
  - **Asiakasohjelmat ottavat yhteyttä palvelimeen ja pyytävät siltä palvelua**

Google, e-Bay, Facebook, YouTube, Amazon, ..



- **Vertaistoimijamalli** (esim. BitTorrent, eMule, Skype)
  - **Vertaisisännät kommunikoivat suoraan keskenään**
  - **Ei tarvitse olla aina toiminnassa, IP-osoite voi muuttua**
  - **Jokainen toimii sekä palvelijana että asiakkaana**

- **Hybridimalli** (esim. Napster, pikaviestimet)



KJ = käyttöjärjestelmä

Prosessien kommunikointi TCP-pistokkeita käyttäen

Osoittaminen

- **Sanomissa oltava lähettäjän ja vastaanottajan IP-osoite ja porttinumero**

- **IP-osoite à oikea kone**

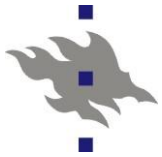
  www.iana.org

  koneen (verkkokortin) yksilöivä 32-bittinen tunniste
  osoitteen verkko-osa yksilöi verkon
  osoitteen koneosa yksilöi koneen verkossa

- **porttinumero à oikea prosessi**

  Yleisillä palveluilla standardoidut tunnetut porttinumerot:
  - www-palvelin kuuntelee porttia 80,
  - Postipalvelin kuuntelee porttia 25

  KJ osaa liittää porttinumeron prosessiin

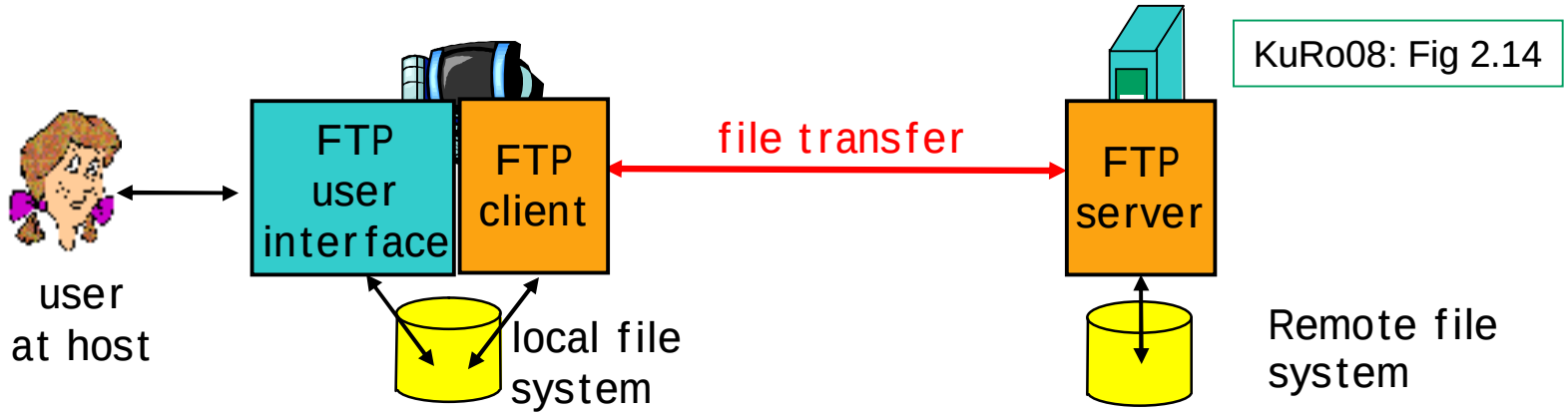


HTTP -protokolla

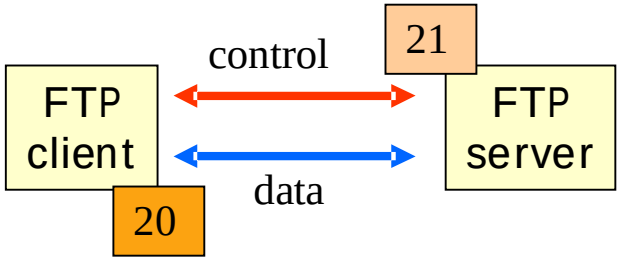
- Request –sanoma
  - GET, HEAD, POST, PUT, DELETE
  - sanoman muoto
- Response- sanoma

- Tilaton protokolla => eväste (cookie)
- Käyttää TCP:tä
  - Säilyvä yhteys
  - Liukuhihna (pipeline)
- Verkkovälimuisti (proxy server)
  - Ehdollinen GET

FTP: File Transfer Protocol (RFC 959)

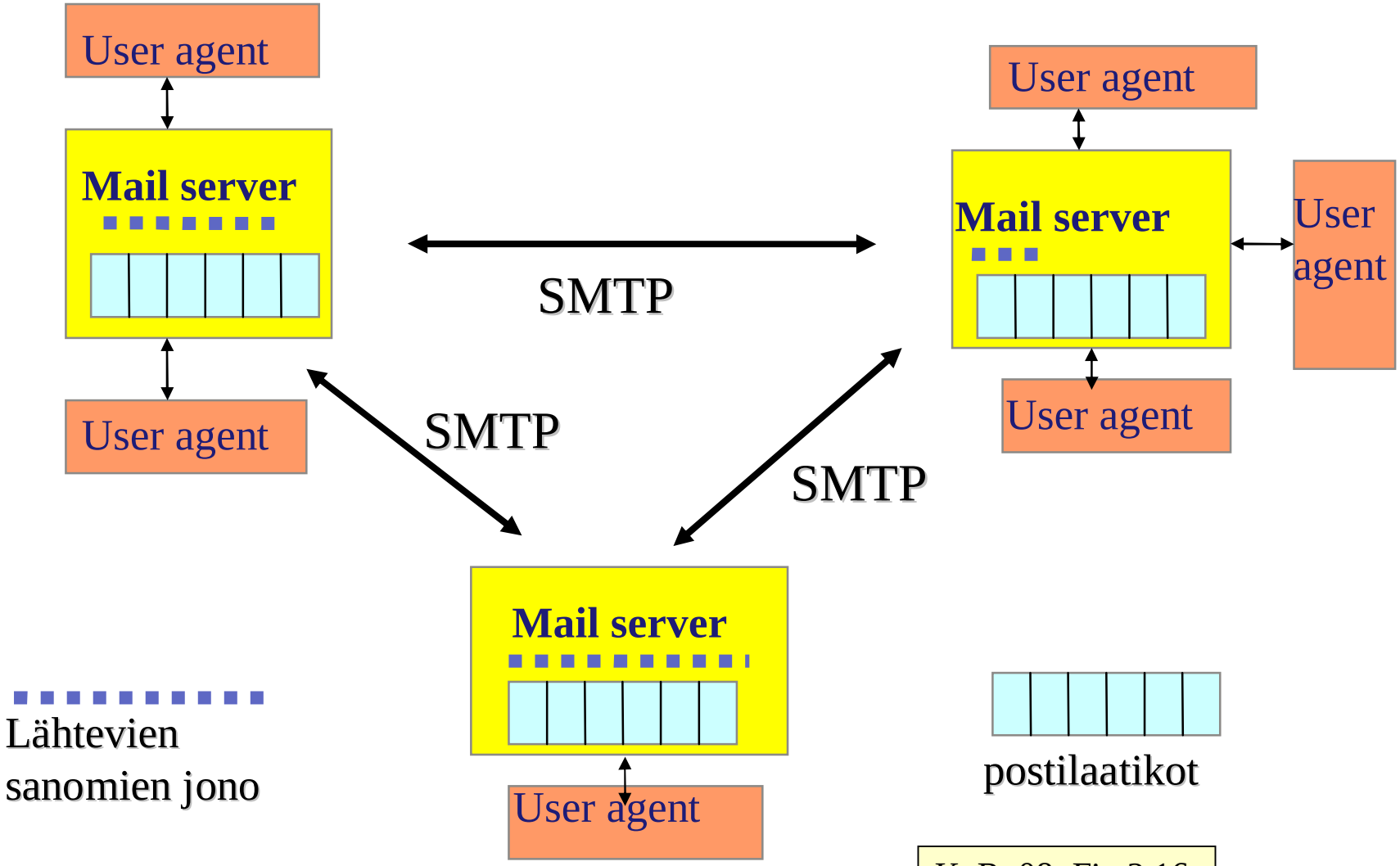


- Tiedostojen kopioiminen koneelta koneelle
  - Asiakas voi selata etäkoneen hakemistoissa FTP–sanomilla, voi noutaa tai tallettaa haluamansa tiedoston (download/upload)
- FTP-palvelin kuuntelee porttia 21
  - yhteys kontrollitiedon välitystä varten
- Asiakas kuuntelee porttia 20
  - palvelija avaa tiedoston siirtoa varten
- FTP-palvelin **ylläpitää tilatietoa**
  - mm. työhakemiston polku, autentikointi

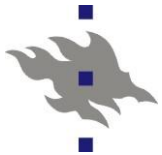


2 TCP-yhteyttä

Sähköpostin komponentit



KuRo08: Fig 2.16



Esimerkki

S: 220 helsinki.fi
C: HELO princeton.edu
S: 250 Hello princeton.edu

SMTP:n kättely

tai EHLO

C: MAIL FROM: <Bob@princeton.edu>
S: 250 <Bob@princeton.edu> OK
C: RCPT TO: <pekka.puupaa@cs.helsinki.fi>
S: 250 <pekka.puupaa@cs.helsinki.fi> OK
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: dataa … dataa
C: dataa … dataa
C: .
S: 250 Message accepted for delivery

Viesti(t)

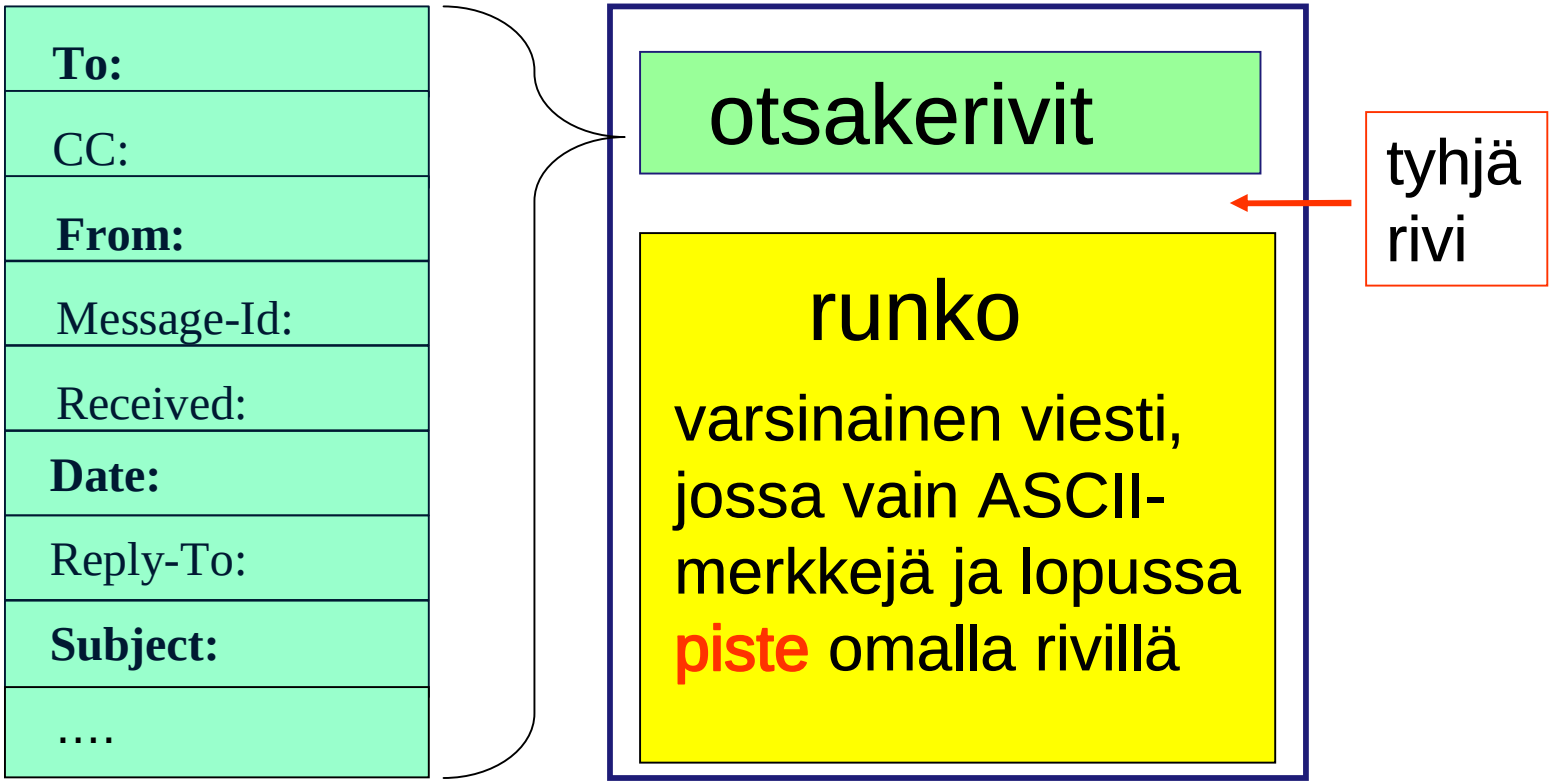
C: QUIT
S: 221 princeton.edu closing connection

SMTP:n lopetus

Sähköpostiviestin rakenne

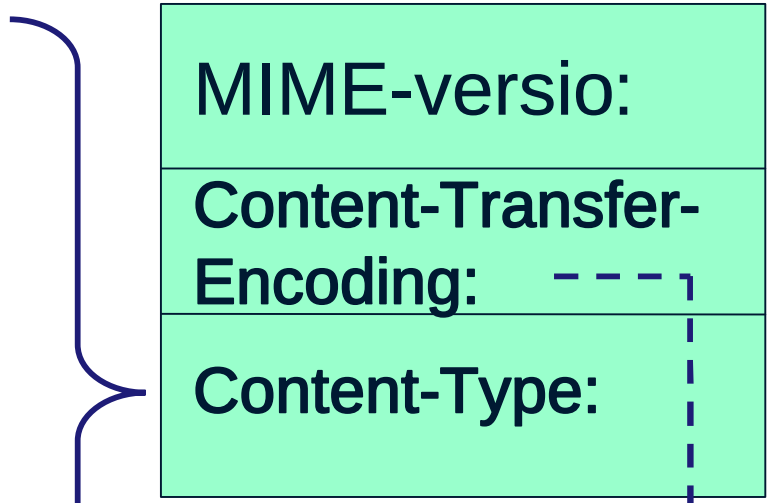
Eri asia kuin SMTP: eri standardit (RFC 822)

Esim.



MIME

- **MIME-sisältötyyppejä**
  - text/plain; charset=us-ascii
  - text/html
  - image/gif, image/jpeg,
  - video/mpeg
  - application/postscript
  - application/msword
  - application/octetstream
  - multipart/mixed



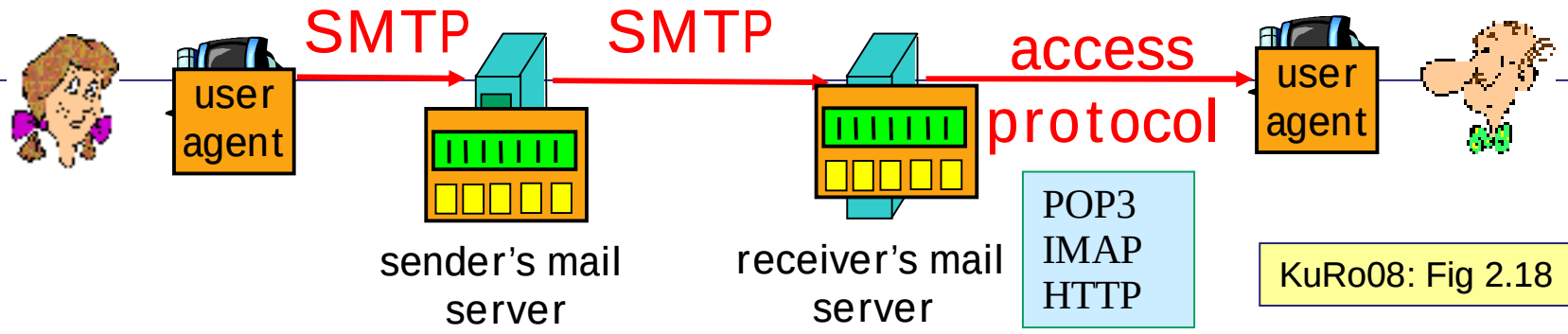
- **Base-64-koodaus**
  - Sanoma 24 bitin ryhmät on jaettu 6 bitin osiksi, jotka kukin on koodattu ASCII-merkiksi.
  - 64 eri vaihtoehtoa

Postinnoutoprotokollat (mail access protocols)

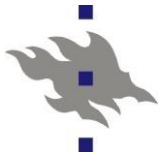
Koska SMTP on ‘PUSH’-protokolla, sitä ei voi käyttää sanomia haettaessa (‘PULL’).

Posti omalta postipalvelimelta postiohjelmaan

- **POP3:** Post Office Protocol versio 3
  Viestien lataamiseen omalle koneelle, ei postikansioita
- **IMAP:** Internet Mail Accress Protocol
  Monipuolisempi: postikansiot (folders), lataa vain otsikot, viestien säilytys postipalvelimelle
- **HTTP**: Esim. TKTL:lla käytettävä IlohaMail, Hotmail, …
  Web-palvelija käyttää IMAP-palvelijaa



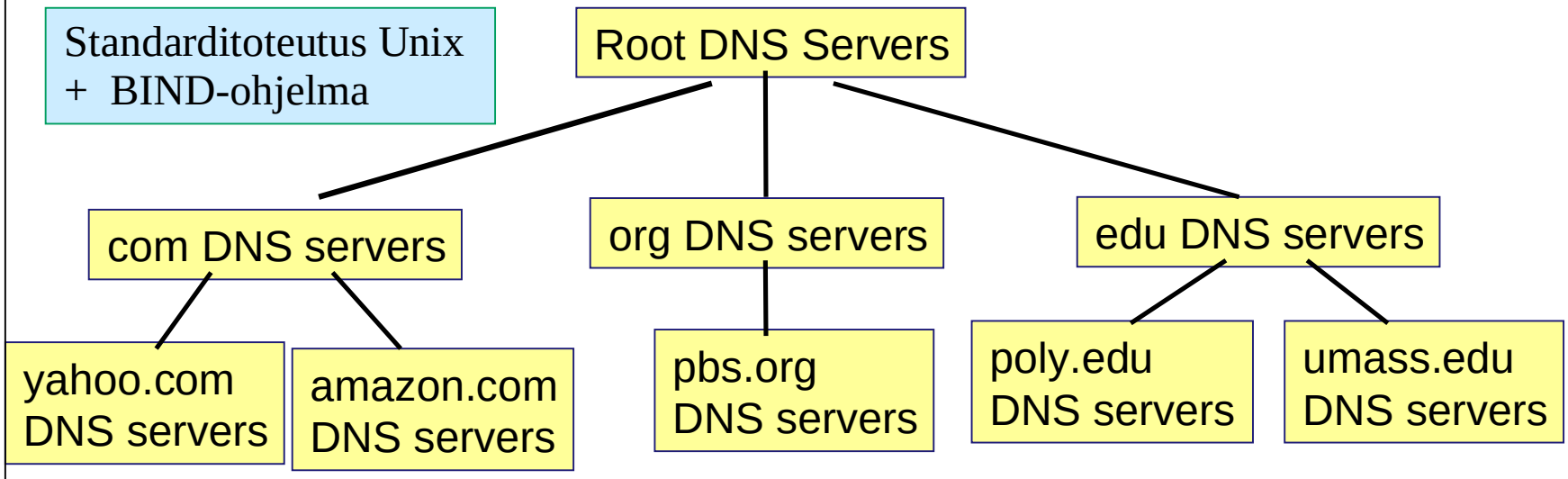
KuRo08: Fig 2.18



DNS (Domain Name System)

- **Hakemistopalvelu ja sovelluskerroksen protokolla**
  - **Isännät ja nimipalvelimet käyttävät**
  - **Käyttää itse UDP-kuljetuspalvelua DNS-sanomien kuljettamiseen**
- **Nimien muuttaminen IP-osoitteiksi (ja päinvastoin)**
  - **Posix: gethostbyname(hydra.cs.helsinki.fi) 218.214.4.29**
  - **Kone = hydra =29, verkko= cs.helsinki.fi = 218.214.4.0**
- **Sallii aliasnimet, palvelijan replikoinnin**
  - **Esim. WWW.cs.helsinki.fi ja cs.helsinki.fi ovat aliasnimiä**
  - **Esim. www-palvelijaan voi liittyä useita IP-osoitteita, rotaatio**
- **Hajautettu, hierarkinen tietokanta (hakemisto)**
  - **Toteutettu useiden replikoitujen nimipalvelimien yhteistyönä**
  - **skaalautuvuus, kuormantasaus, ylläpito, vikasietoisuus, ..**
  - **Jos oma nimipalvelija ei tunne, se kysyy muilta.**

Hajautettu, hierarkinen tietokanta



KuRo08: Fig 2.19

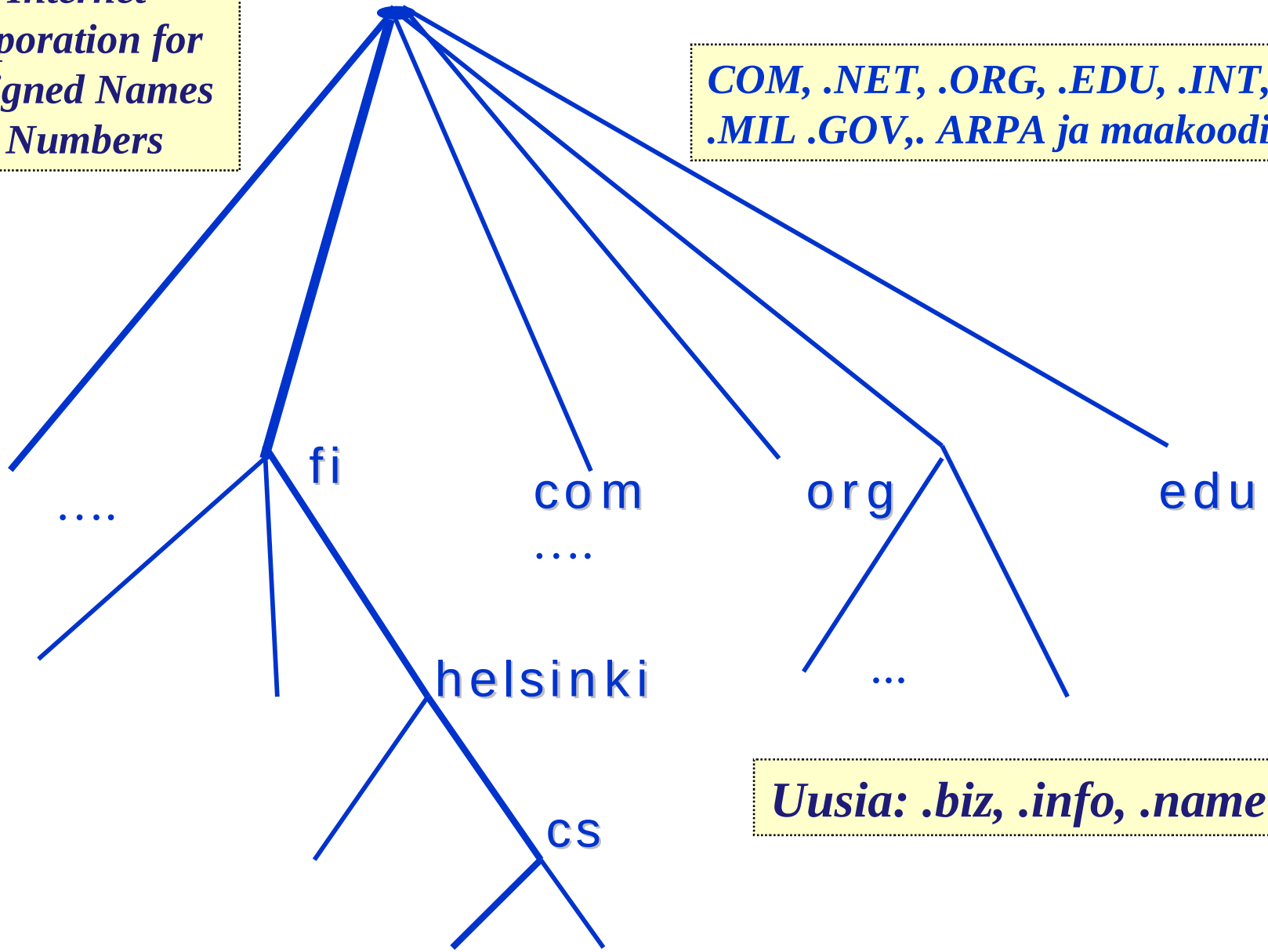
- **13 juuritason nimipalvelija**
  Replikoituja, kaikilla samat tiedot
- **Ylätason palvelimet maa- ja yleistunnuksille** (n. 265 kpl)
  …, fi, fr, uk, … edu, net, com, org, …
- **Autorisoidut aluepalvelimet** (domain) (2-taso)
  Isoilla yliopistoilla ja firmoilla omansa, pienet käyttävät jonkun muun ylläpitämää

www.iana.org

Domain -nimiavaruus

ICANN
The Internet Corporation for Assigned Names and Numbers

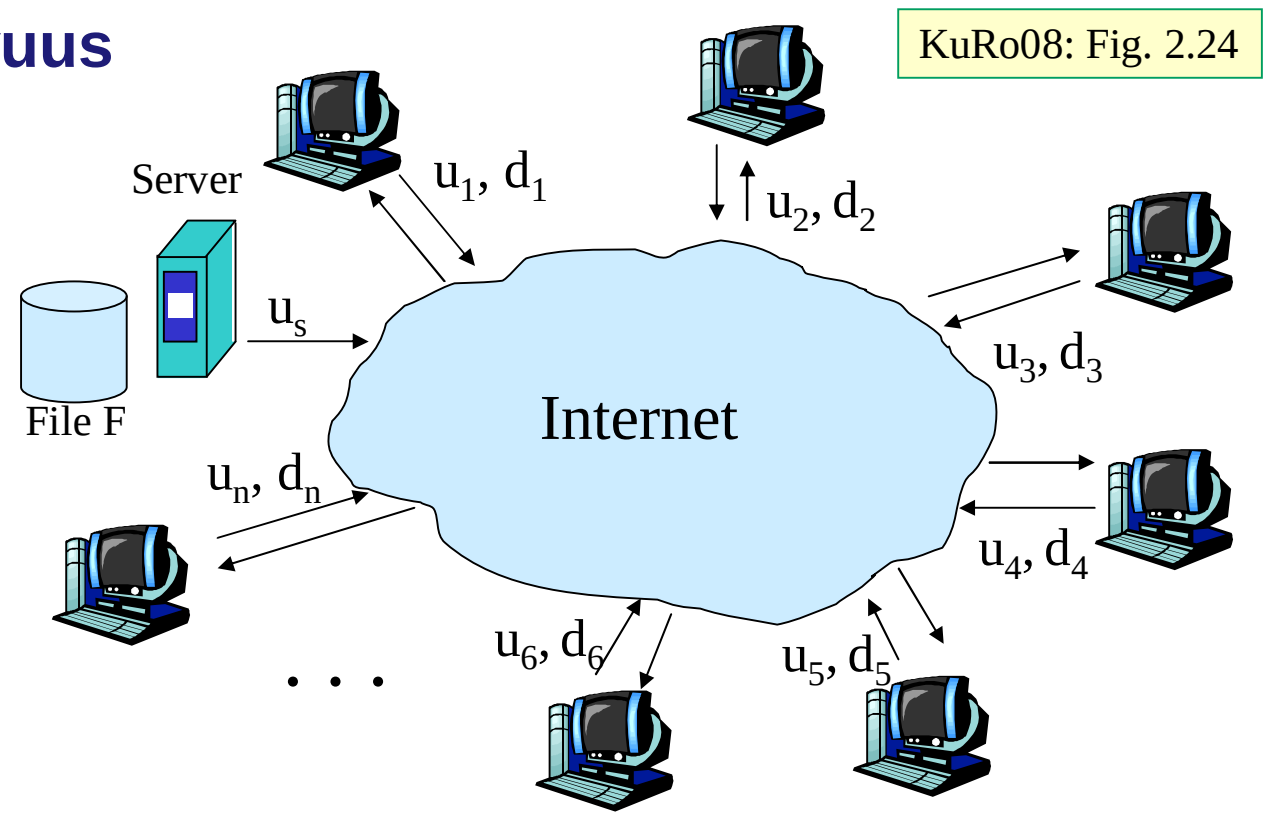
COM, .NET, .ORG, .EDU, .INT, .MIL .GOV,. ARPA ja maakoodit



Uusia: .biz, .info, .name

Skaalautuvuus

KuRo08: Fig. 2.24



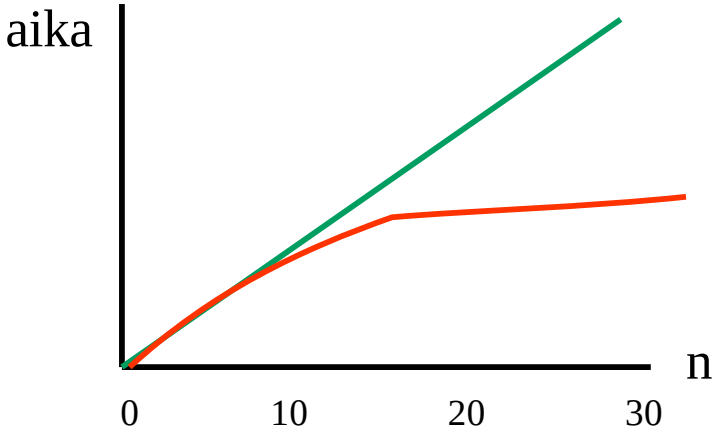
Asiakas-palvelinmalli:

Palvelimen siirrettävä n*F bittiä => siirtoaika = nF/u_s.

Hitain asiakas d_{min} saa tiedoston ajassa F/ d_{min}

Siirtoaika = max (nF/u_s, F/ d_{min})

Kun n kasvaa, palvelimen kuorma kasvaa ja siirtoaika kasvaa.

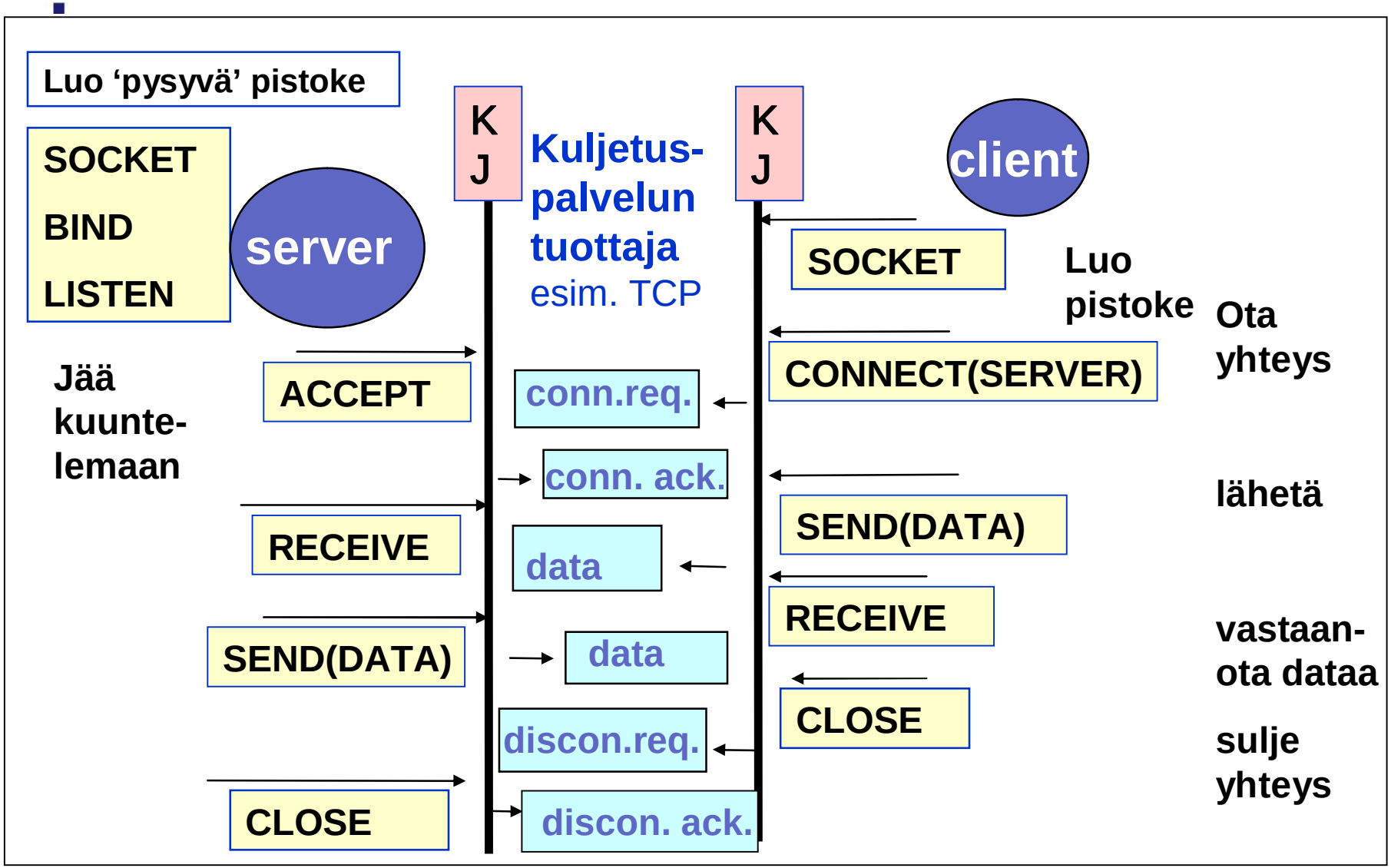


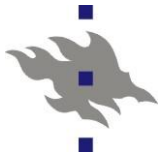
Vertaistoimijamalli (alussa tiedosto on palvelimella)

Siirtoaika = max (F/u_s, F/ d_{min}, $nF/(u_s + b\ u_i)$)

Summamerkki

TCP-kuljetuspalvelu



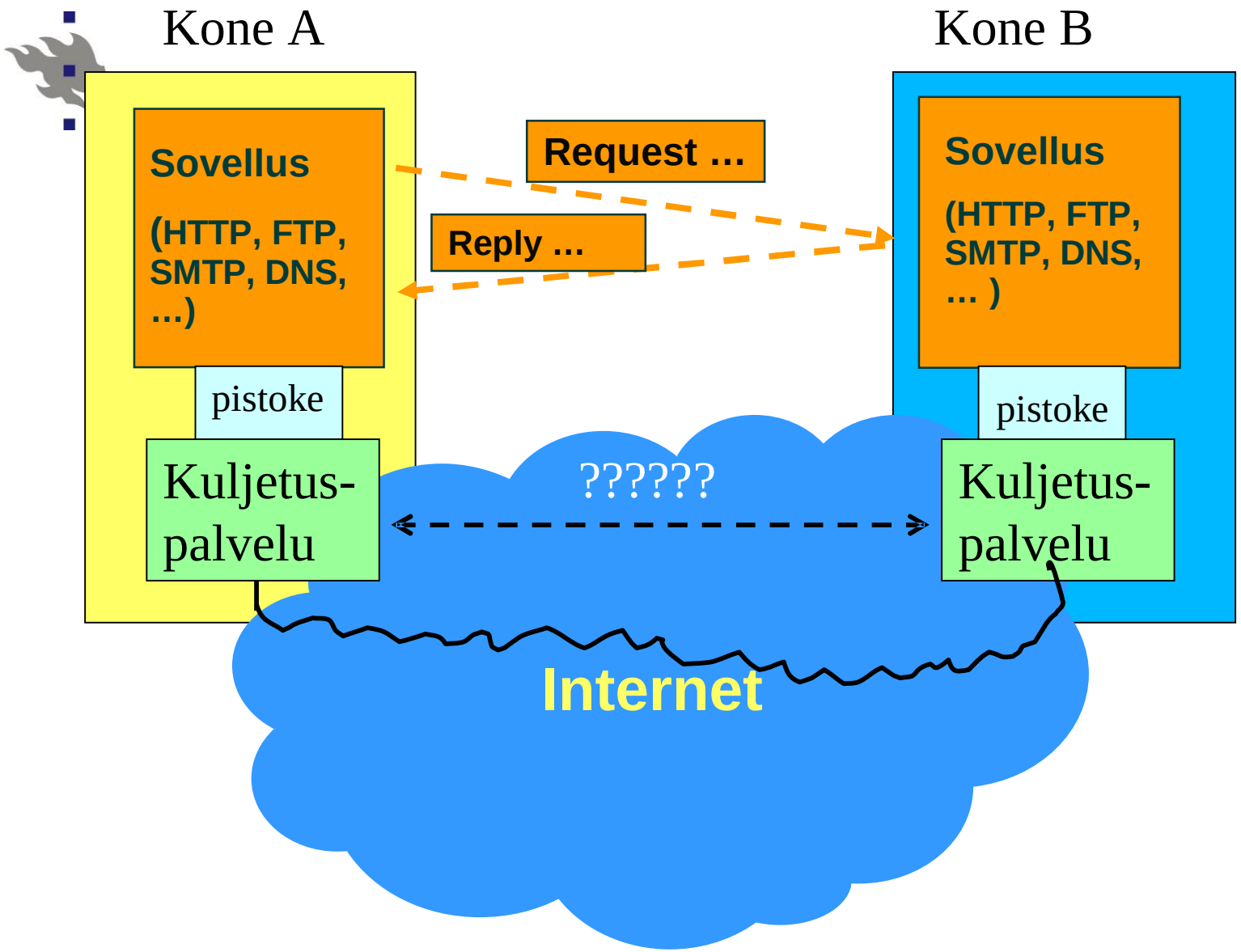


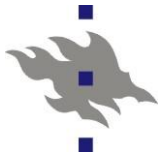
Esimerkki: TCP-asiakas (Java)

```
import java.io.*;  import java.net.*;
class TCPClient {
    public static void main(String argv[]) throws Exception {
        String sentence;
        String modifiedSentence;
        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader(System.in));
        Socket clientSocket = new Socket("hostname", 6789);
        DataOutputStream outToServer =
              new DataOutputStream(clientSocket.getOutputStream());
        BufferedReader inFromServer =
              new BufferedReader(new InputStreamReader(
                    clientSocket.getInputStream()));
        sentence = inFromUser.readLine();
        outToServer.writeBytes(sentence + '\n');
        modifiedSentence = inFromServer.readLine();
        System.out.println("FROM SERVER: " + modifiedSentence);
        clientSocket.close();
    }
}
```

yhteyspyyntö

Sulkee myös TCP-yhteyden


Kone A
Kone B
Sovellus
(HTTP, FTP, SMTP, DNS, ...)
Request ...
Reply ...
Sovellus
(HTTP, FTP, SMTP, DNS, ...)
pistoke
pistoke
Kuljetus-palvelu
??????
Kuljetus-palvelu
Internet



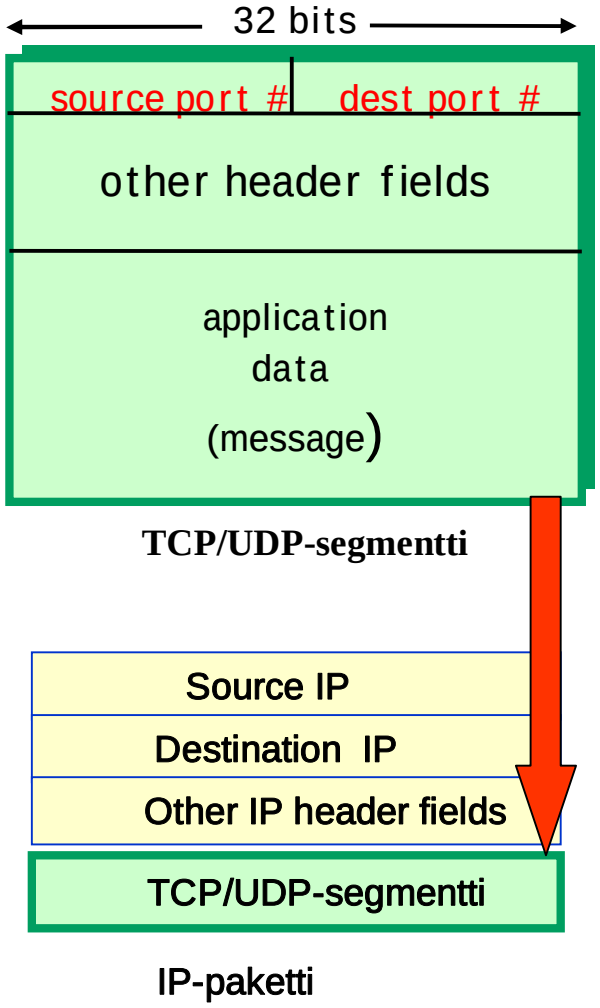
Mikä kone /Mikä prosessi?

- Kuljetuskerros tarjoaa päästä-päähän yhteyden
  - Prosessilta prosessille (= pistokkeesta pistokkeeseen)
  - Prosessi lukee ja kirjoittaa sanomia halutessaan
- Datan lisäksi on välitettävä osoitetietoja
  - Vastaanottajan ja lähettäjän tiedot
  - Eri koneiden prosessit voivat käyttää samaa palvelua
  - Saman koneen prosessit voivat käyttää eri palveluita
- Kuljetuskerros: mikä prosessi = mikä portti
- Verkkokerros: mikä kone = mikä IP-osoite
- Porttinumero
  - 16-bittinen: 0 – 65535
  - Portit 0 – 1024 on varattu kukin tietylle palvelulle (well known ports)
    - Esim. www-palvelulle portti 80, SMTP-postipalvelulle portti 25

Mikä kone /Mikä prosessi?

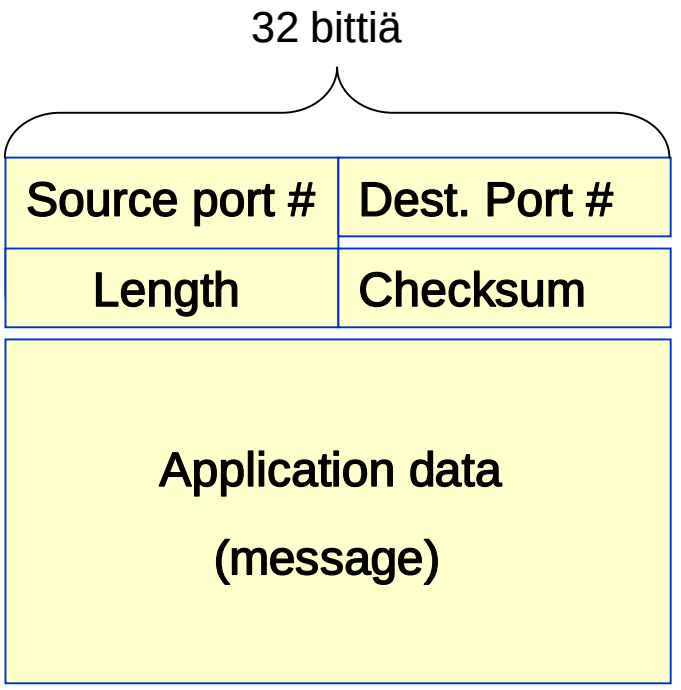
Lähetys (asiakas)

- **Kuljetuskerros**
  - **Segmentin otsakkeessa lähde- ja kohdeprosessin porttinumero**
  - **Antaa segmentin verkkokerroksen välitettäväksi**
  - **TCP: huolehtii myös luotettavuudesta**
  - **UDP: tarjoaa pelkän välityspalvelun**
- **Verkkokerros**
  - **Paketin otsakkeessa lähde- ja kohdekoneen IP-osoite à reitittimet osaavat ohjata oikealle koneelle**



UDP-segmentin rakenne

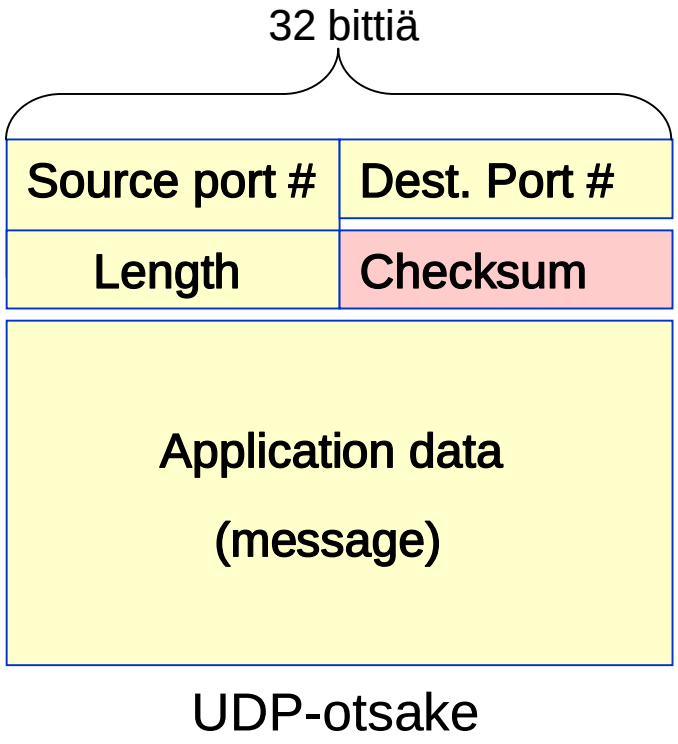
- **Porttinumerot**
  - Koska on prosessien välinen palvelu
- **Length**
  - Segmentin kokonaispituus otsake (8 B) mukaanluettuna
- **Checksum (optionaalinen)**
  - Bittivirheen havaitsemiseen
  - UDP ei yritä toipua, hävittää segmentin
- **Data**
  - Pitkä sanoma pilkottuna useaksi segmentiksi
- **IP-osoitteet vasta verkkokerroksen otsakkeessa**
  - Näitä tarvitaan **reitityksessä**



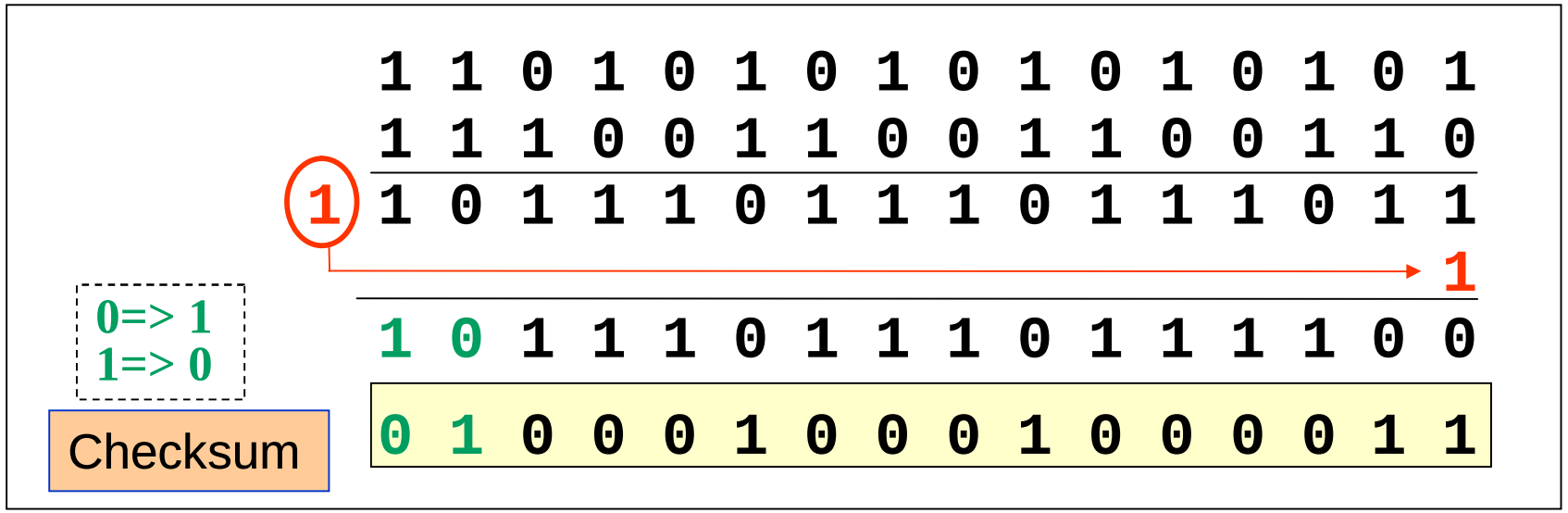
UDP-otsake

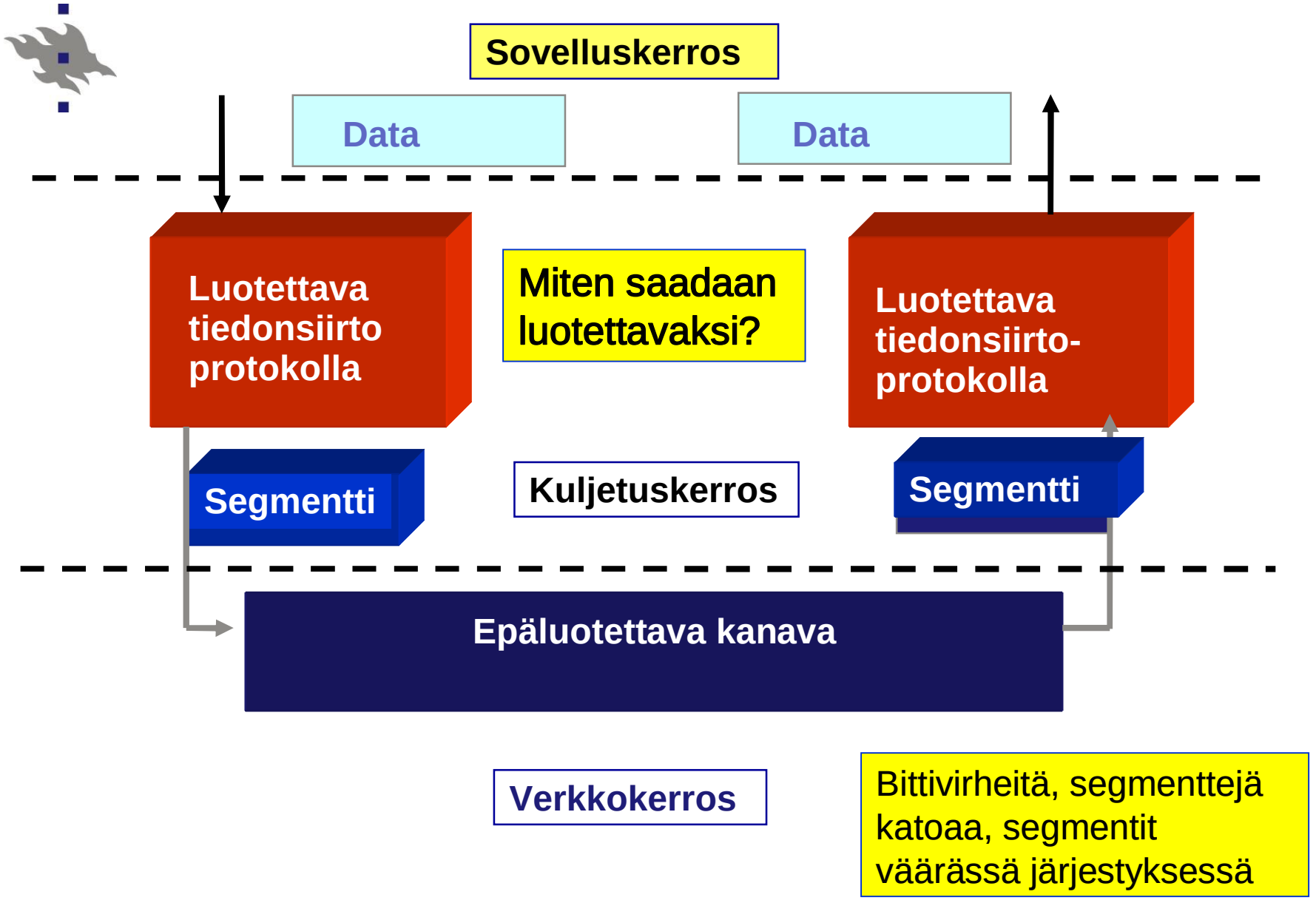
UDP-tarkistussumma

- **Lähetys**
  - **Summaa 16 bitin kokonaisuudet (otsake + pseudo-otsake mukana), ylivuotobitit lasketaan mukaan, talleta yhden komplementtina**
- **Vastaanotto**
  - **Summaa 16 b kokonaisuudet (myös tarkistussumma).**
  - **Jos tuloksena on 16 ykköstä, niin OK!**



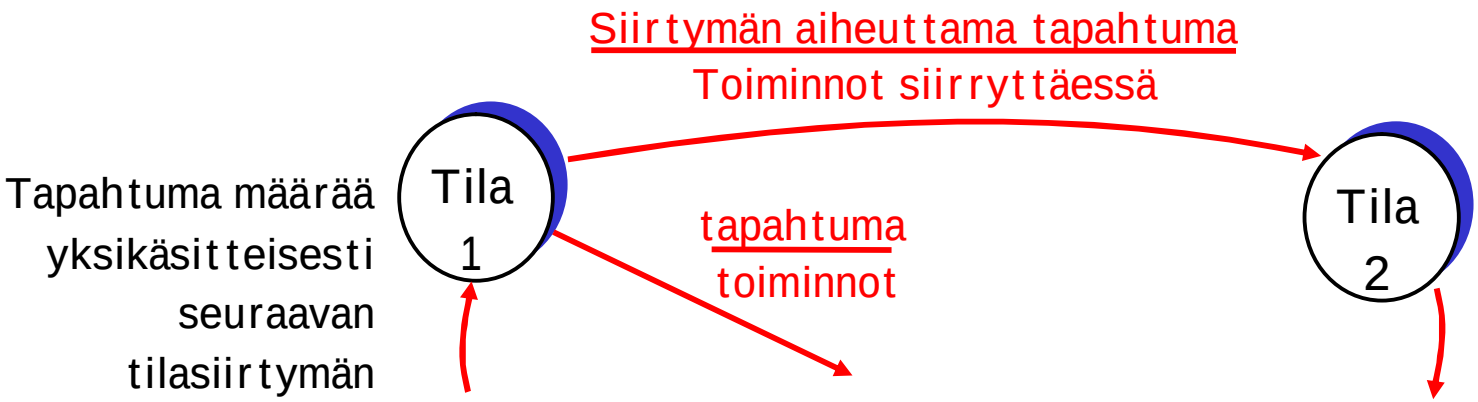
UDP-otsake



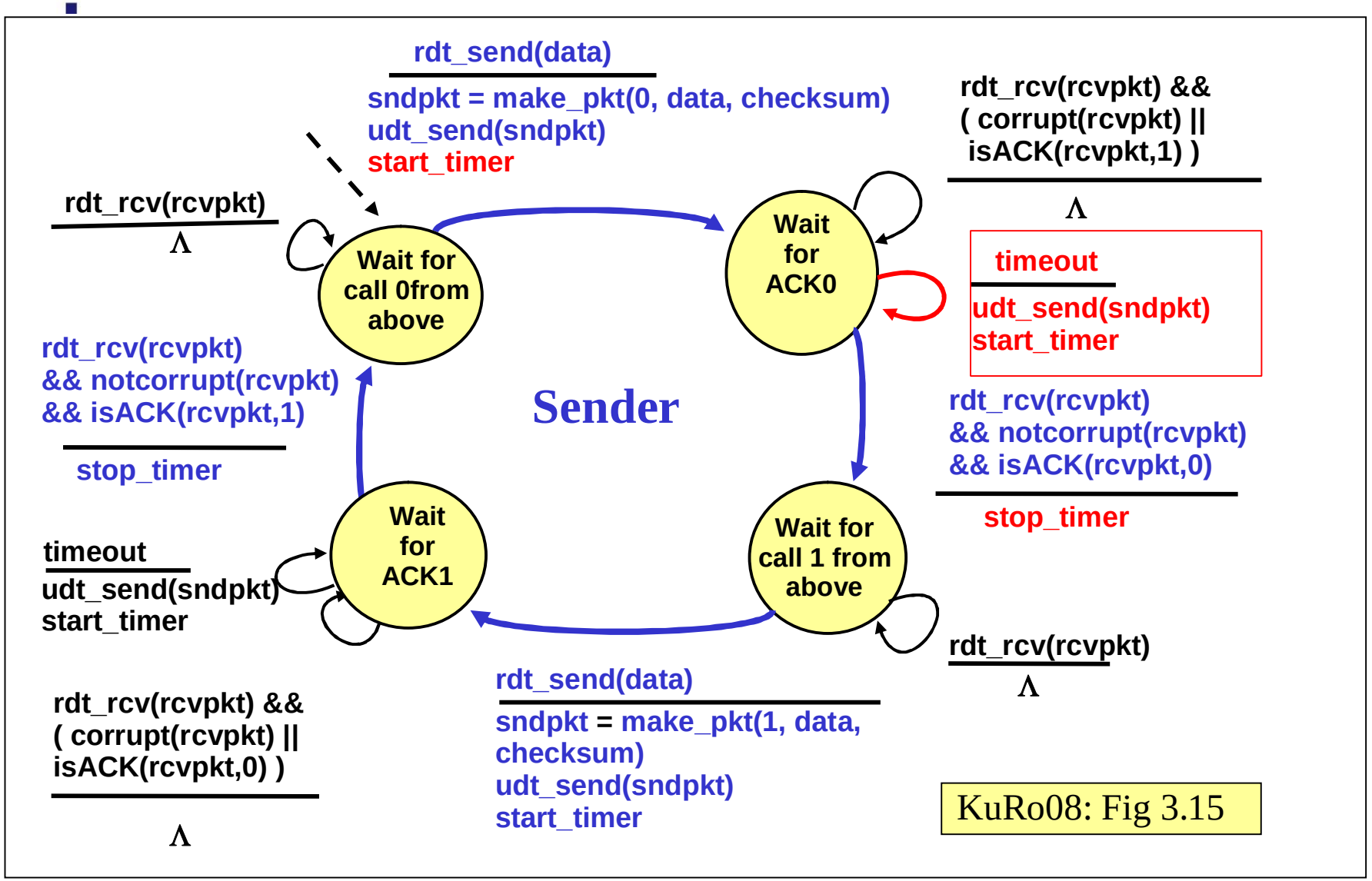

Sovelluskerros
Data
Data
Luotettava tiedonsiirto protokolla
Miten saadaan luotettavaksi?
Luotettava tiedonsiirto-protokolla
Segmentti
Kuljetuskerros
Segmentti
Epäluotettava kanava
Verkkokerros
Bittivirheitä, segmenttejä katoaa, segmentit väärässä järjestyksessä

Kuinka saada luotettavaksi?

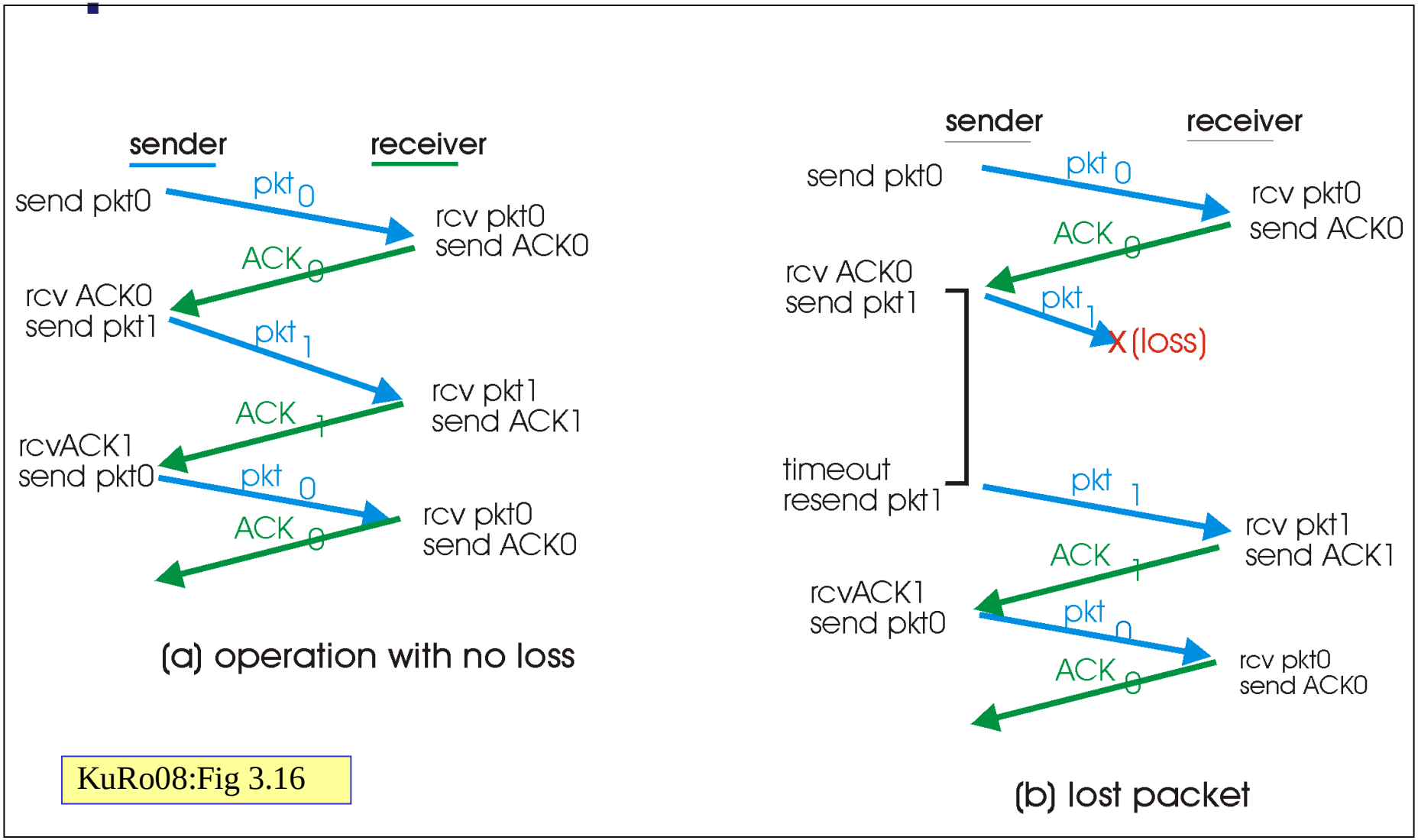
- Tarkastellaan yleisesti luotettavan tiedonsiirron ongelmia ja erilaisia ratkaisuyrityksiä
  - Edeten ideaalitilanteesta yhä ongelmaisempaan
  - Käyttäen äärellisiä tila-automaatteja lähettäjän ja vastaanottajan toiminnan kuvaamiseksi



rdt3.0

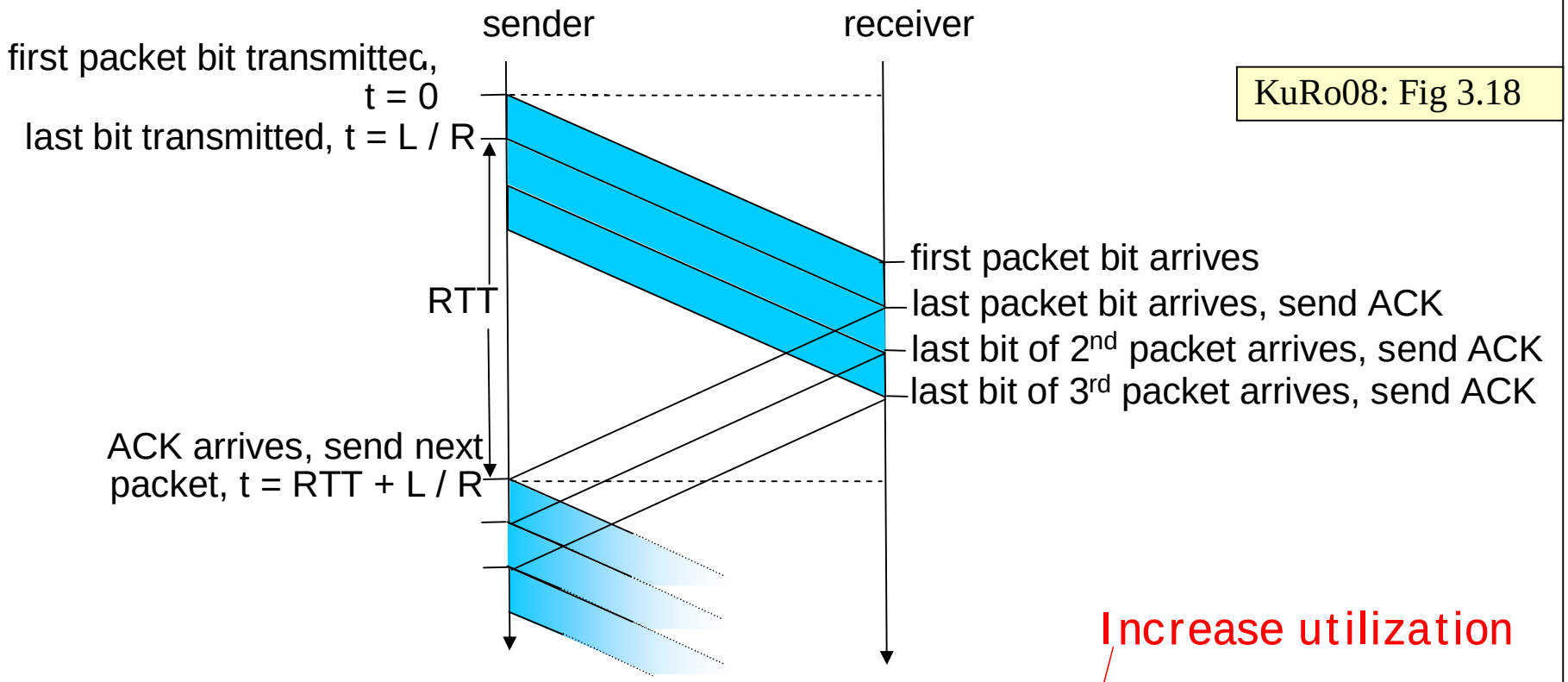

rdt_send(data)
sndpkt = make_pkt(0, data, checksum)
udt_send(sndpkt)
start_timer
rdt_rcv(rcvpkt) &&
(corrupt(rcvpkt) ||
isACK(rcvpkt,1))
Λ
rdt_rcv(rcvpkt)
Λ
Wait for call 0from above
Wait for ACK0
timeout
udt_send(sndpkt)
start_timer
rdt_rcv(rcvpkt)
&& notcorrupt(rcvpkt)
&& isACK(rcvpkt,1)
stop_timer
Sender
rdt_rcv(rcvpkt)
&& notcorrupt(rcvpkt)
&& isACK(rcvpkt,0)
stop_timer
Wait for ACK1
Wait for call 1 from above
timeout
udt_send(sndpkt)
start_timer
rdt_rcv(rcvpkt)
Λ
rdt_rcv(rcvpkt) &&
(corrupt(rcvpkt) ||
isACK(rcvpkt,0))
Λ
rdt_send(data)
sndpkt = make_pkt(1, data, checksum)
udt_send(sndpkt)
start_timer
KuRo08: Fig 3.15

rdt3.0 toiminnassa



KuRo08:Fig 3.16

Liukuhihnoitus: käyttöasteen kasvattaminen

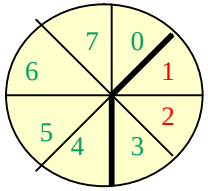


KuRo08: Fig 3.18

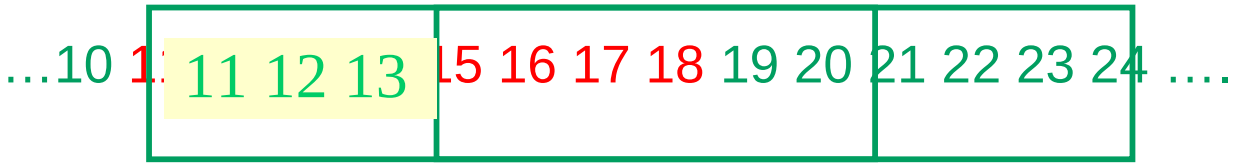
Increase utilization by a factor of 3!

$$U_{sender} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008 \quad (0.00028)$$

Liukuva ikkuna (sliding window)



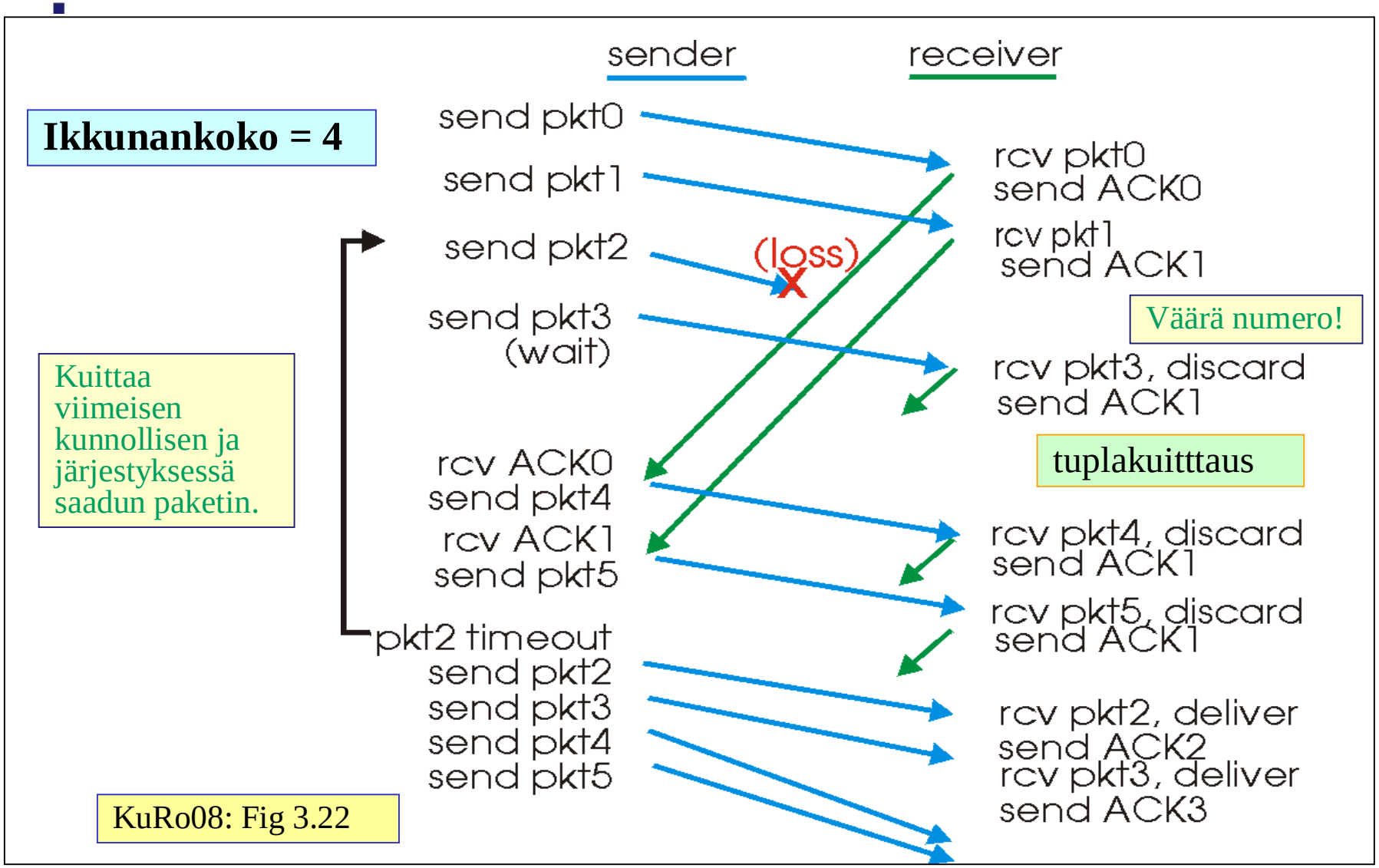
- **Säätelee pakettien lähettämistä ja vastaanottoa**
  - **Kertoo millä pakettinumeroilla on lähetetty/vastaanotettu, mistä saatu/lähetetty kuittaukset ja millä numeroilla voi vielä lähettää/vastaanottaa paketteja**
  - **Ikkunan koko** riippuu yhteyden tyypistä ja puskurien koosta



- **Lähetysikkuna** (sender window)
  - Ikkunan koko = montako pakettia saa olla kuittaamatta
  - Mitkä pakettinumerot on käytetty, mutta kuittaamatta
  - Mitä pakettinumeroita voi vielä käyttää
- Lähettäjän on odotettava, jos kaikki ikkunan numerot on käytetty
- Kun kuittaus saapuu, ikkuna liukuu
  - Seuraavat numerot tulevat luvallisiksi

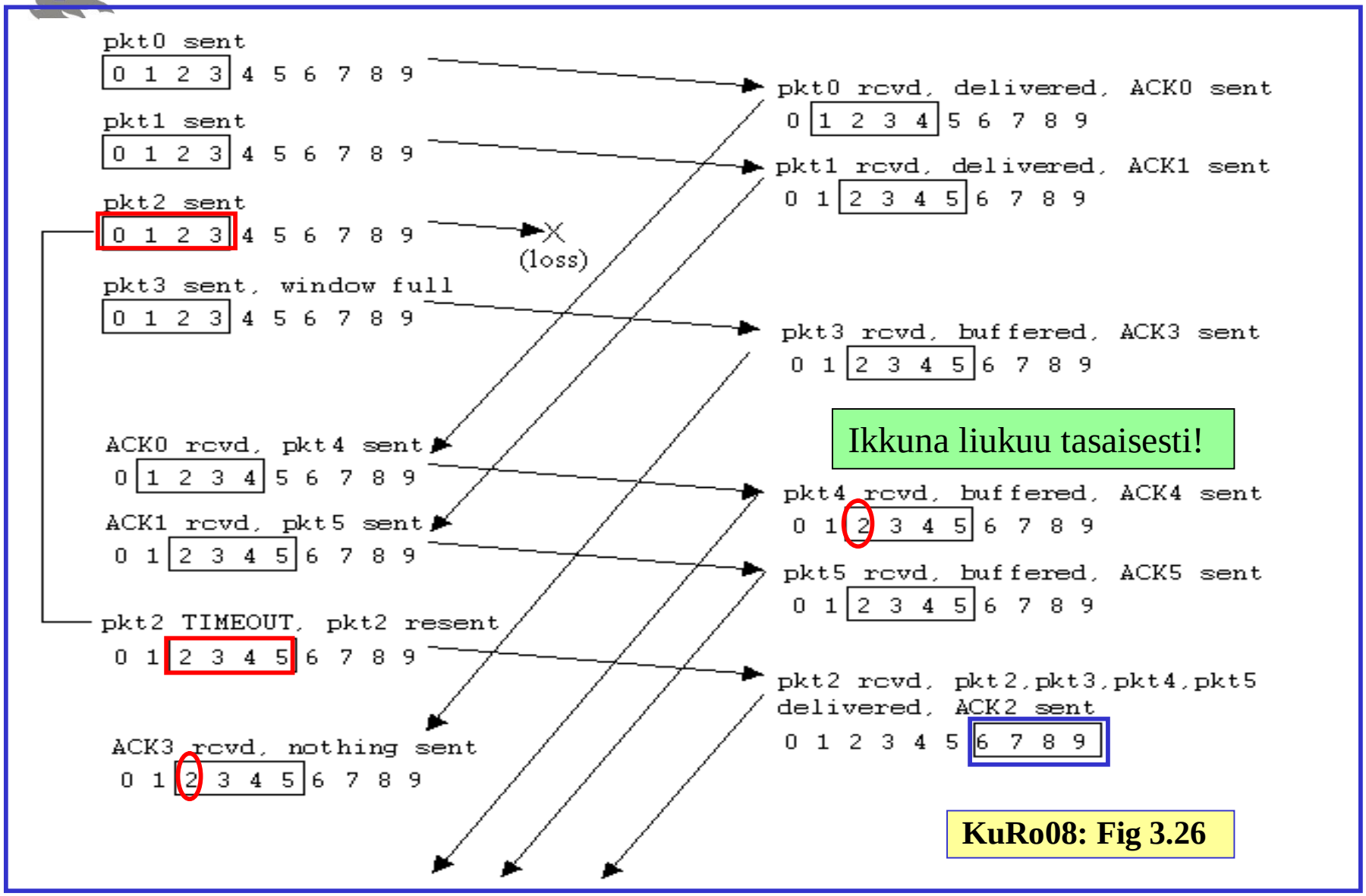
Go-Back-N: Esimerkki

Kumulatiivinen kuittaus



Esimerkki: Valikoiva toisto

Jokainen sanoma kuitattava erikseen



Ikkuna liukuu tasaisesti!

KuRo08: Fig 3.26

Yhteenveto menetelmistä

- Ks. KuRo08 Table 3.1

- Tarkistussumma
- Ajastin
- Järjestysnumero
- Kuittaukset
  - Positiiviset ACK, tuplakuittaukset
  - Negatiiviset NAK
- Ikkunat, liukuhihnoitus

Yhteyden muodostus

syn-segmentti on yhden tavun kokoinen!

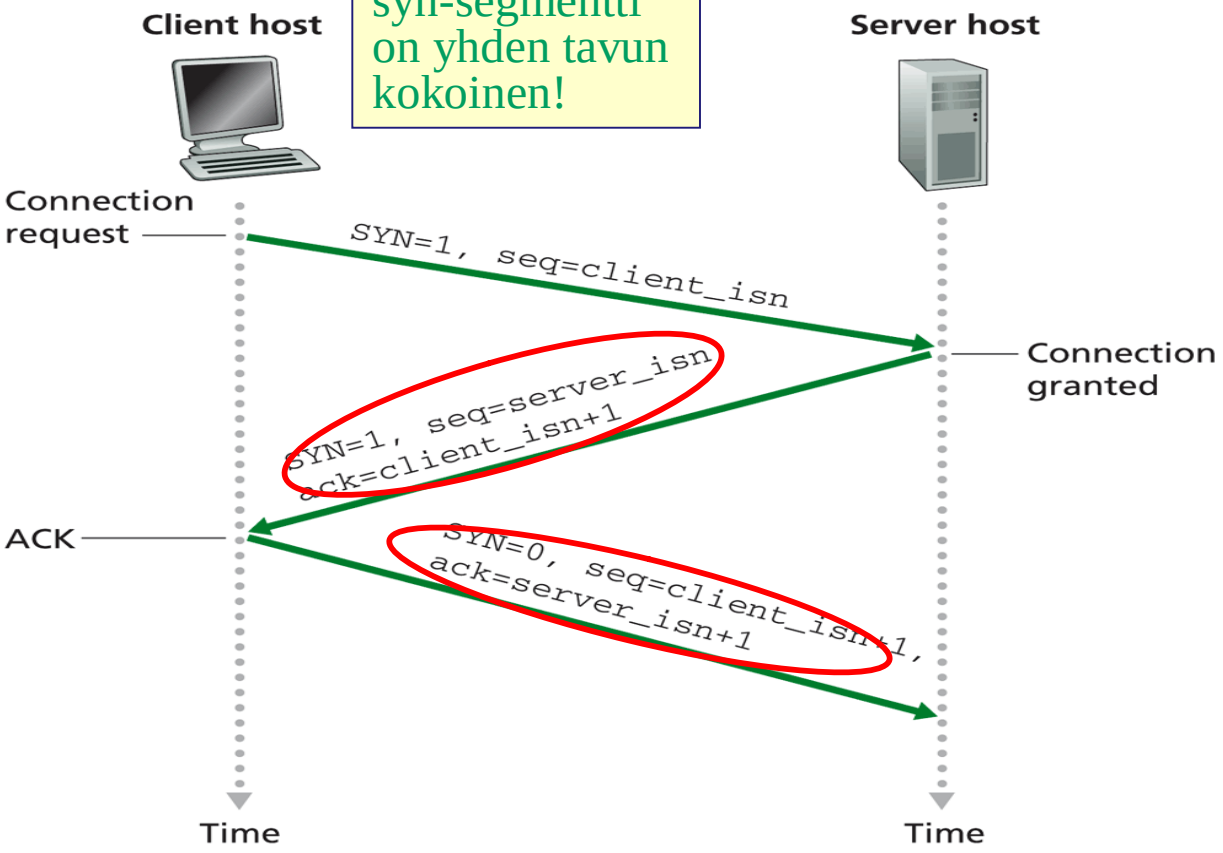
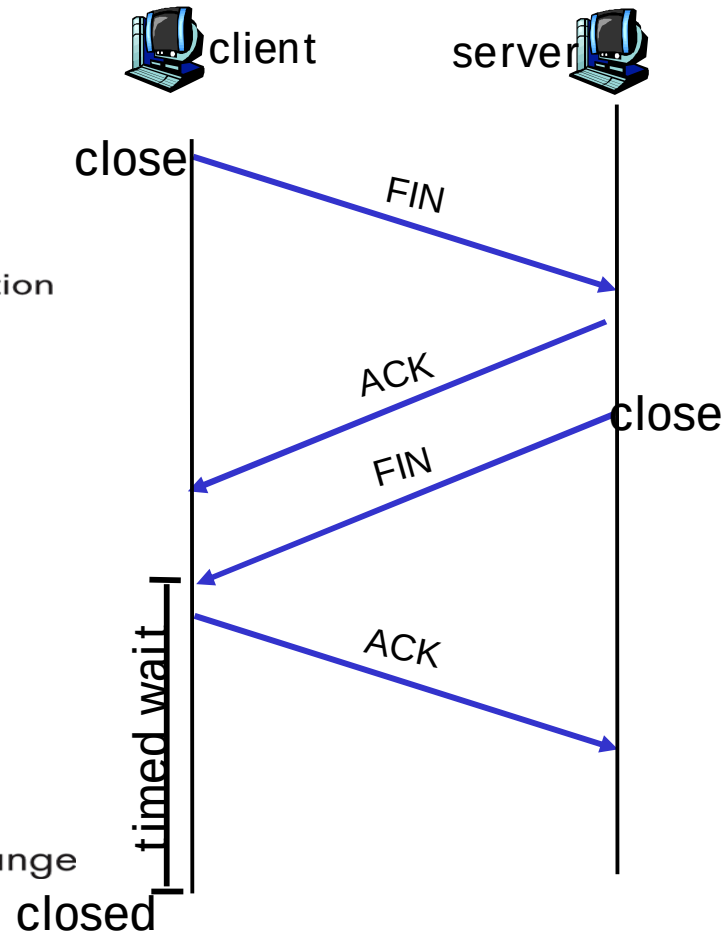
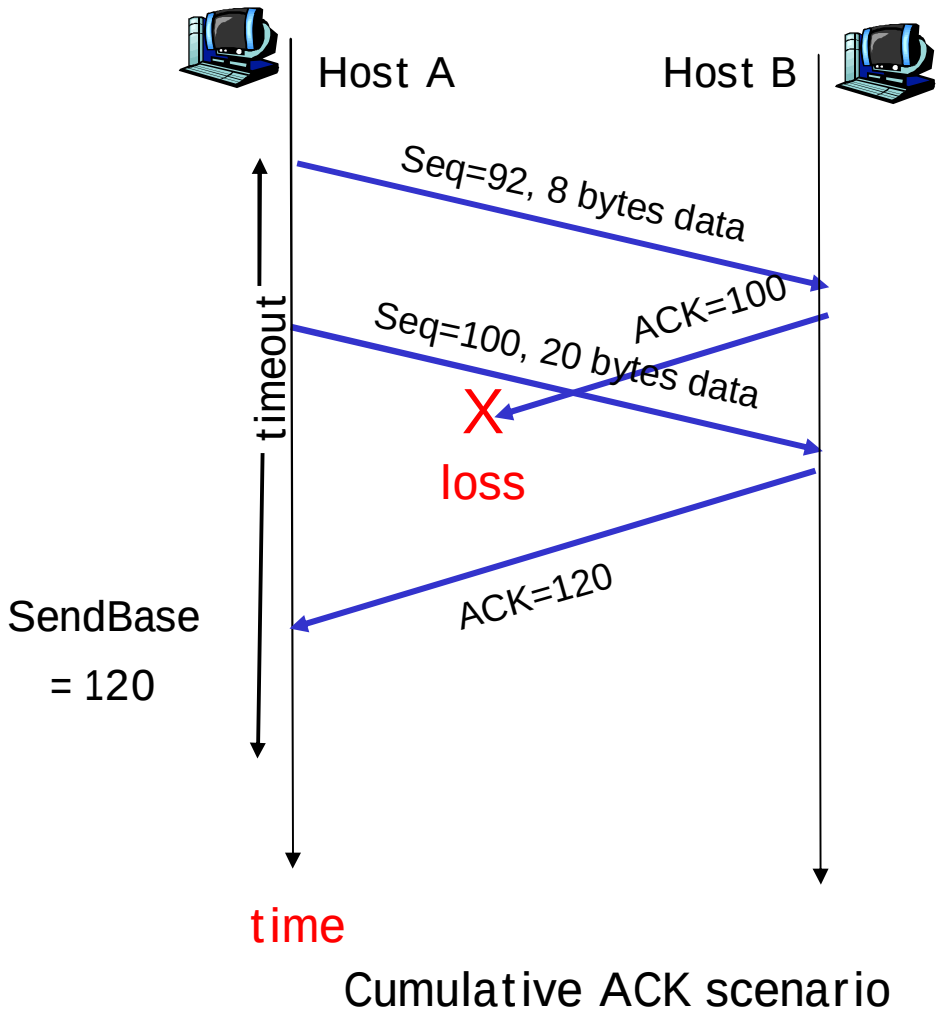


Figure 3.38 ♦ 3.39 three-way handshake: segment exchange

Yhteyden purku



TCP: uudelleenlähetys (2)



Cumulative ACK scenario

KuRo08: Fig 3.36

- Checksum
- Tavunumerointi
- Kumuloiva kuittaus
  Toistokuittaus
- Ajastin
- Uudelleenlähetys
- Nopea uudelleenlähetys

Tavunumerointi

Kuittauksia ei kuitata ja ne eivät siirrä numerointia!

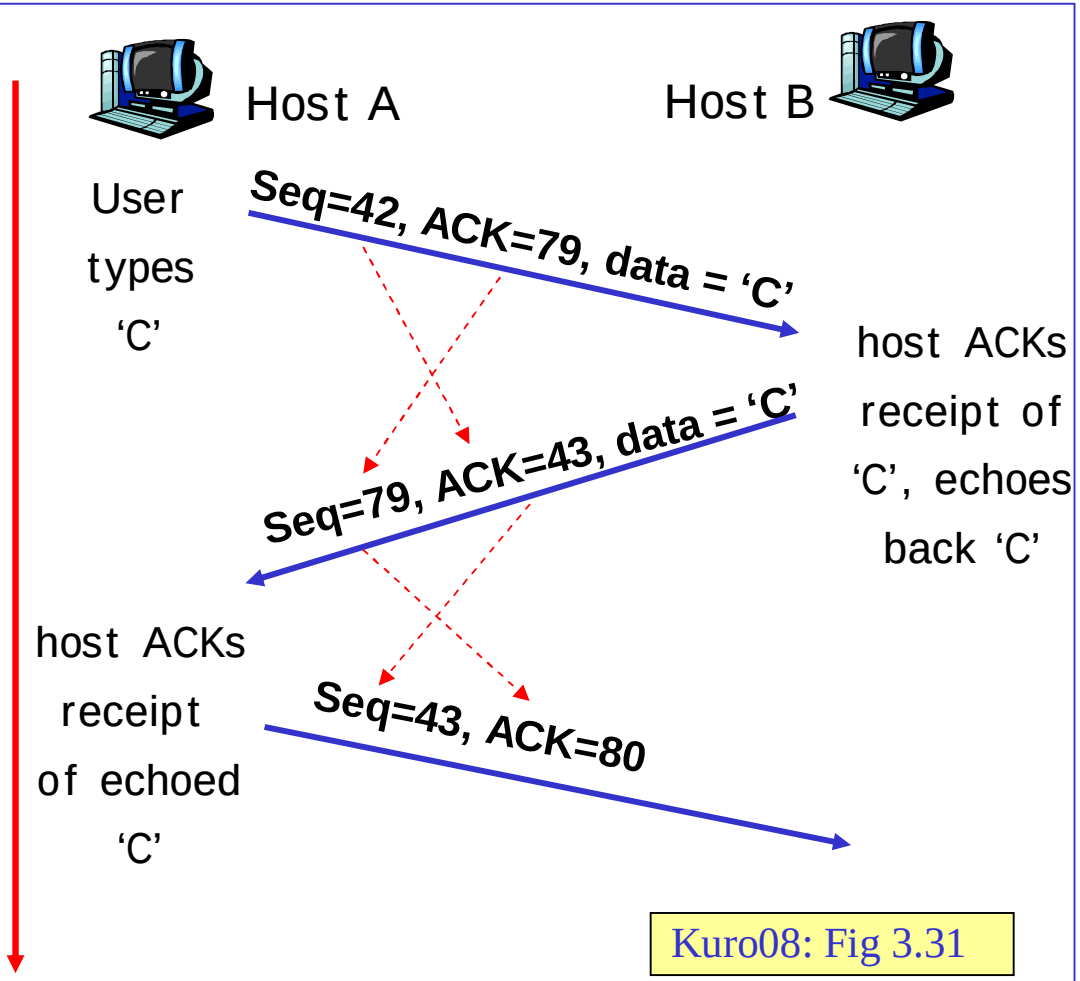
Niissä ei siirretä tavuvirtaa.

Tavuvirtaa …

Segmentit voivat olla erikokoisia (=< MSS)

Segmentin 'numero'=
- ensimmäisen tavun numero
- alkuarvot sovitaan yhteyttä muodostettaessa

Kuittaus
- seuraavaksi odotetun tavun numero
- kumulatiivinen
- kylkiäisenä (piggybacked) mikäli mahdollista

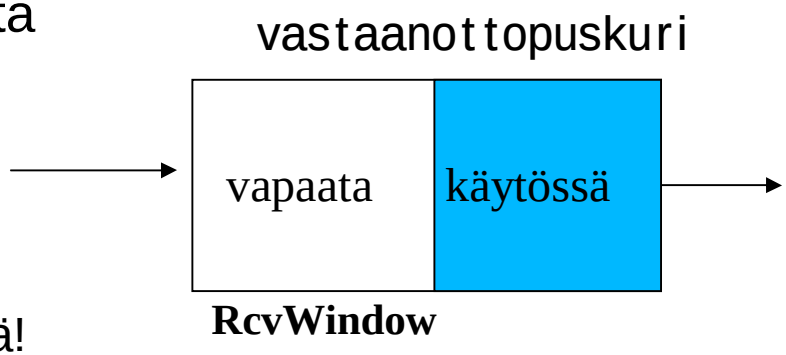


Kuro08: Fig 3.31

simple telnet scenario

Vuonvalvonta

- Jotta lähettäjä ei tukahduta vastaanottoa
  - Siirtonopeus sovitettava vastaanottavan sovelluksen mukaan
- Kuittaus on irroitettu vuonvalvonnasta
- **Liukuva ikkuna, koko vaihtelee**
  - Kuittaus siirtää ikkunaa
  - Vuonvalvonta määrää ikkunankoon
  - Kun ikkunankoko = 0, ei saa lähettää!



- Vastaanottaja kertoo, montako tavua puskureihin vielä mahtuu
  - TCP-segmentin otsakkeen kenttä **Receive window**
  - **Sovellus lukee tavut silloin kun haluaa**
  - **Koko on mukana jokaisessa TCP-segmentissä (molempiin suuntiin)**
- **Myös ruuhkanhallinta rajoittaa lähettämistä**

TCP Reno: Hidas aloitus (slow start) ja ruuhkanvälttely (congestion avoidance)

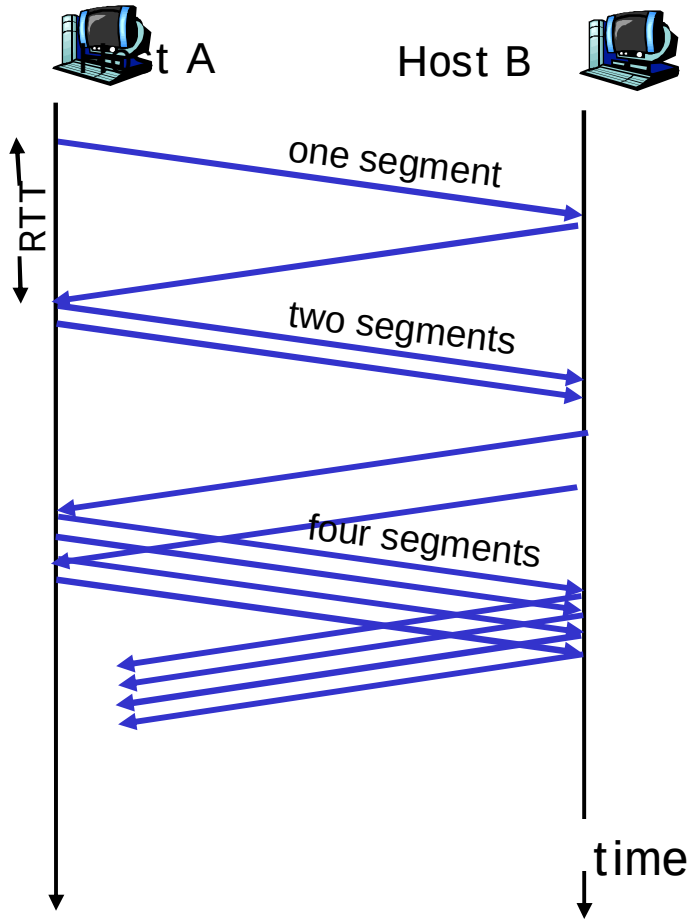
- **Aluksi ruuhkaikkuna = yksi segmentti**
  - Alussa hidas siirtonopeus= MSS/RTT
- **Kukin kuittaus kasvattaa yhdellä ruuhkaikkunan kokoa**
  - Eksponentiaalinen kasvu
  - Ikkuna kaksinkertaistuu yhden RTT:n aikana

hidas aloitus

- **Jos uudelleenlähetys, ruuhkaikkunan kooksi 1 segmentti**
  - Multiplicative decrease
- **Sen jälkeen kasvata ikkunaa yksi segmentti/RTT**
  - Lineaarinen kasvu (Additive increase)
  - Ruuhkan välttely (congestion avoidance)

ruuhkanvälttely

- Siirtonopeus = CognWin / RTT tavua/sek



TCP Reno: Tarkennus

- **Saatu 3 ACK-kaksoiskuittausta** (double ACK) (4 samaa kuittausta!)
  - Verkko pystyy välittämään dataa!
  - Ei siis (pahaa) ruuhkaa, ehkä paketissa bittivirhe tai paketti kadonnut jostain muusta syystä
- Nopea uudelleenlähetys (fast retransmit)
- **Nopea toipuminen** (fast recovery)
  - Puolita ruuhkaikkunan arvo ja kasvata sitten lineaarisesti

kynnysarvo?

- **Timeout (= uusi hidas aloitus)**
  - Verkko pahasti ruuhkautunut!
  - Pudota ikkunankoko arvoon 1
  - Kasvata eksponentiaalisesti kynnysarvoon asti
  - Kasvata sitten lineaarisesti

Hidas aloitus

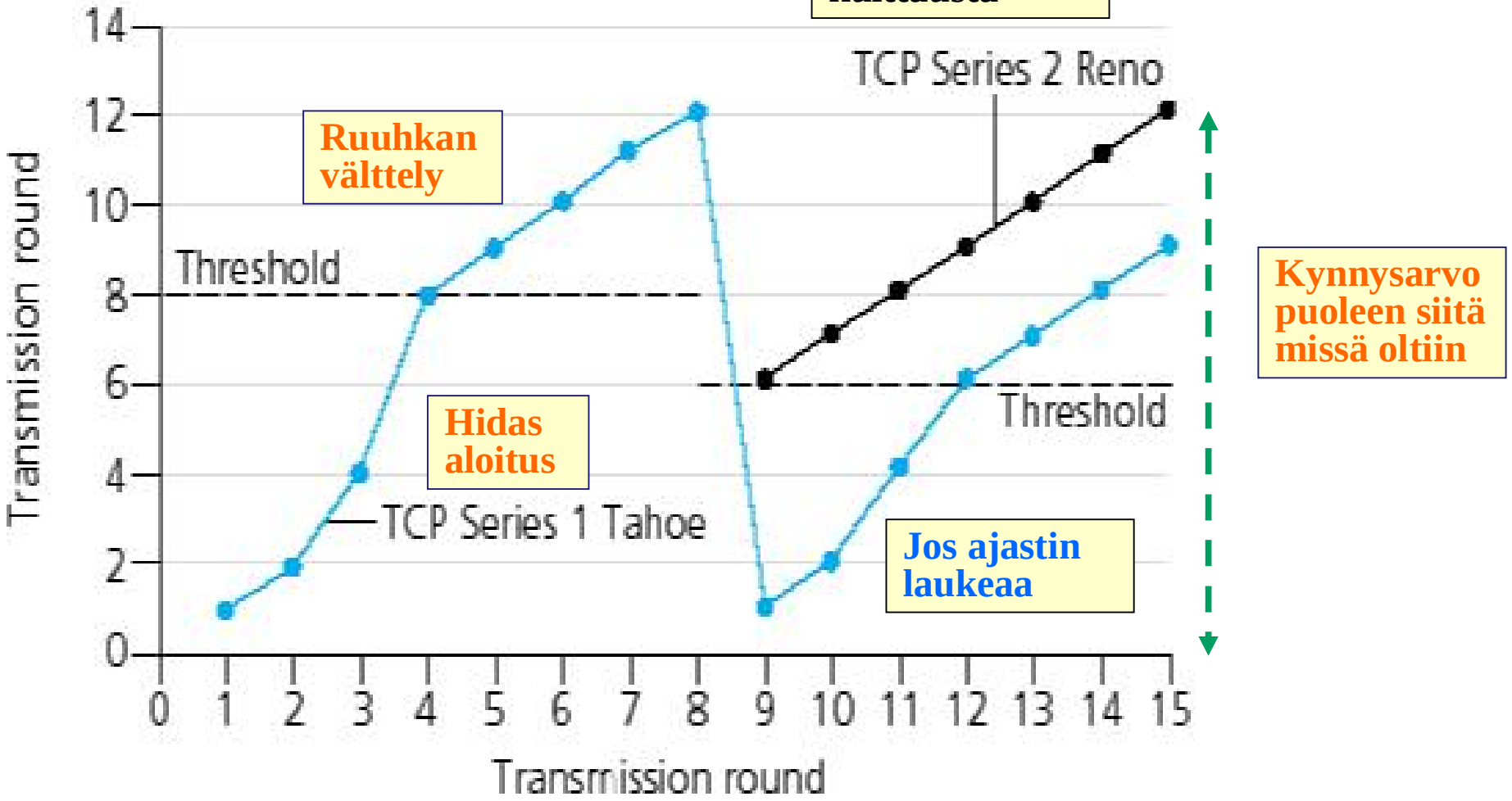
ruuhkanvälttely

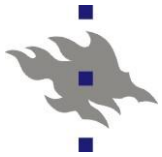
Vanha TCP Tahoe pudotti aina kokoon 1.

TCP Tahoe vs. TCP Reno

Jos 3 toisto-kuittausta

KuRo08: Fig 3.53





Ajastimen arvo?

- Aseta ajastin, kun segmentti on lähetetty
- Liian lyhyt aika
  - Ennenaikainen timeout, turha uudelleenlähetys
  - Turhat ruuhkatoiminnot
- Liian pitkä aika
  - Turhan hidas reagointi segmentin katoamiseen
  - Ei huomata ruuhkaa ajoissa
- Alkujaan: *Timeoutinterval = 2*RTT*
- *Kuittausaika vaihtelee suuresti ja nopeasti =>käytössä dynaaminen arvo*
  - *Saadaan jatkuvien mittausten perusteella*
- *Jos ajastin laukeaa, tuplaa Timeout*
  - *Exponential backoff*

TCP-segmentti

Otsake aina vähintään 20 B
Optio-osa tarvittaessa

Segmentti- ja kuittaus-numerot tavunumeroina

Ikkunankoko: paljonko tilaa vastaanottopuskurissa (tavua)

ACK= kuittausnumero validi,
RST (reset),
SYN yhteydenmuodostus
FIN yhteydenpurku
URG, PSH ei yleensä käytetä

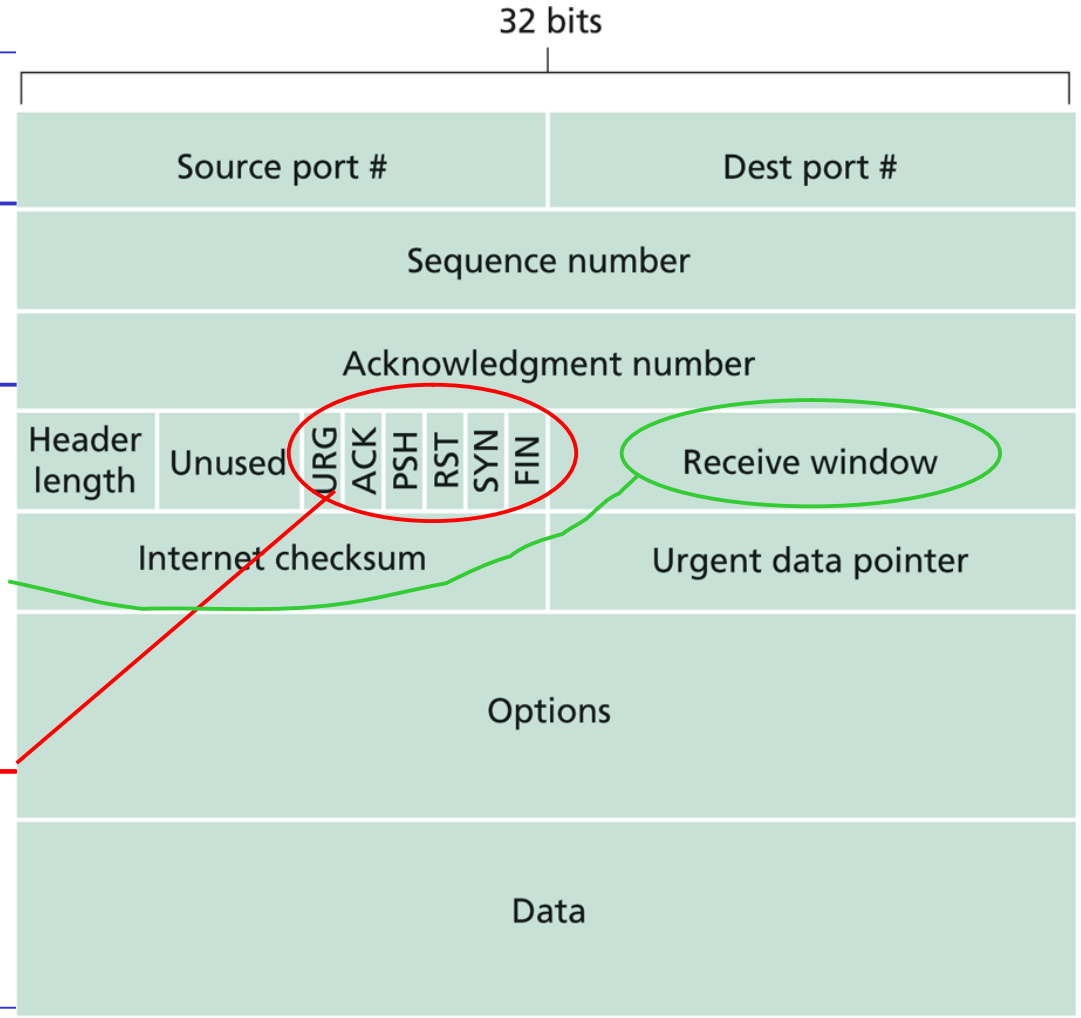
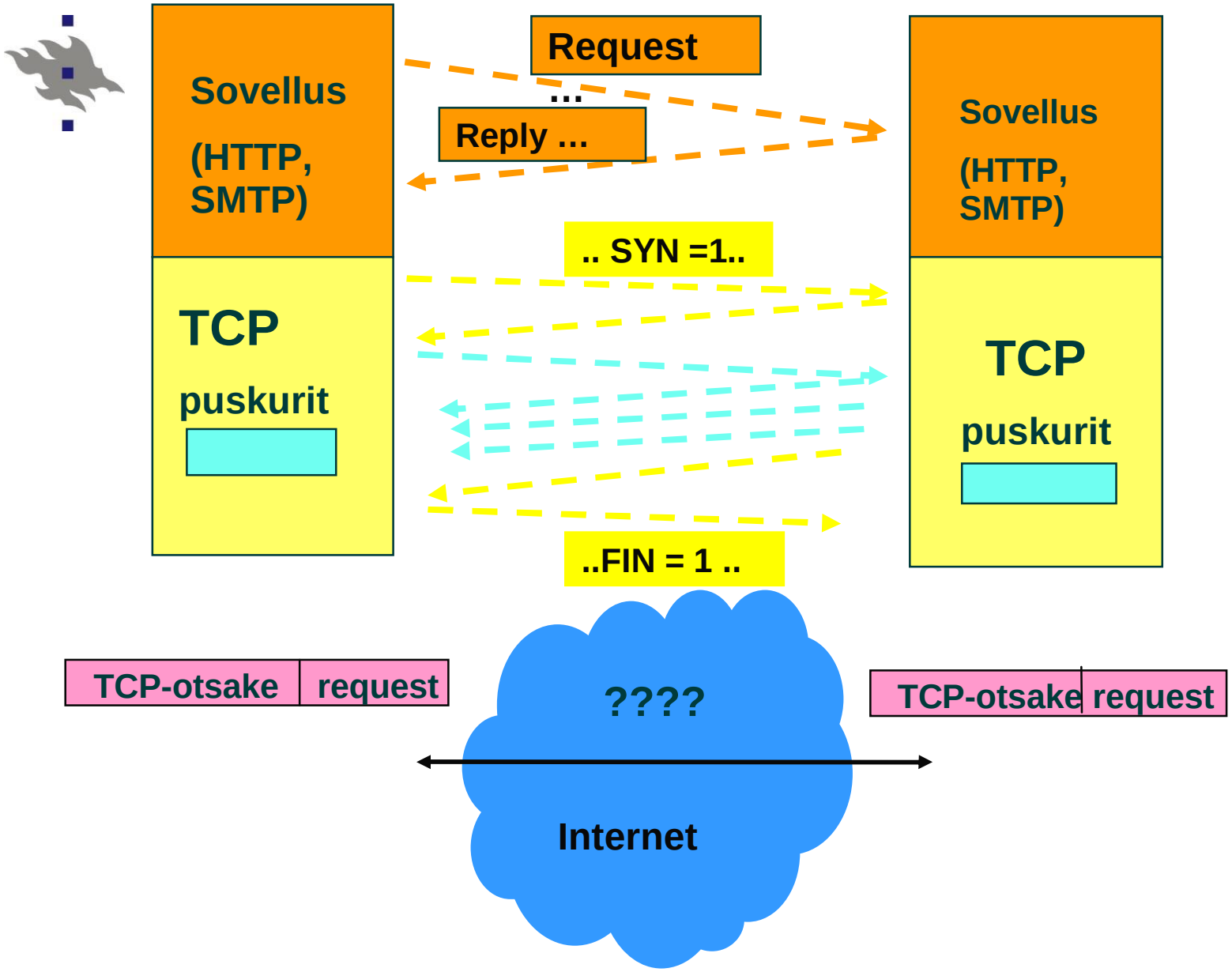
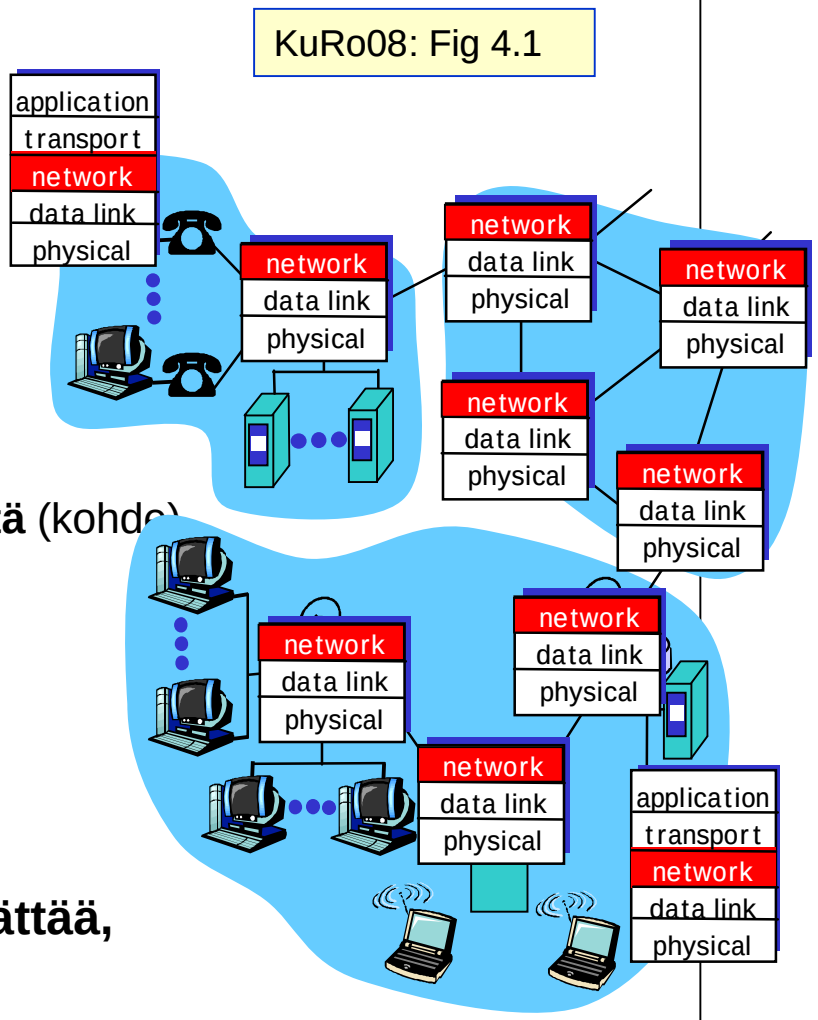


Figure 3.29 ♦ TCP segment structure

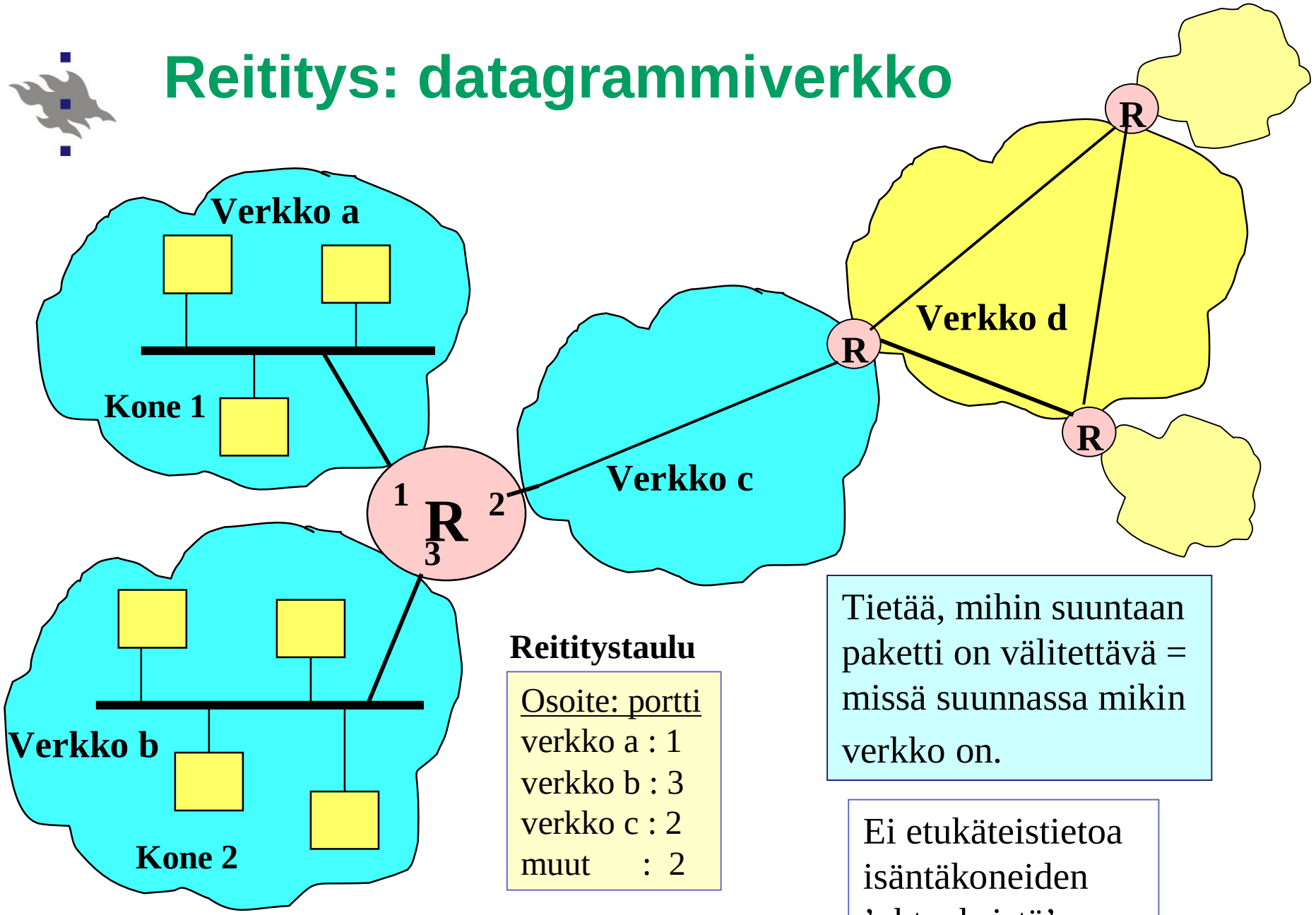

Sovellus
(HTTP, SMTP)
TCP
puskurit
Request
...
Reply ...
.. SYN =1..
..FIN = 1 ..
Sovellus
(HTTP, SMTP)
TCP
puskurit
TCP-otsake
request
????
Internet
TCP-otsake
request

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, linkkiin se lähetetään seuraavaksi**



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.

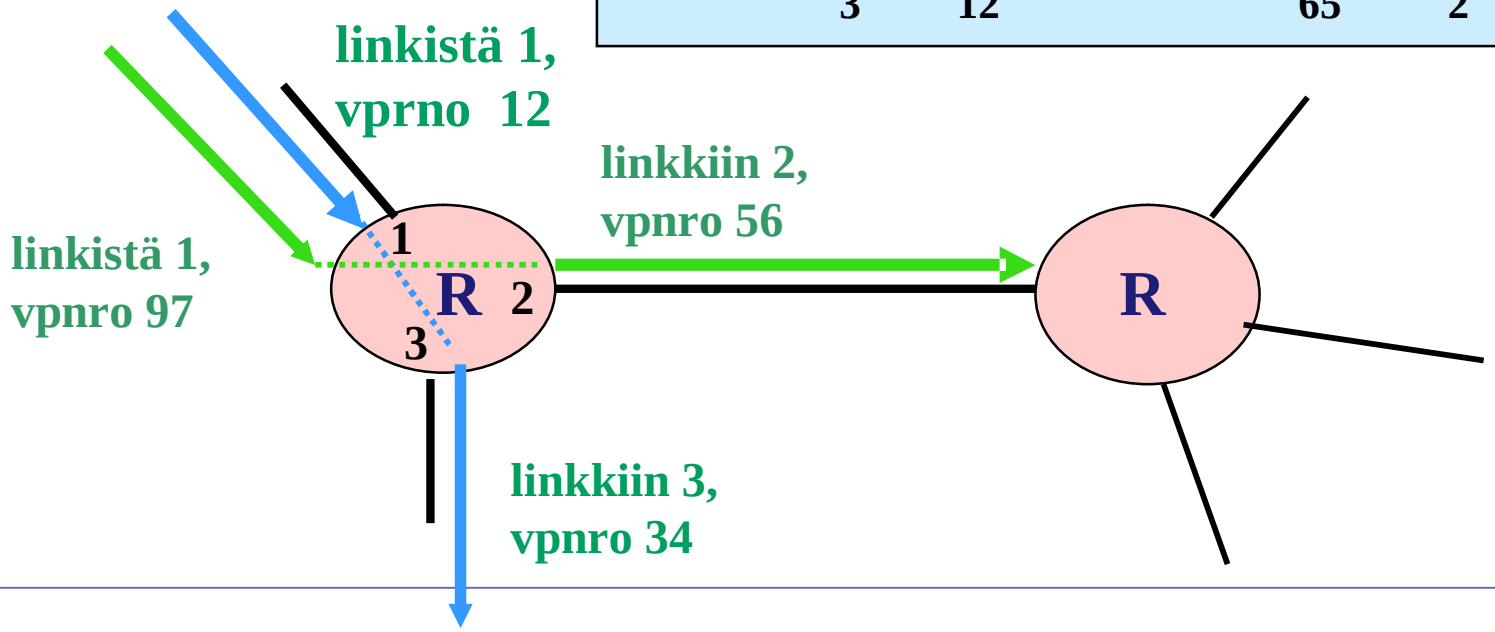
Ei etukäteistietoa isäntäkoneiden ’yhteyksistä’

Reititys: Virtuaalipiiriverkko

- **1. paketti muodostaa reitin, muut paketit kulkevat samaa reittiä**
  - otsakkeessa kohdeosoitteen
  - reititin ylläpitää tietoa piirinur

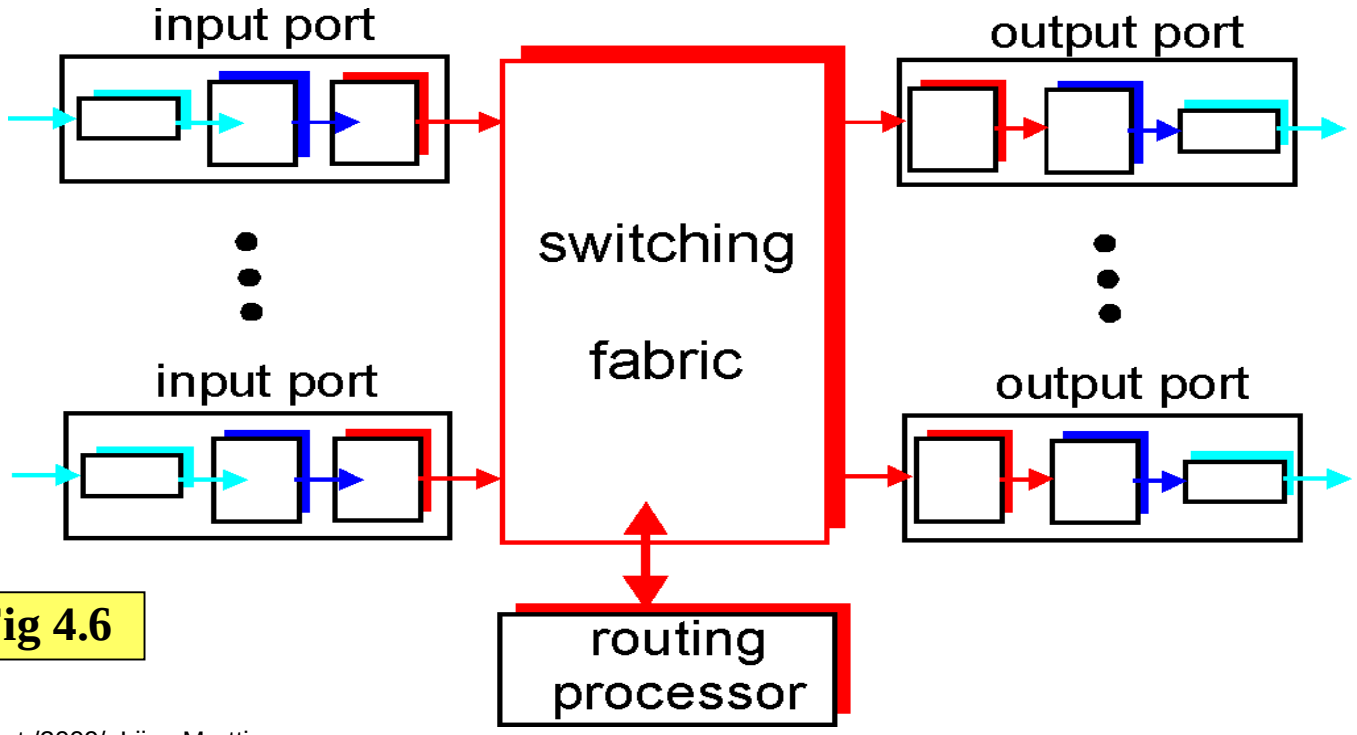
  Reititys = selvitä vpnro

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



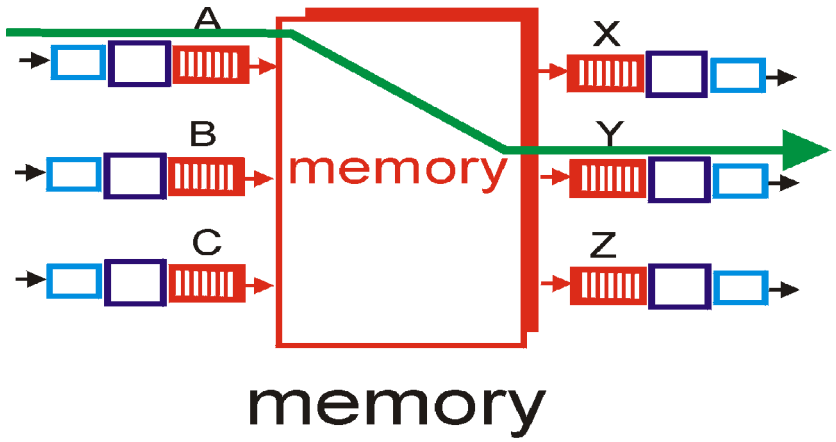
Reitittimen arkkitehtuuri

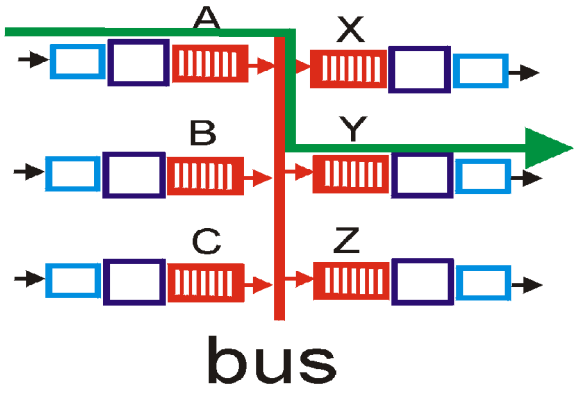
- Kaksi tehtävää
  - Välitä paketteja tulolinkeistä ulosmenolinkkeihin
  - Suorita reititysalgoritmia / -protokollaa
- Portti ~verkkokortti
  - Useita portteja niputettu yhteen linjakorteiksi (line card)

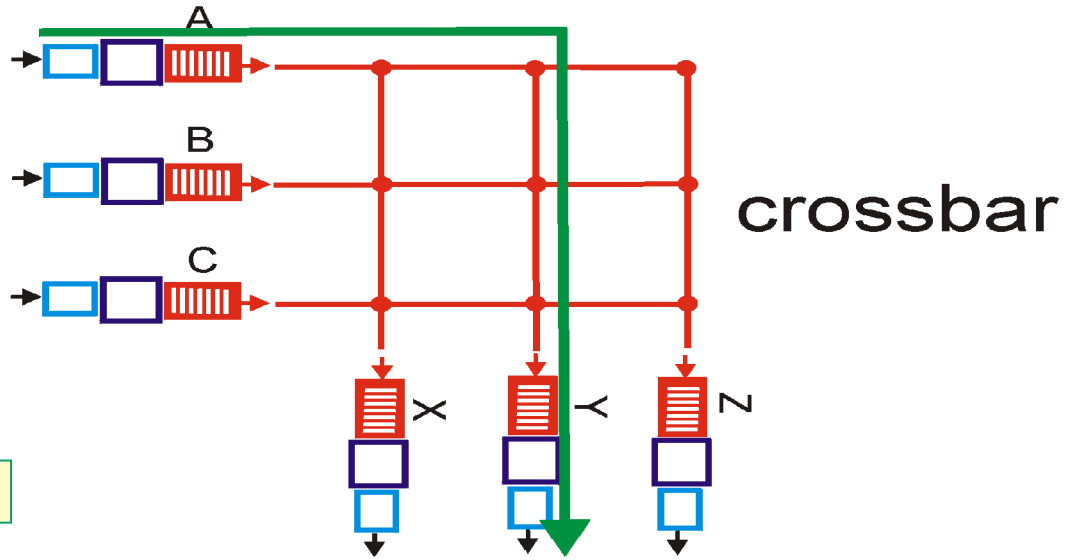


KuRo08:Fig 4.6

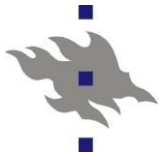
Kolme erilaista kytkentätapaa:





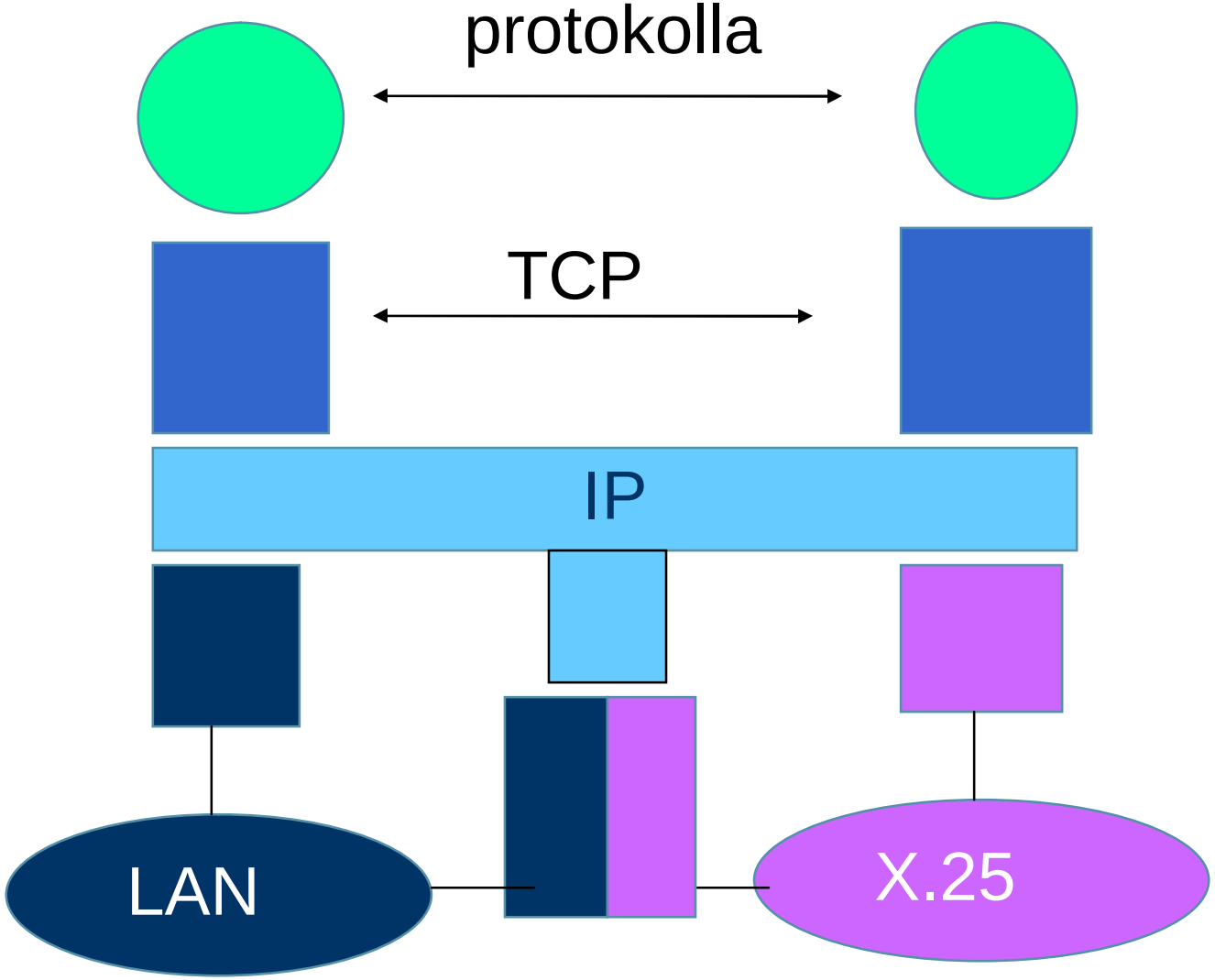


KuRo08: Fig. 4.8



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)


Sovellus-
protokolla
TCP
IP
LAN
X.25

IP-paketin rakenne (IPv4)

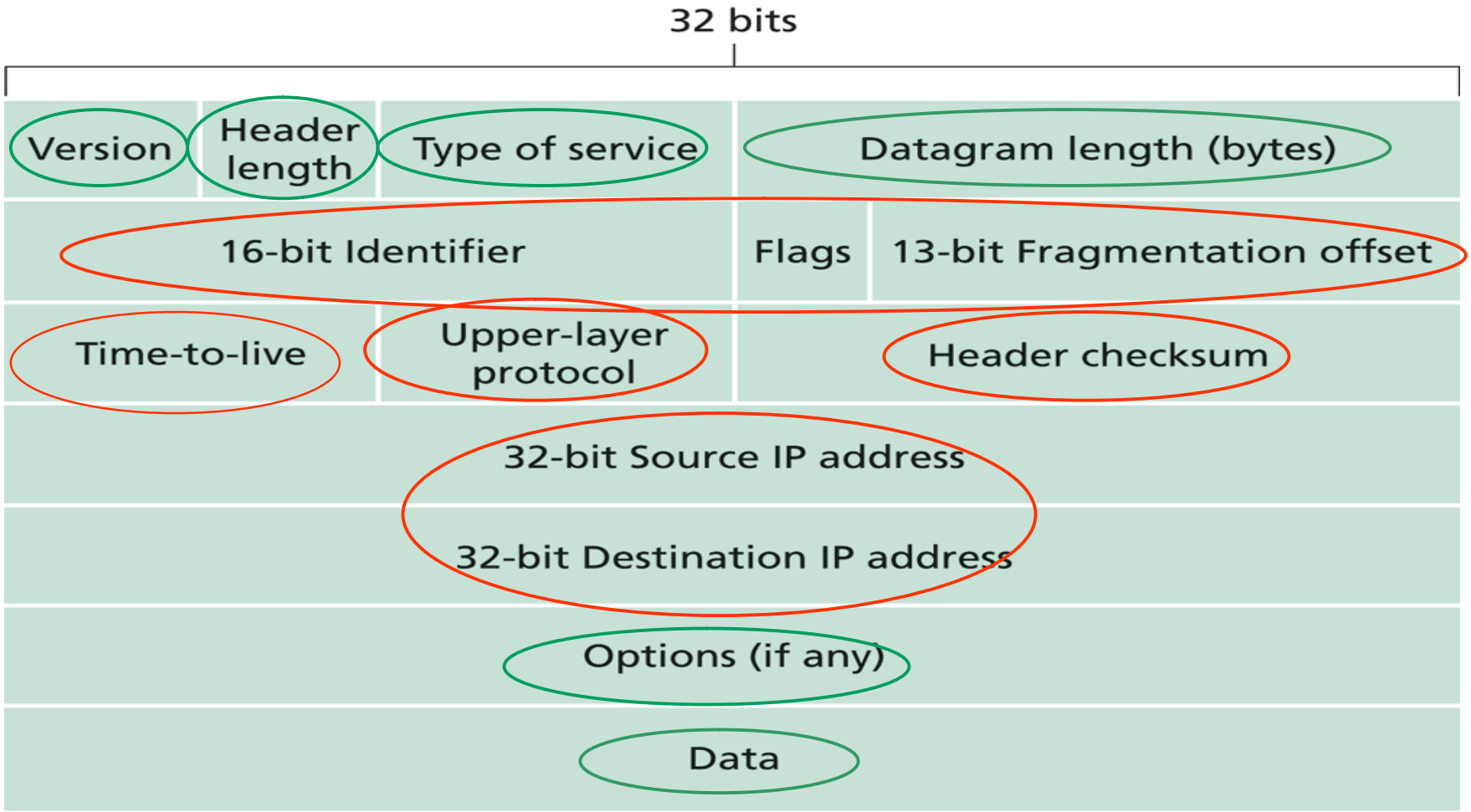


Figure 4.13 ♦ IPv4 datagram format

IP-pakettien paloittelu (fragmentointi)

Maximum transfer Unit (MTU)

suurin mahdollinen IP-paketti

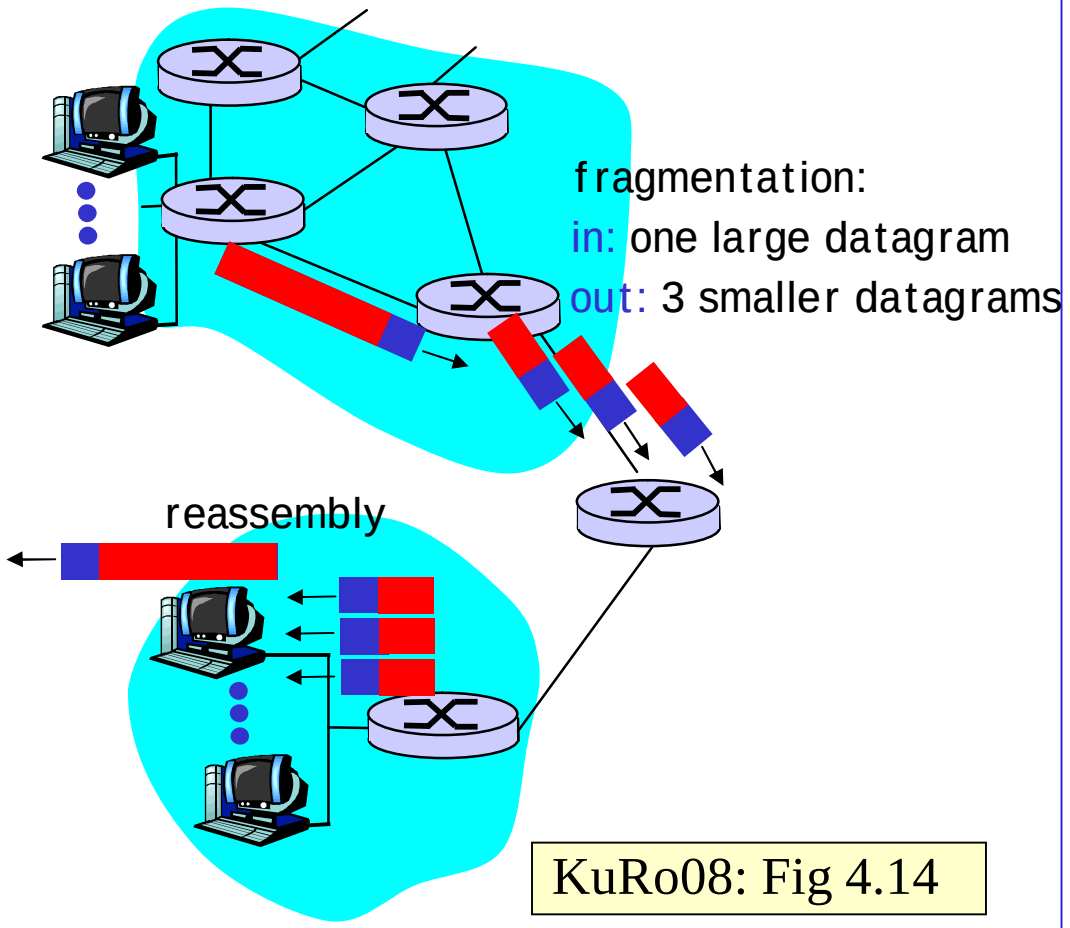
eri linkeillä eri koko

Esim. Ethernet 1500 B

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

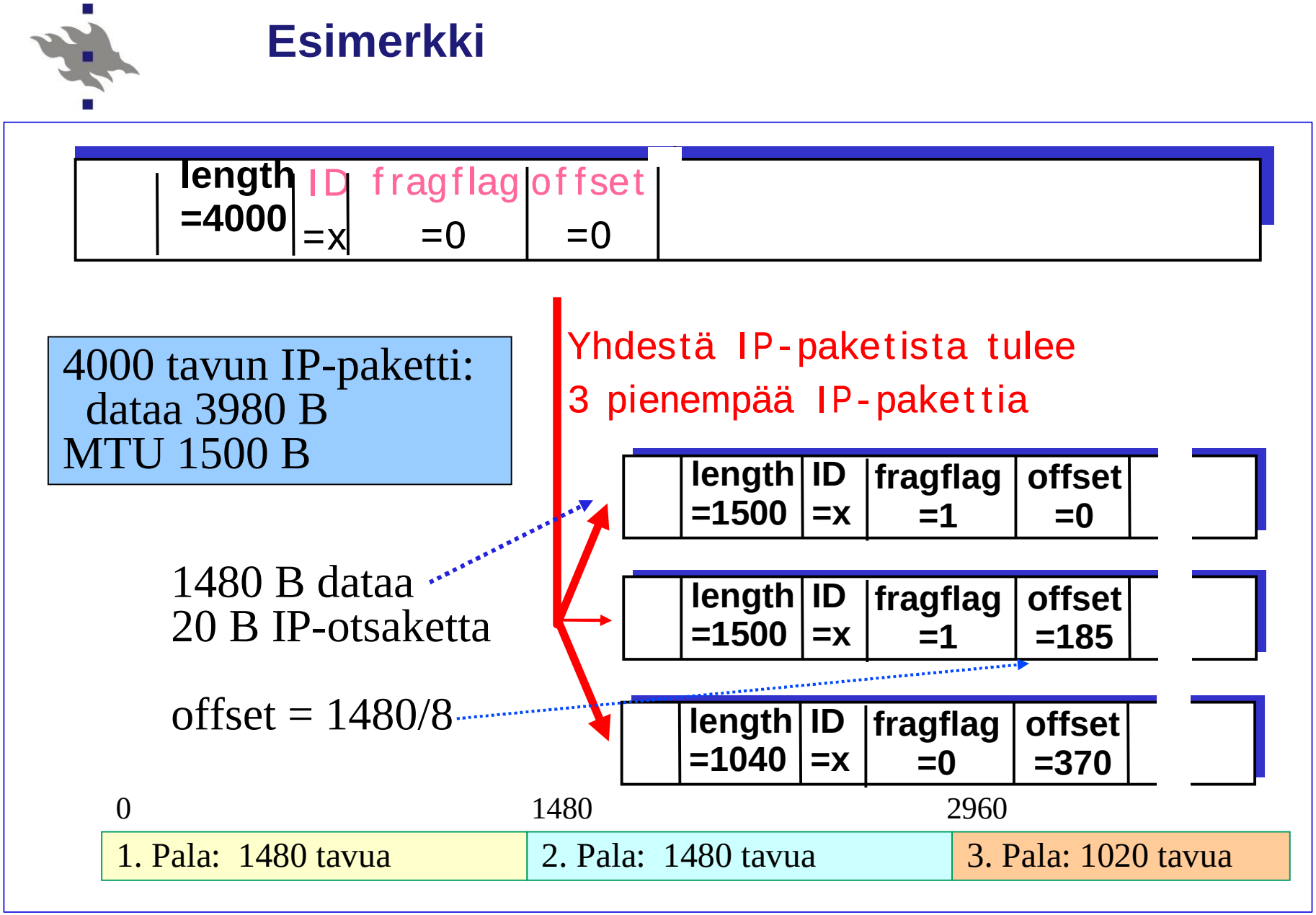
voivat kukin kulkea eri reittiä

IP-otsakkeessa kentät yhteenkuuluvien fragmenttien tunnistamiseksi



KuRo08: Fig 4.14

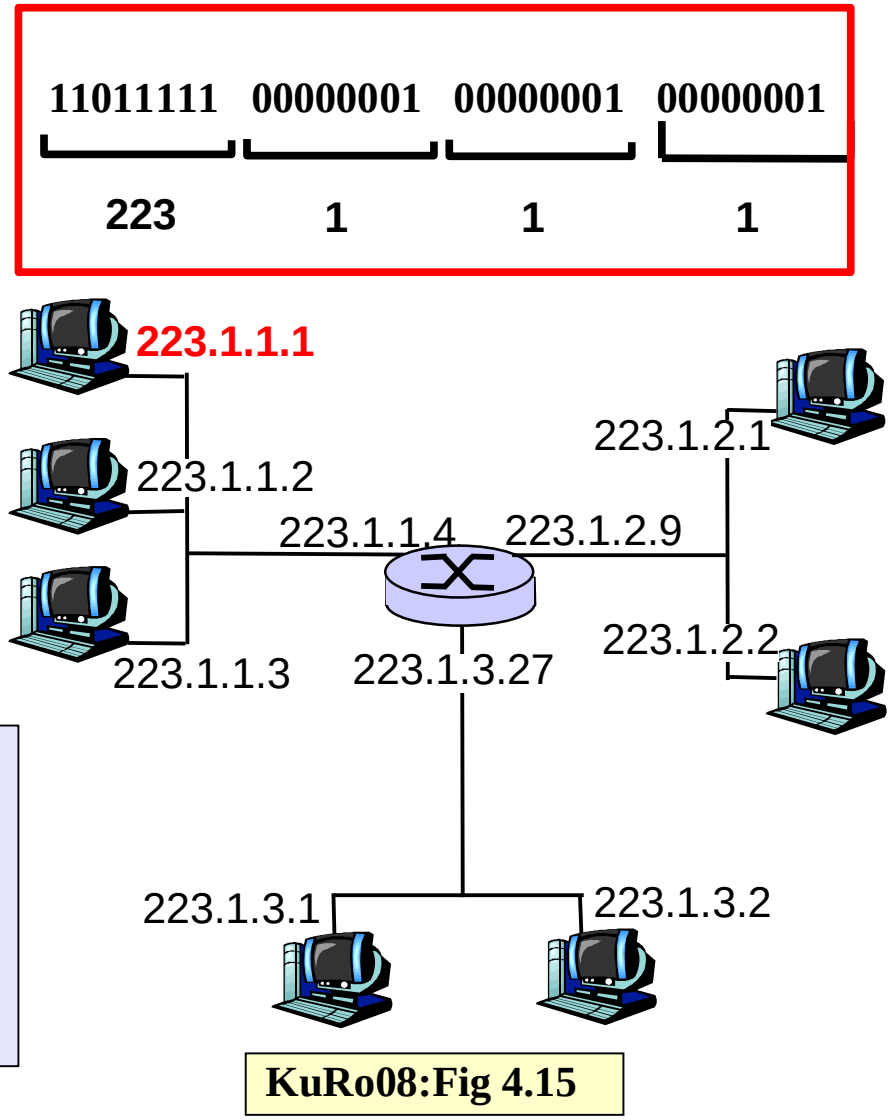
Esimerkki


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

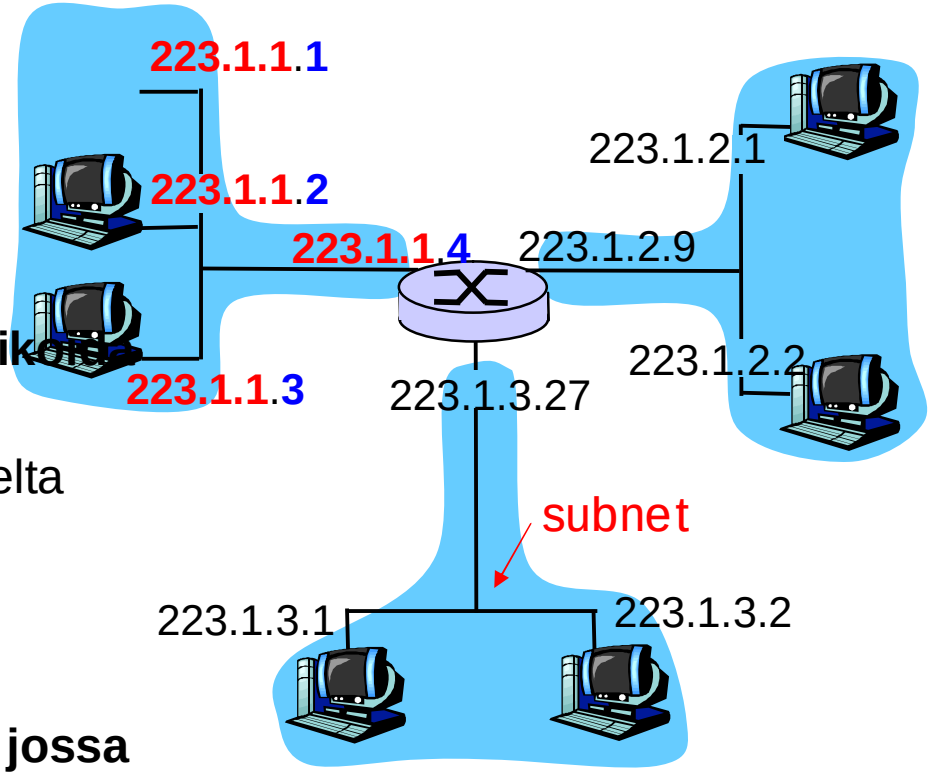
ICANN Internet Corporation for Assigned names and Numbers
verkkonumerot palvelun tarjoajille, nämä edelleen aliverkoiksi



KuRo08:Fig 4.15

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

CIDR: Classless InterDomain Routing

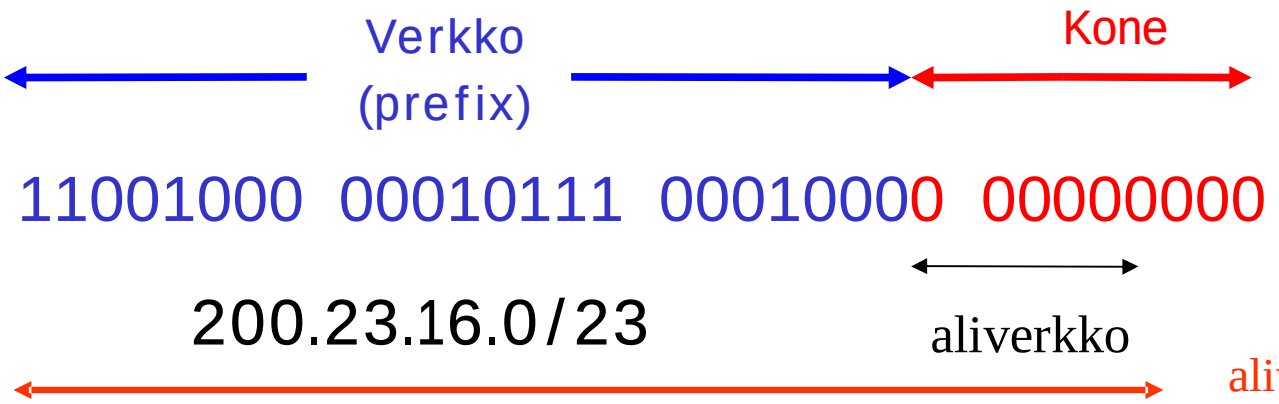
- Verkko-osa voi olla minkä tahansa kokoinen

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

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

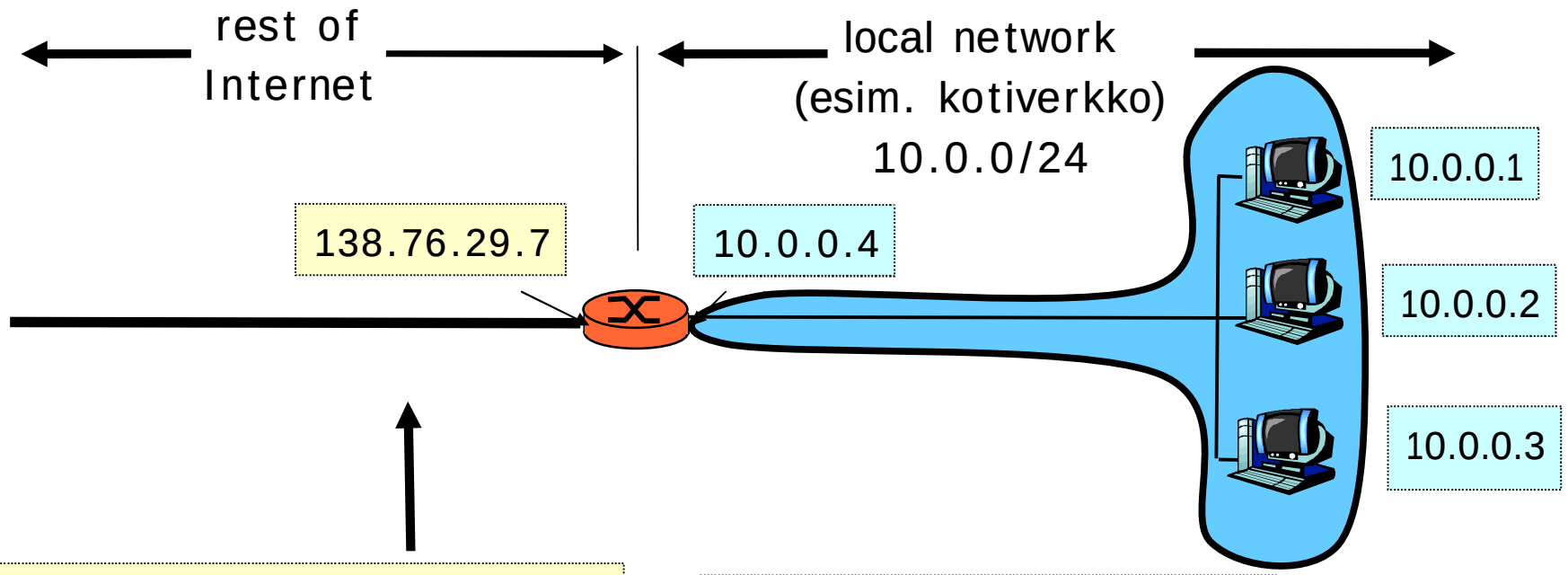
  x ilmoittaa verkko-osan bittien lukumäärän (prefix)

  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.



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)

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 = ∞

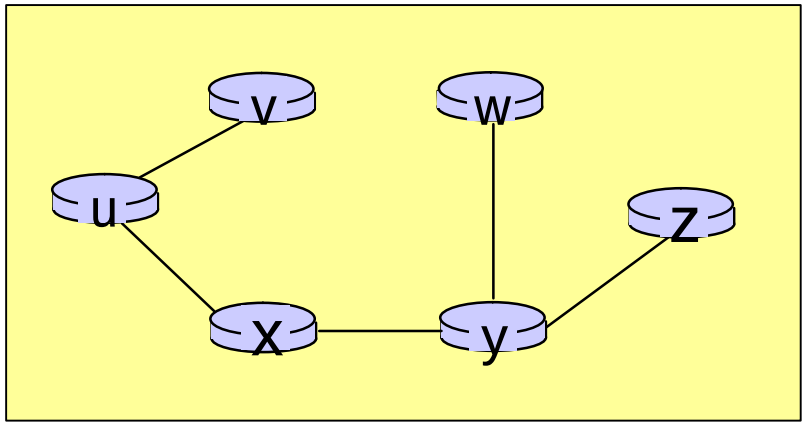
D(v) toistaiseksi edullisin kustannus solmuun v

p(v) solmun v edeltäjä reitillä

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

Lyhyimmät reitit ja reititystaulukko

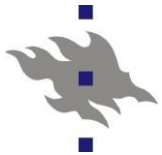
Resulting shortest-path tree from u:



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

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

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

node x table

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

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

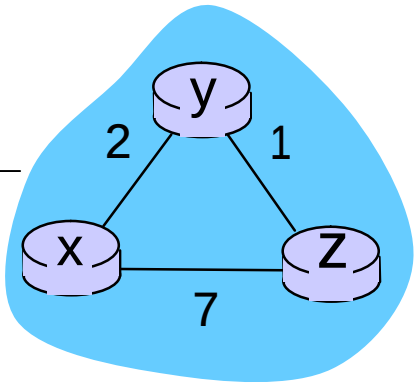
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

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

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

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



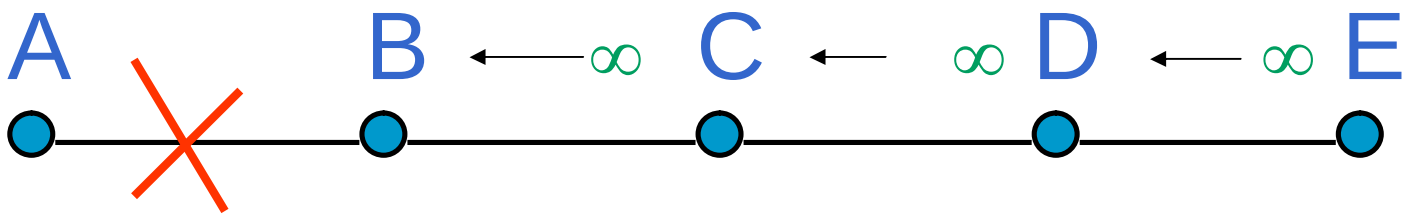
node z table

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

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

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

Huono uutinen etenee nopeasti:
“poisoned reverse”



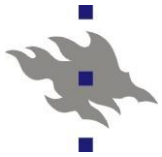
Ratkaisu count-to-infinity-ongelmaan!

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
∞	∞	∞	∞



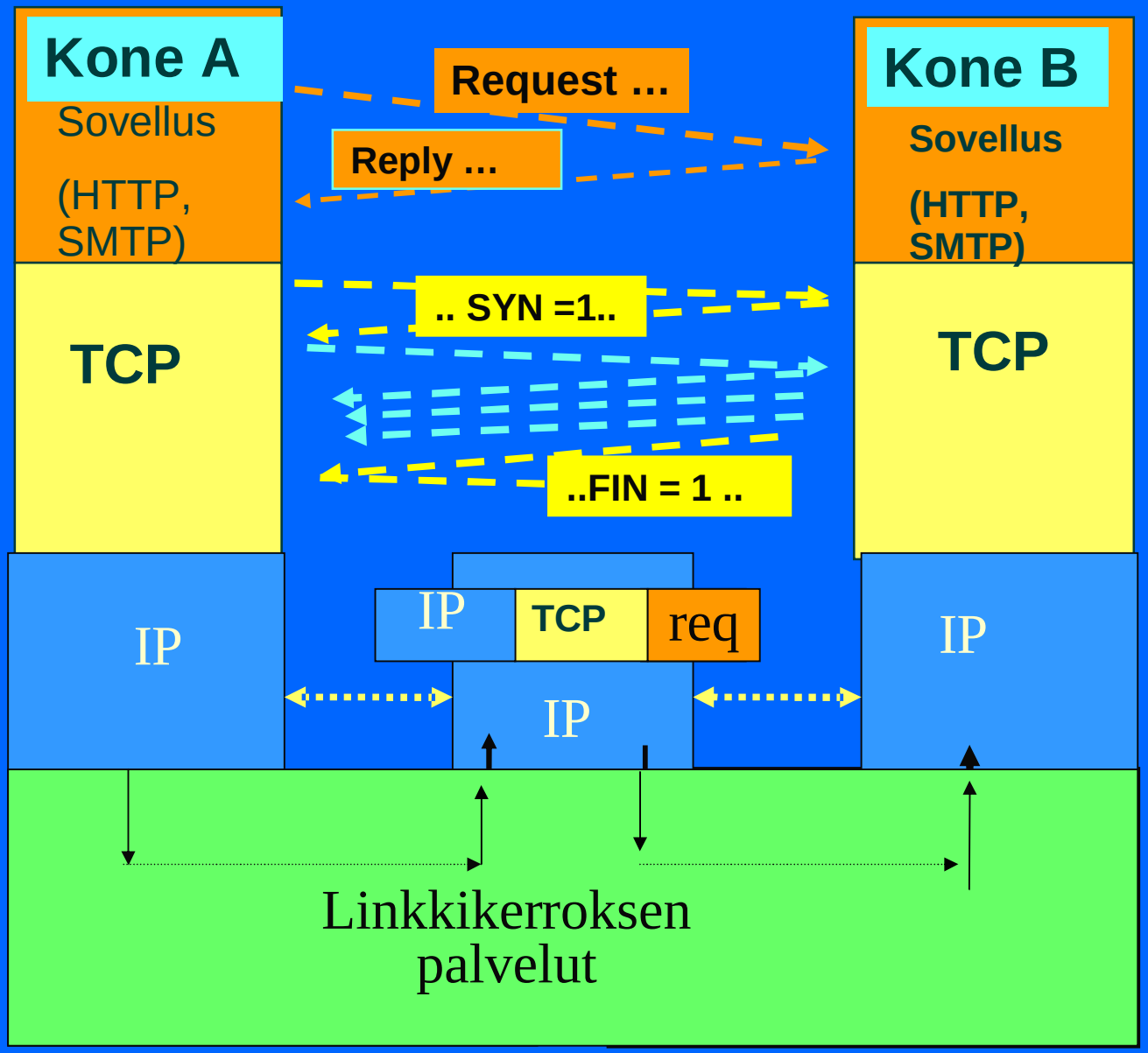
3) Hierarkkinen reititys

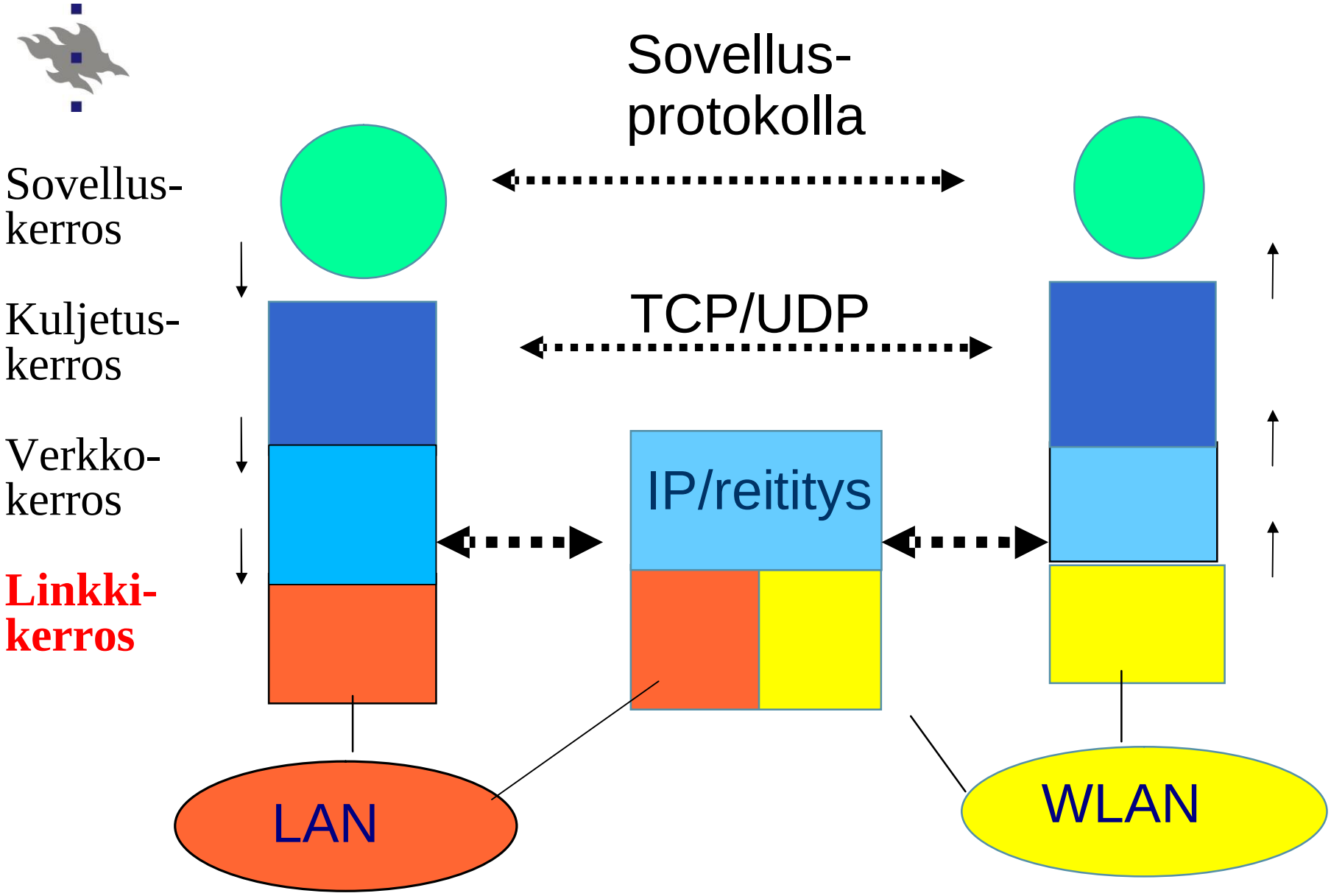
Jo CIDR ja IP-osoitteiden jakaminen lohkoina pienentää reititystauluja!

- Reitityksen skaalautuus?
  - Isossa verkossa runsaasti reitittimiä
    - Kaikki eivät voi tuntea kaikkia muita
    - Reititystaulut suuria, reittien laskeminen raskasta
    - Reititystietojen vaihtaminen kuluttaa linjakapasiteettiä

netstat -r

- 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)


Kone A
Sovellus
(HTTP, SMTP)
TCP
IP
Request …
Reply …
.. SYN =1..
..FIN = 1 ..
IP
TCP
req
IP
Kone B
Sovellus
(HTTP, SMTP)
TCP
IP
Linkkikerroksen palvelut


Sovellus-
protokolla
Sovellus-
kerros
Kuljetus-
kerros
TCP/UDP
Verkko-
kerros
IP/reititys
Linkki-
kerros
LAN
WLAN

Linkkikerroksen tehtäviä (2)

NIC (Network Interface Card)
linkki- ja fyysinen kerros

- **Vuonvalvonta, puskurointi**

  Kytkimessä on useita erinopeuksisia linkkejä

- **Virhevalvonta**

  signaali vaimenee, taustakohina häiritsee, ...

  Kehyksessä on tarkistustietoa (error detection and correction bits)

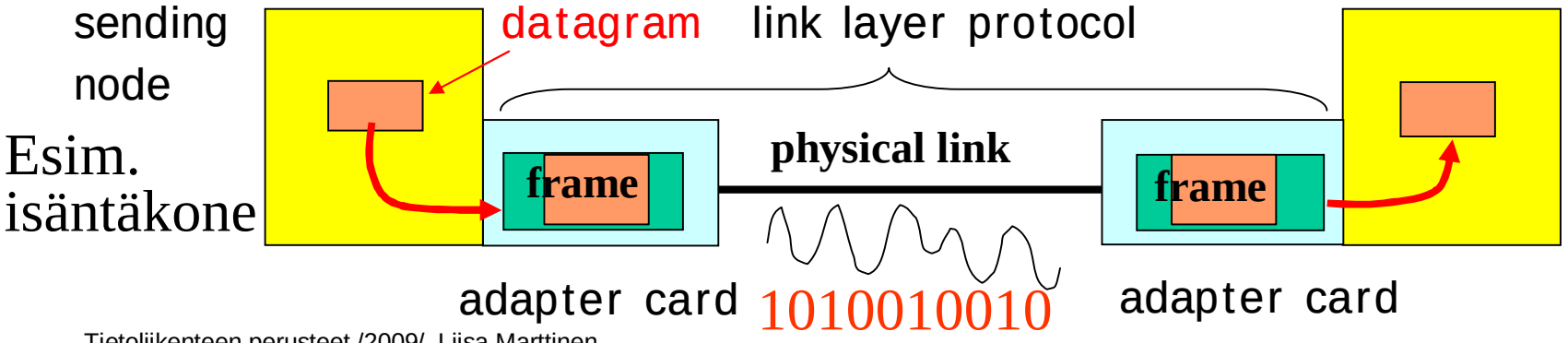
  Vastaanottava solmu korjaa, jos pystyy

  Jos ei pysty, pyytää uudelleen tai hävittää

- **Yksisuuntainen /kaksisuuntainen liikenne**

  Yksisuuntainen: lähetysvuorojen hallinta

Esim. reititin



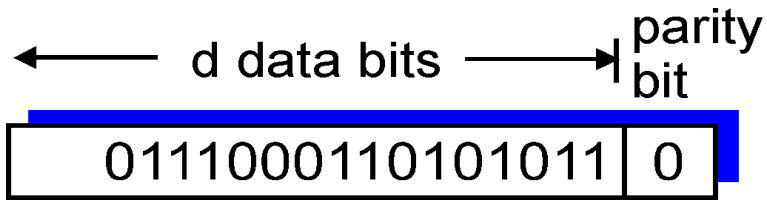
Linkkikerroksen tehtäviä

- **Kehystys** (framing)
  - Kehyksen rakenne ja koko riippuu siitä, millainen linkki on kyseessä
  - Otsake, data, lopuke

- **Kohteen ja lähteen osoittaminen**
  - Yhteiseen linkkiin voi olla liitettynä useita laitteita
  - Käytössä laitetason MAC-osoite (Medium access control)
- **Yhteisen linkin varaus ja käyttö** (link access)
  - Esim. langaton linkki, keskittimiin yhdistetyt linkit
- **Luotettava siirto**
  - Langattomilla linkeillä suuri virhetodennäköisyys
    - Linkkitaso huolehtii oikeellisuudesta
  - Miksi tästä täytyy huolehtia vielä kuljetuskerroksella?
    - Jotkut linkkityypit eivät huolehdi lainkaan!
    - Jos kehys hävitettävä ..

Pariteettitarkistus

- **Pariteettibitti**
  - Parillinen vs. pariton pariteetti
  - Virheryöpyssä jopa 50% voi jäädä huomaamatta



- **Kaksiulotteinen pariteetti**
  - Erikseen horisontaalinen (parillinen) ja vertikaalinen (pariton) pariteetti
  - Pystyy korjaamaan yhden bitin virheen.

- **Hamming-koodi**
  - Korjaa yhden bitin virheen

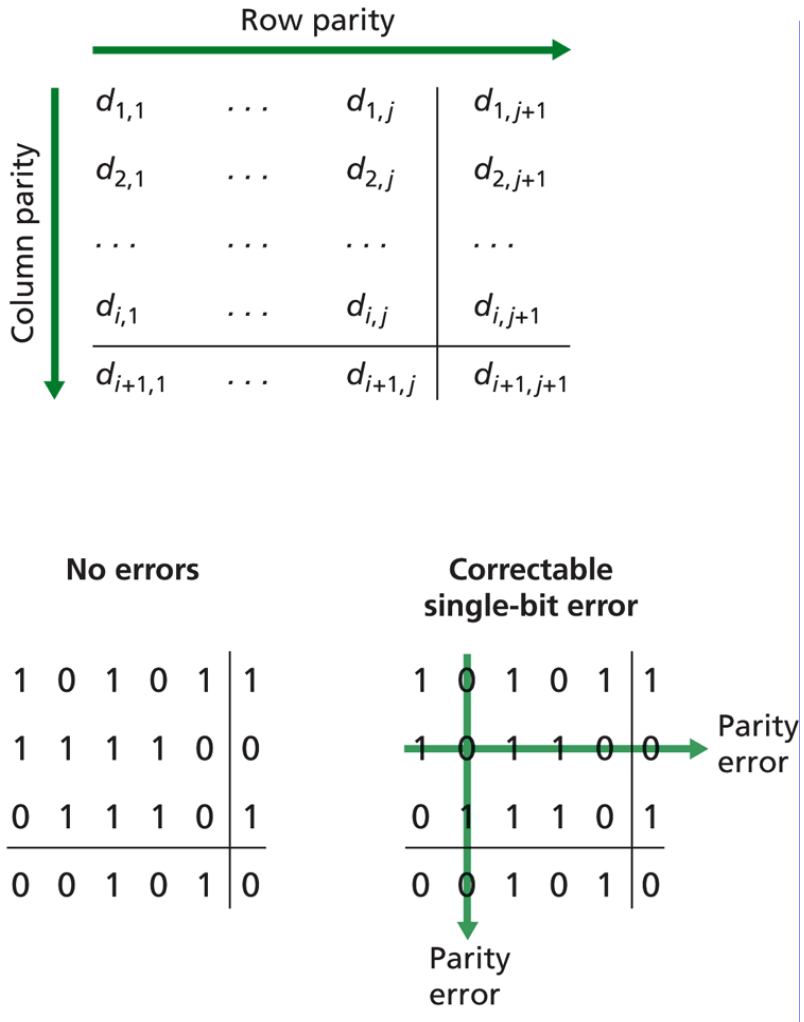


Figure 5.6 ♦ Two-dimensional even parity

CRC-esimerkki

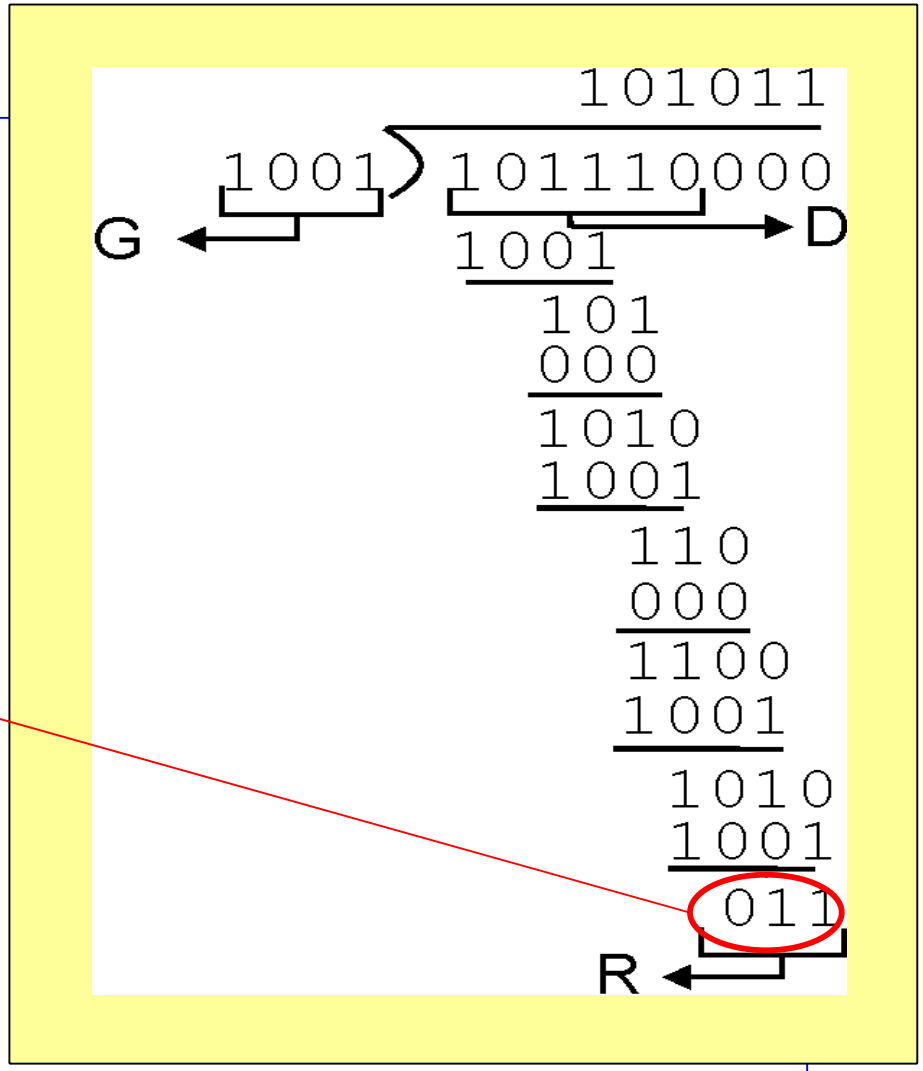
Data: 101110

G: 1001, polynomina

$1*x^3 + 0*x^2 + 0*x^1 + 1*x^0$

<D,R>: 101110???

Lähetä: 101110011



Modulo 2-aritmetiikka

vähennyslasku yhteenlaskuna

ei lainaamista, ei muistinumeroita

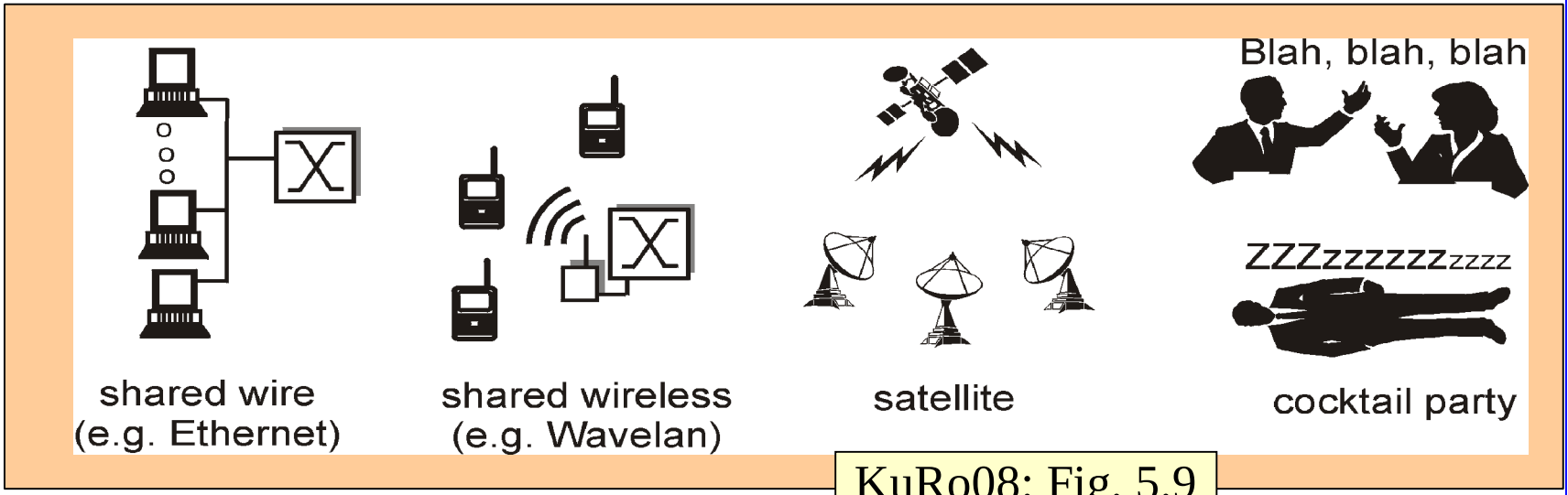
= bittitason XOR

1+1 =0, 1+0 = 0+1 = 1, 0+0=0

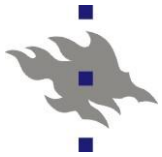
KuRo08:Fig 5.8

Yksi kanava

- **Kaksipisteyhteys** (point-to-point)
  - PPP-protokolla, puhelinyhteys (dial-up access)
  - Ethernet-piuha kytkimen ja isäntäkoneen välissä
- **Yleislähetysyhteys** (broadcast)
  - Alkuperäinen Ethernet, Ethernet keskittimen ja isäntäkoneen välissä, kaapelimodeemiyhteys (upstream), WLAN, satelliitti,



KuRo08: Fig. 5.9



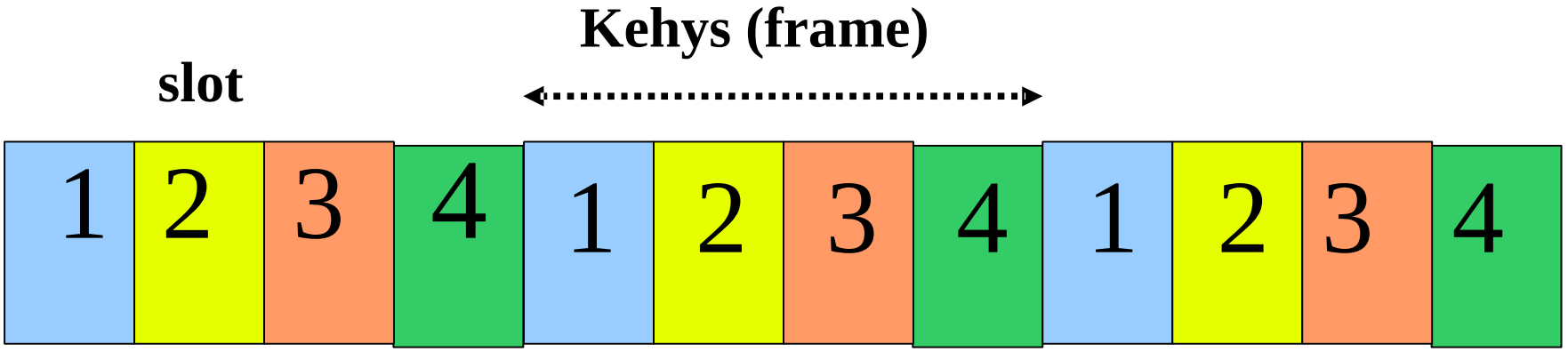
Lähetysvuorojen jakelu

1) Kanavanjakoprotokollat (channel partitioning protocol)
   Jaa kanavan käyttö 'viipaleisiin' (time slots, frequency, code)
   Kukin solmu saa oman viipaleensa
   TDMA, FDMA, CDMA
   “käytä sinä tätä puolta, minä tätä toista”

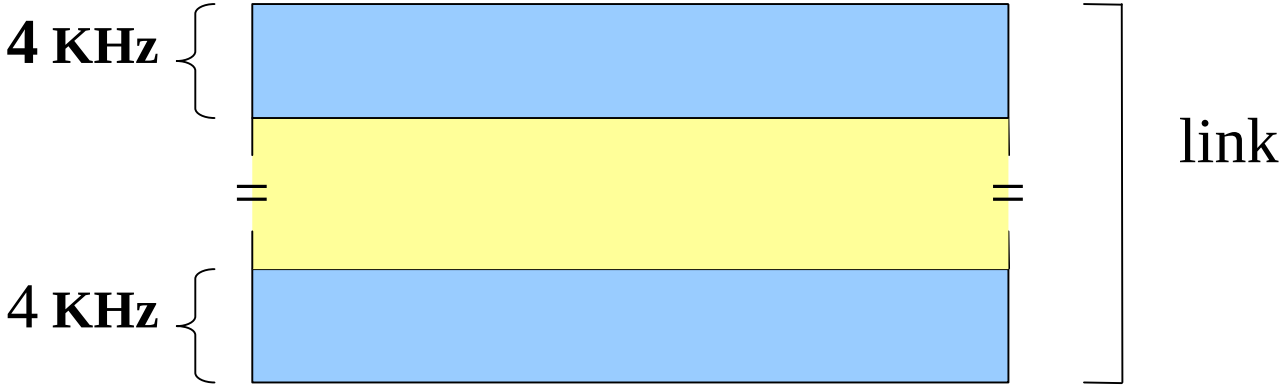
2) Kilpailuprotokollat (random access protocols)
   “Se ottaa, joka ehtii.”
   Jos sattuu törmäys, yritä myöhemmin uudelleen.
   Aloha,CSMA, CSMA/CD

3) Vuoronantoprotokollat (taking-turns protocols)
   Jaa käyttövuorot jollakin sovitulla tavalla:
   pollaus, vuoromerkki, ...
   “Minä ensin, sinä sitten.”

TDMA:



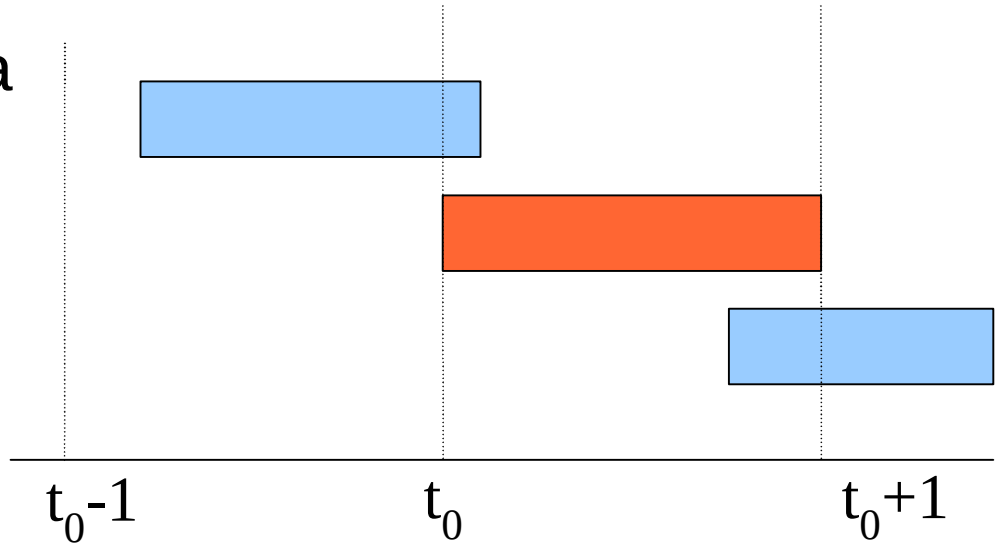
FDMA:



CDMA: Eri lähettäjillä omat keskenään ortogonaaliset sirukoodit biteille => yhteislähetyksestä kyetään erottamaan eri lähettäjät

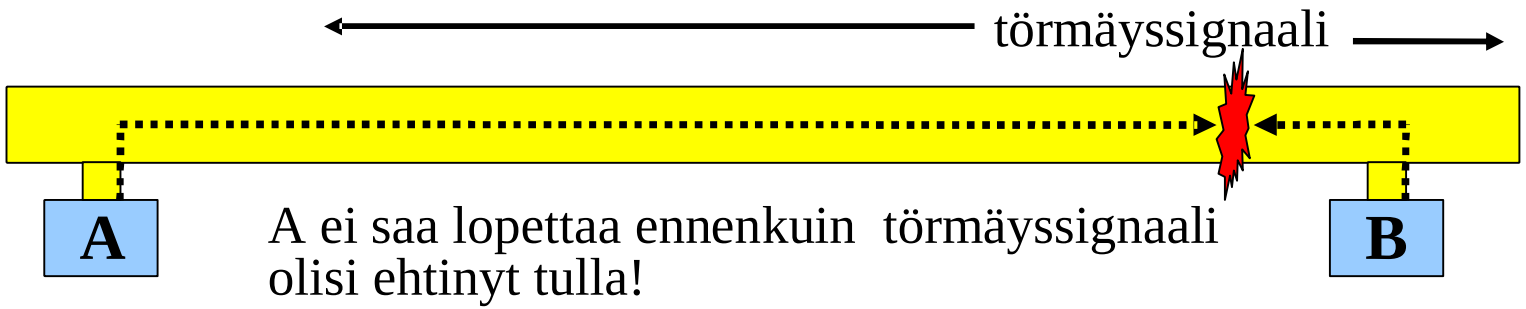
Aloha

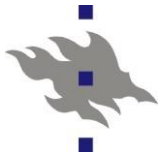
- Hawaijilla, 70-luvulla radiotietä varten
- Lähetä heti, kun on lähetettävää
  - Ei mitään kuuntelua ennen lähetystä
- Kuuntele sitten, onnistuiko lähetys
  - Lähiverkossa törmäys havaitaan 'heti', sillä siirtoviive on pieni (toisin kuin satelliitilla)
- Jos törmäys, niin odota satunnainen aika ja yritä uudelleen
- Yksinkertainen
- Törmäyksen td. suuri
  - Max tehokkuus ~ 18%



CSMA/CD (with Collision Detection)

- Asema kuuntelee myös lähettämisen jälkeen
  - Langallinen LAN: törmäys => signaalin voimakkuus muuttuu
    - Esim. Ethernet
  - Langaton LAN: hankalaa
- Jos törmäys
  - Niin keskeytä heti lähettäminen
  - ja yritä uudestaan satunnaisen ajan kuluttua
  - Näin törmäyksen aiheuttama hukka-aika pienenee
- Kauanko kuunneltava?
  - 2* maksimi etenemisviive solmujen välillä





Linkkikerroksen fyysinen osoite

- 32 bitin IP-osoite verkkokerroksella
  - Reitityksen tapa viitata koneeseen
- Erilaisilla linkkikerroksilla omat tapansa osoittaa oikea linkki (~ verkkokortti)
  - Siirtokehys on kuljetettava fyysisen linkin yli jollekin toiselle samaan verkkoon (LAN) kytketyistä laitteista
- **MAC-osoite** (Media Access Control Address)
  - Käytetään myös nimiä LAN-osoite, fyysinen osoite, laiteosoite, Ethernet-osoite, ...
  - Liitetty valmistusvaiheessa kiinteästi laitteeseen

Analogia:

IP-osoite ~ katuosoite MAC-osoite ~ henkilötunnus

ARP-protokolla (Address Resolution Protocol)

- Ratkaisuna ARP-protokolla ja ARP-taulu
  - **ARP-protokolla** lähettää **yleislähetysosoitteella** kyselyn, jonka kaikki vastaanottavat.
    Oman osoitteensa tunnistava laite **vastaa kyselijän MAC-osoitteeseen** ja kertoo oman MAC-osoitteensa

“aa-bb-cc-dd-ee-ff”, “FF-FF-FF-FF-FF-FF”
“Kenen IP-osoite on “xx:yy:zz:vv”?

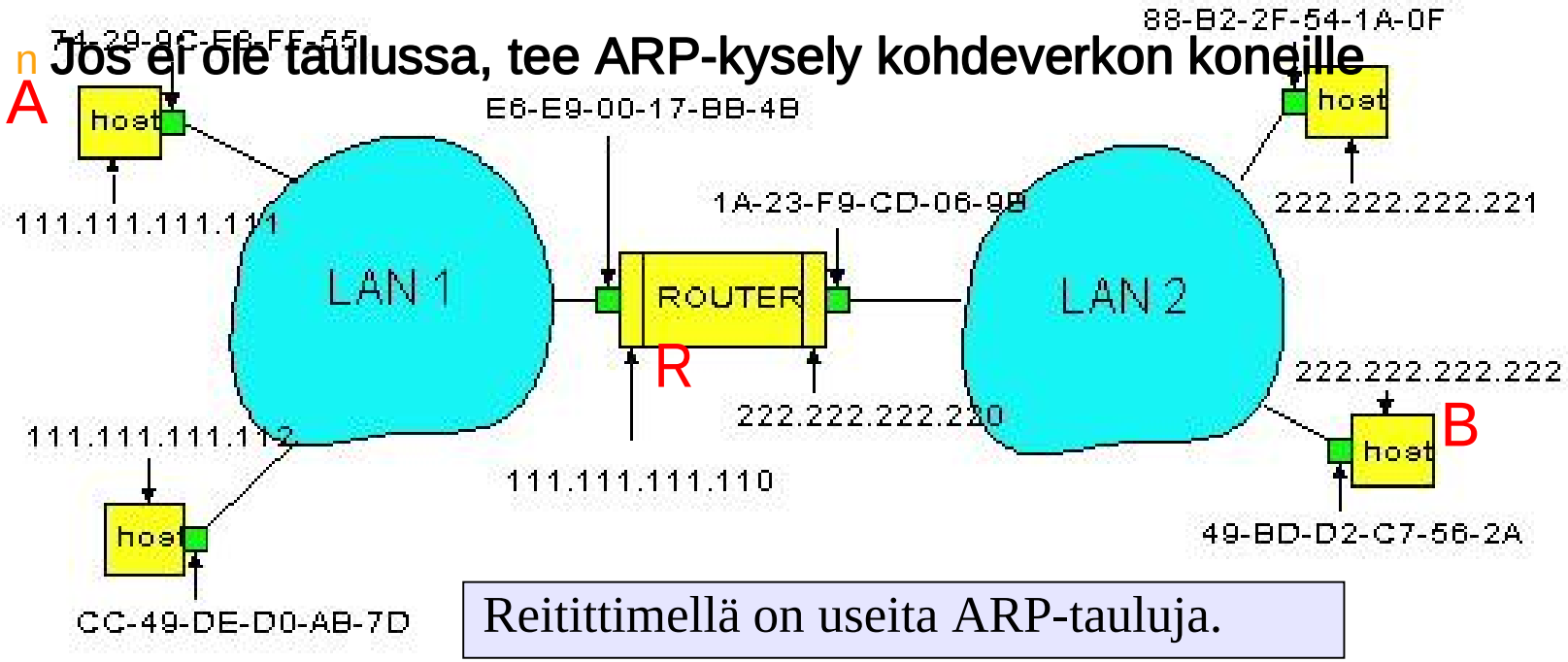
MAC-yleislähetysosoite: FF-FF-FF-FF-FF-FF

“kk-ll-mm-nn-oo-pp”, “aa-bb-cc-dd-ee-ff”

  - **ARP-taulu** pitää tallessa kyselyjen vastauksia: IP-osoite, MAC-osoite, TTL)
    Kussakin koneessa (myös reitittimessä) jokaiselle aliverkolle oma taulunsa
    Tiedot vanhenevat n. 20 minuutissa (time-to-live)

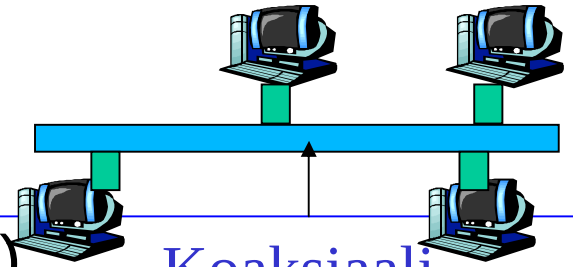
Lähettäminen toiseen verkkoon (1)

- Ensin omalle reitittimelle sen MAC-osoitteella ja reitin ohjaa eteenpäin
  - **Reititystaulussa on verkko-osoite, jonne paketti seuraavaksi ohjattava**
  - **Katso kohdeverkon ARP-taulusta kohteen MAC-osoite**
  - **Jos ei ole taulussa, tee ARP-kysely kohdeverkon koneille**



Reitittimellä on useita ARP-tauluja.

10BaseT ja 100BaseT



10 Mbps tai 100Mbps (Fast Ethernet, FE)

- T = Twisted Pair eli kierretty parikaapeli

Maks. etäisyys keskittimeen 100 m

Keskitin (hub) toistaa bitit heti sellaisenaan muille

- Fyysisen tason toistin (repeater);
- Yleislähetys, Signaalin vahvistus

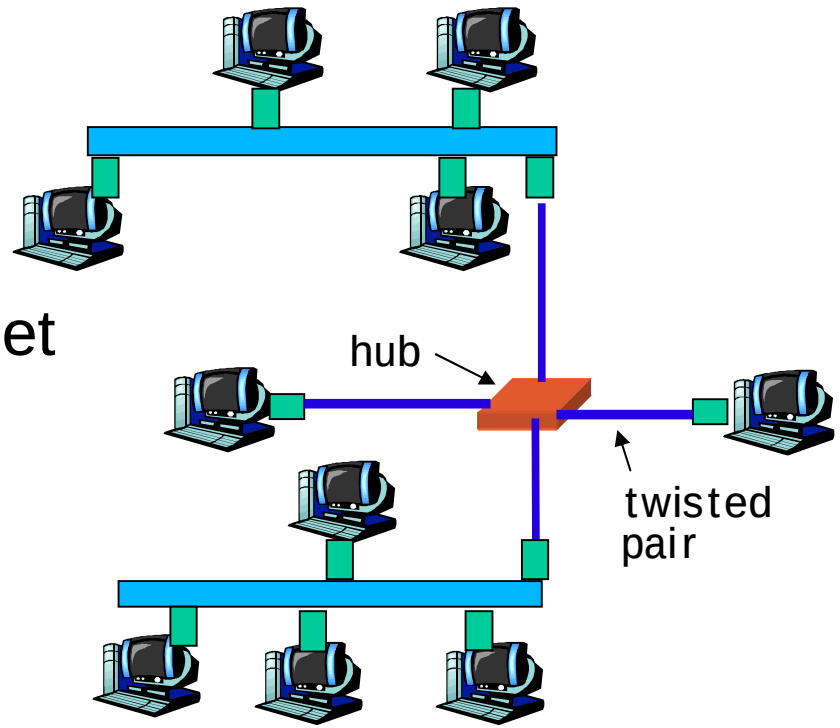
Verkkokortit käsittelevät törmäykset

- Maks. 30 konetta / keskitin

Keskitin osaa jättää huomiotta vikaantuneen kortin

Kerää myös tietoa liikenteestä

- Törmäysten lkm, keskim. kehyskoko, ...



Gigabitin Ethernet (GE)

1 Gbps tai 10 Gbps

Edelleen sama kehysformaatti

- Taaksepäin yhteensopiva

Yhteiskäyttöiset linkit edelleen OK

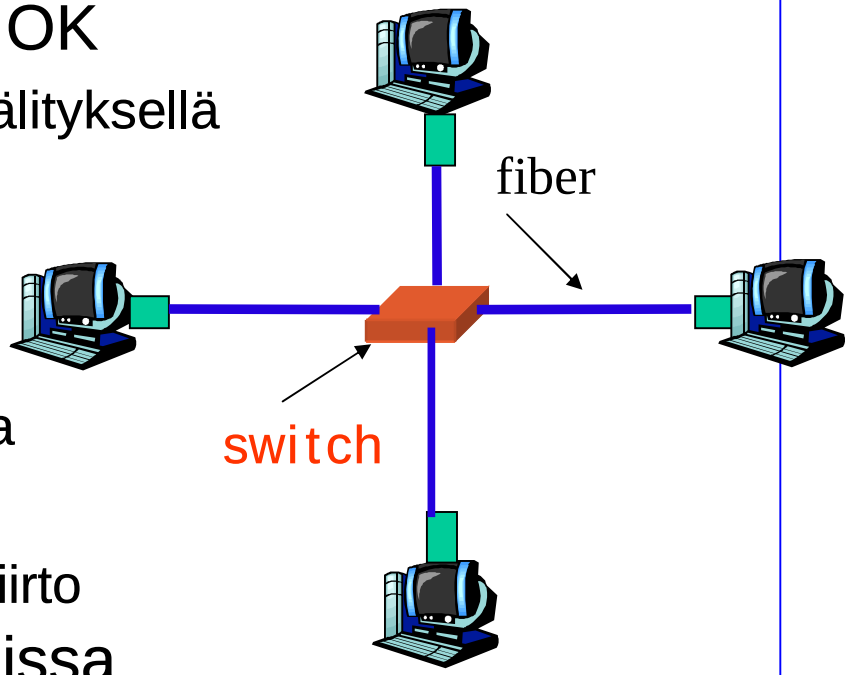
- Koneiden yhdistely keskittimen välityksellä
- CSMA/CD

Kaksipisteyhteydet

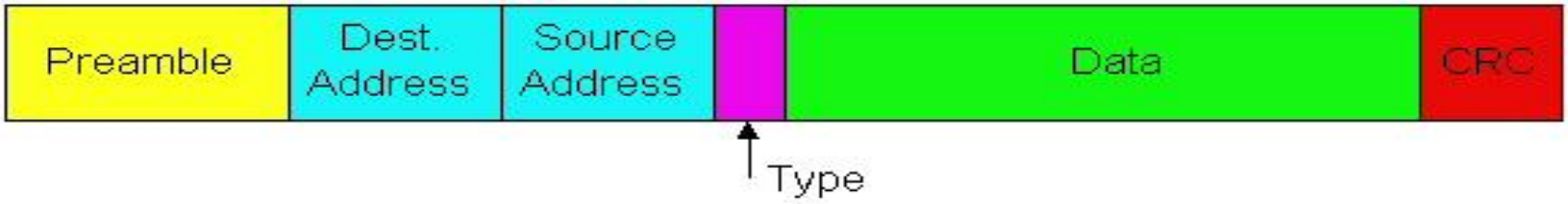
- ei törmäyksiä
- koneet yhdistetty **kytkimien** kautta
- pitkät välimatkat mahdollisia
- kaksisuuntainen täysivauhtinen siirto

Käytetään yleisesti runkoverkoissa

- verkkojen yhdistely (reititin -> reititin)
- valokaapeli, myös cat5/cat6 parikaapeli



Ethernet-kehys



Tahdistuskuvio (preample) (8 B)

- 7 tavussa 10101010 kellojen tahdistusta varten
- 8. tavu 10101011 kertoo varsinaisen kehyksen alkavan

Kohteen ja lähteen MAC-osoitteet (6 + 6 B)

Type (2 B)

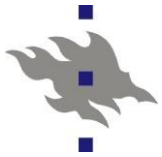
- verkkoprotokolla, jolle vastaanottaja luovuttaa kehyksen datan
- IP, ARP, jokin muu esim, Apple Talk, Novell IPX, ..

Data (46 ... 1500 B)

- Ethernet MTU = 1500 B

CRC (4 B eli 32 bittiä)

- tarkistusbitit, tahdistuskuvio mukana laskennassa

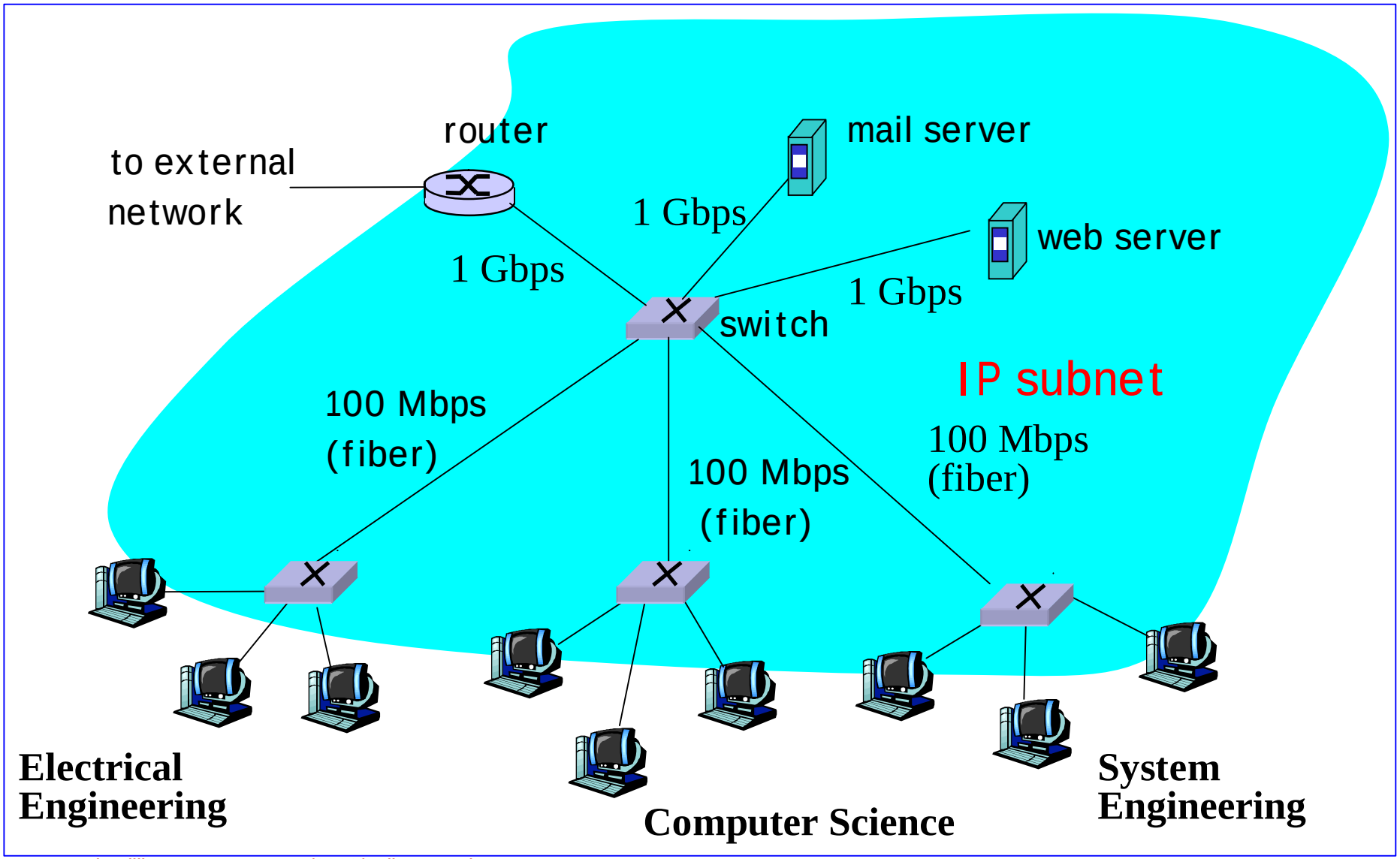


Ethernet varaus: CSMA/CD

(klassinen Ethernet-verkko on yleislähetysverkko!)

- Carrier Sense
  - Kuuntele, onko väylä vapaa (96 b:n ajan)
  - Jos vapaa, lähetä heti
  - Muuten odota ja lähetä, kun linja vapautuu
- Collision Detection
  - Kun lähetetty, kuuntele onnistuiko
- Törmäys?
  - Huomaa signaalin voimakkuudesta
  - Lopeta kehyksen lähetys heti
  - Lähetä 48 bitin sotkusignaali (jam): muutkin huomaavat varmasti
- Random Access
  - Odota törmäyksen jälkeen satunnainen aika

LAN, verkkosegmentit



Takaperin oppiminen: Kytkentätaulu (switching table)

- Aluksi taulu on tyhjä
- Saapuva kehys
  - Lähteen MAC-osoite x, kohteen MAC-osoite y, tuloportti p, yms
- Lähde X ei ole taulussa
  - Lisää (X, p ,TTL) tauluun eli kytkin oppii, että osoite X on saavutettavissa portin p kautta
- Lähde X on taulussa => päivitä TTL
- Kohde Y ei ole taulussa
  - Lähetetään kehys kaikkiin muihin portteihin = tulvitus (flooding)
  - Opitaan myöhemmin Y:n oikea portti jostain sen lähettämästä kehyksestä
- Lähde X ja kohde Y ovat jo taulussa
  - X ja Y samassa portisssa => hylkää kehys (on jo oikeassa aliverkossa)
  - X ja Y eri porteissa => lähetä kehys Y:n porttiin