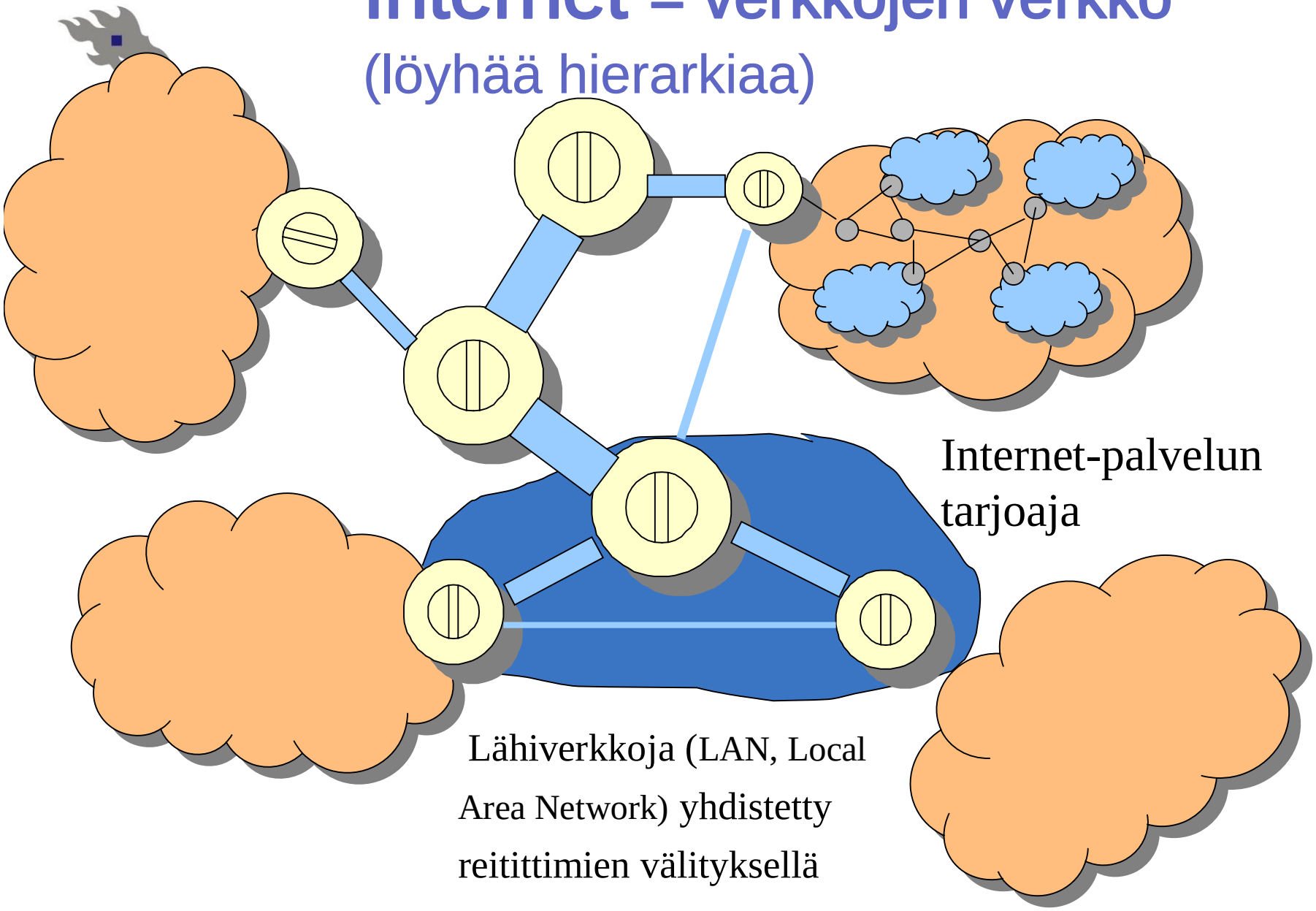
Tietoliikenteen perusteet

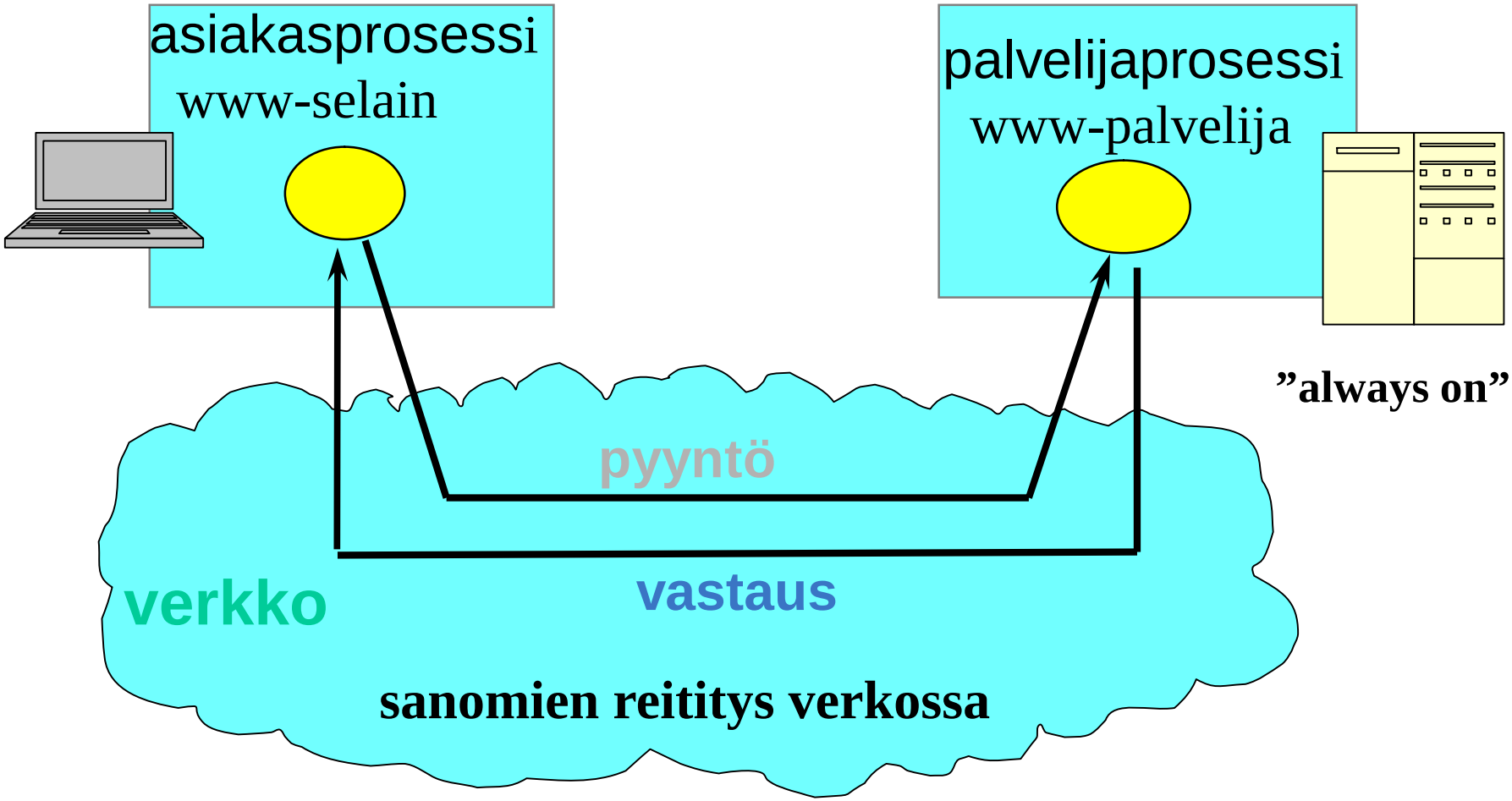
Vähän kertausta

Internet = verkkojen verkko

(löyhää hierarkiaa)



Asiakas-palvelija-malli /vertaistoimijamalli



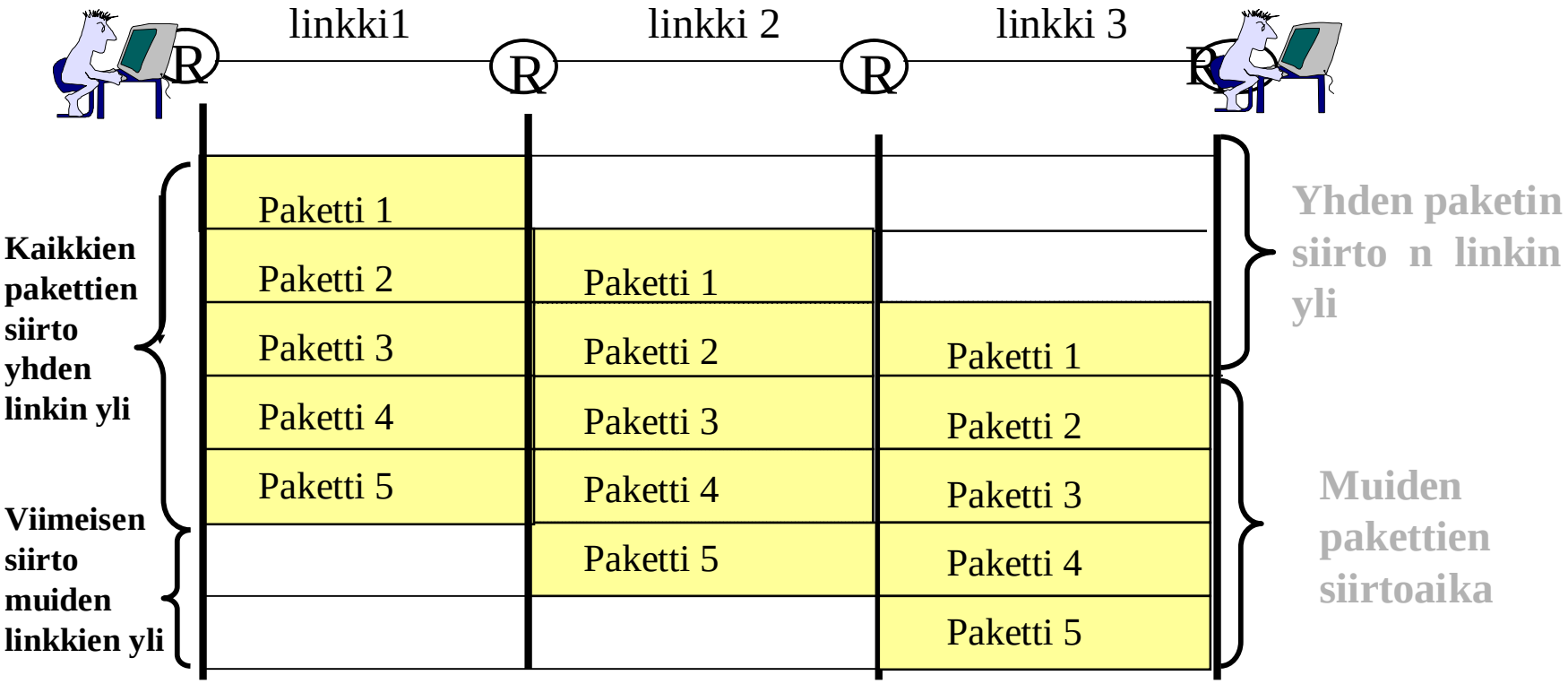
Oikea kone, oikea prosessi

Pakettivälitys siirto-aika

Olkoon siirtoaika a:

a) ka+(n-1)a = (k+n-1)a

b) na +(k-1)a = (n+k-1)a



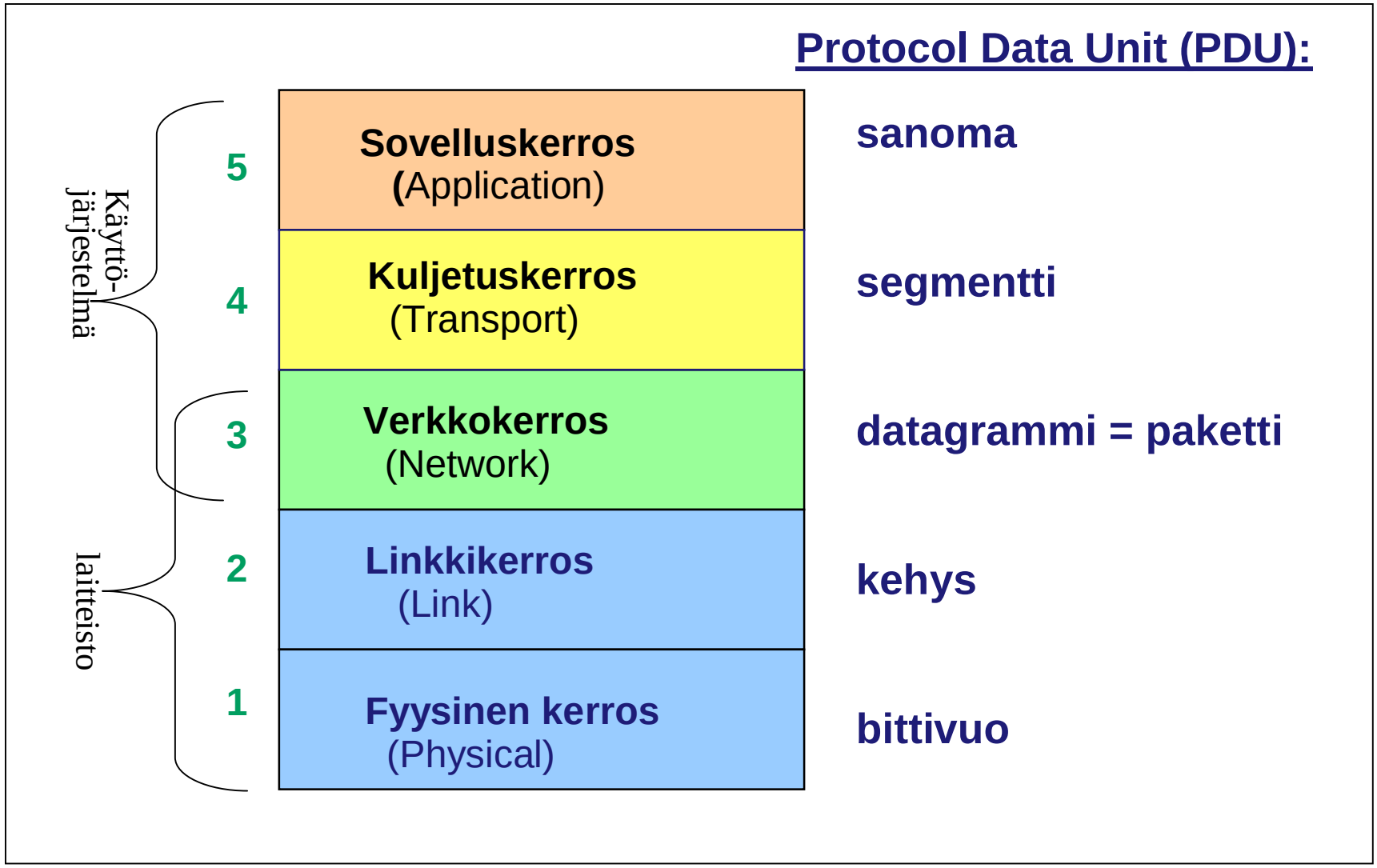
Sanoman siirtoaika, kun sanomassa on k pakettia ja linkkejä on n kappaletta

a) k:n paketin siirto 1. linkin yli + viimeisen paketin siirto n-1 linkin yli.

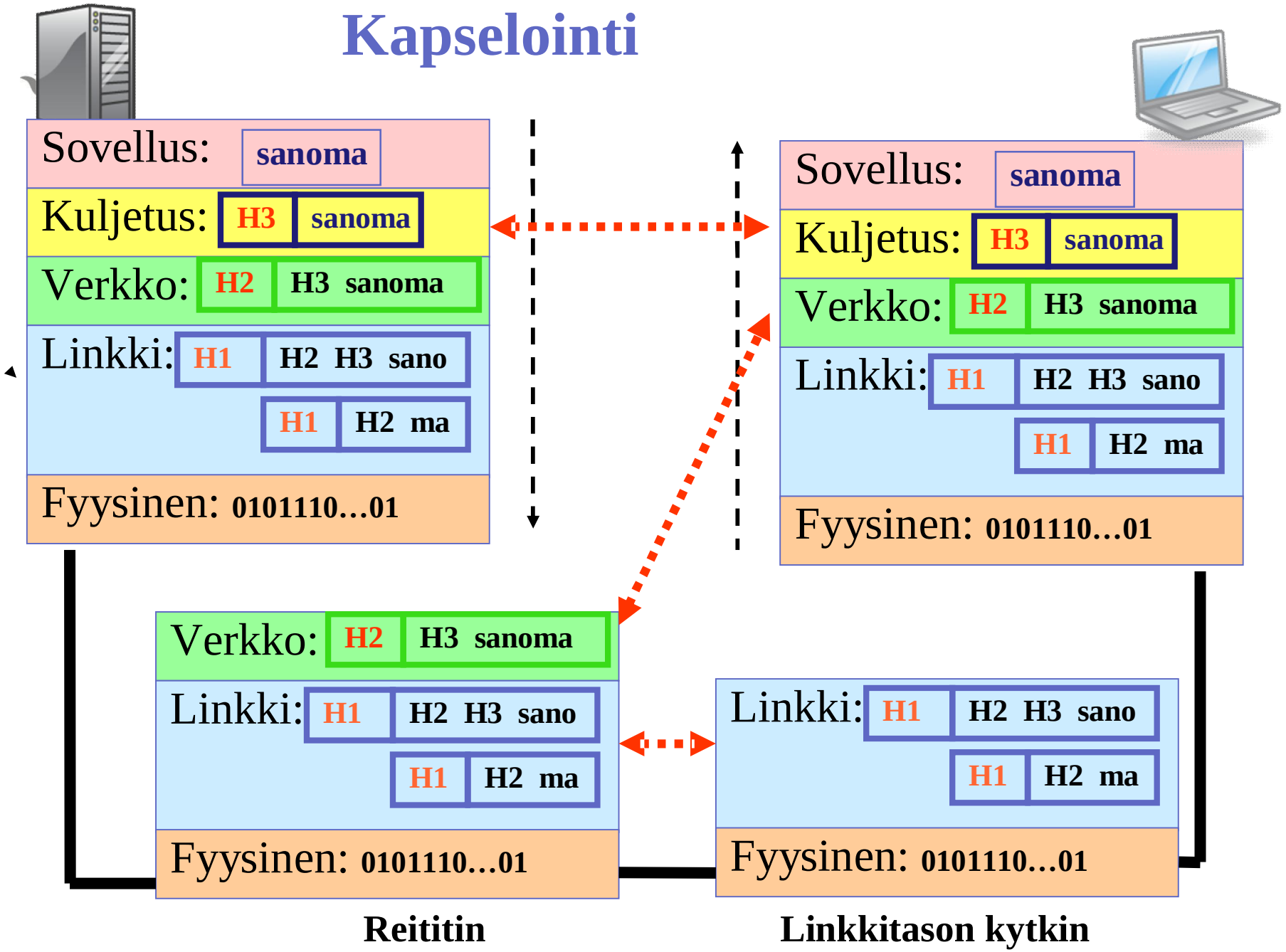
b) 1. paketin siirto n:n linkin yli + muiden k-1 paketin siirto yhden linkin li

Animaatio: http://wps.aw.com/aw_kurose_network_4/63/16303/4173750.cw/index.html

Internet-protokollapino

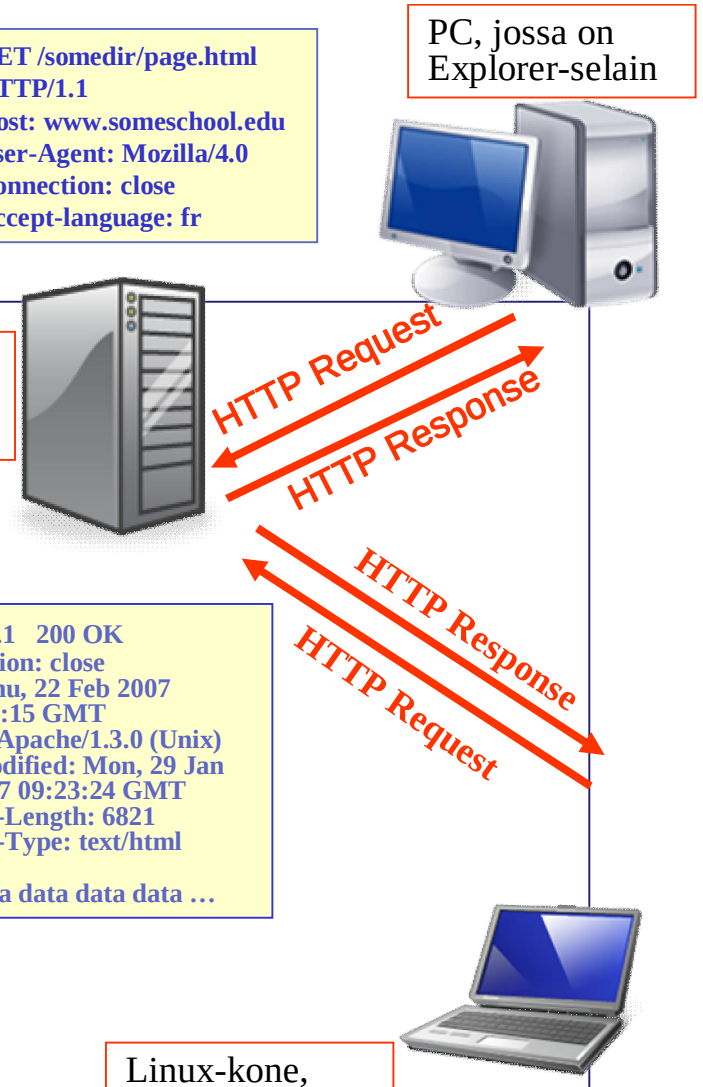


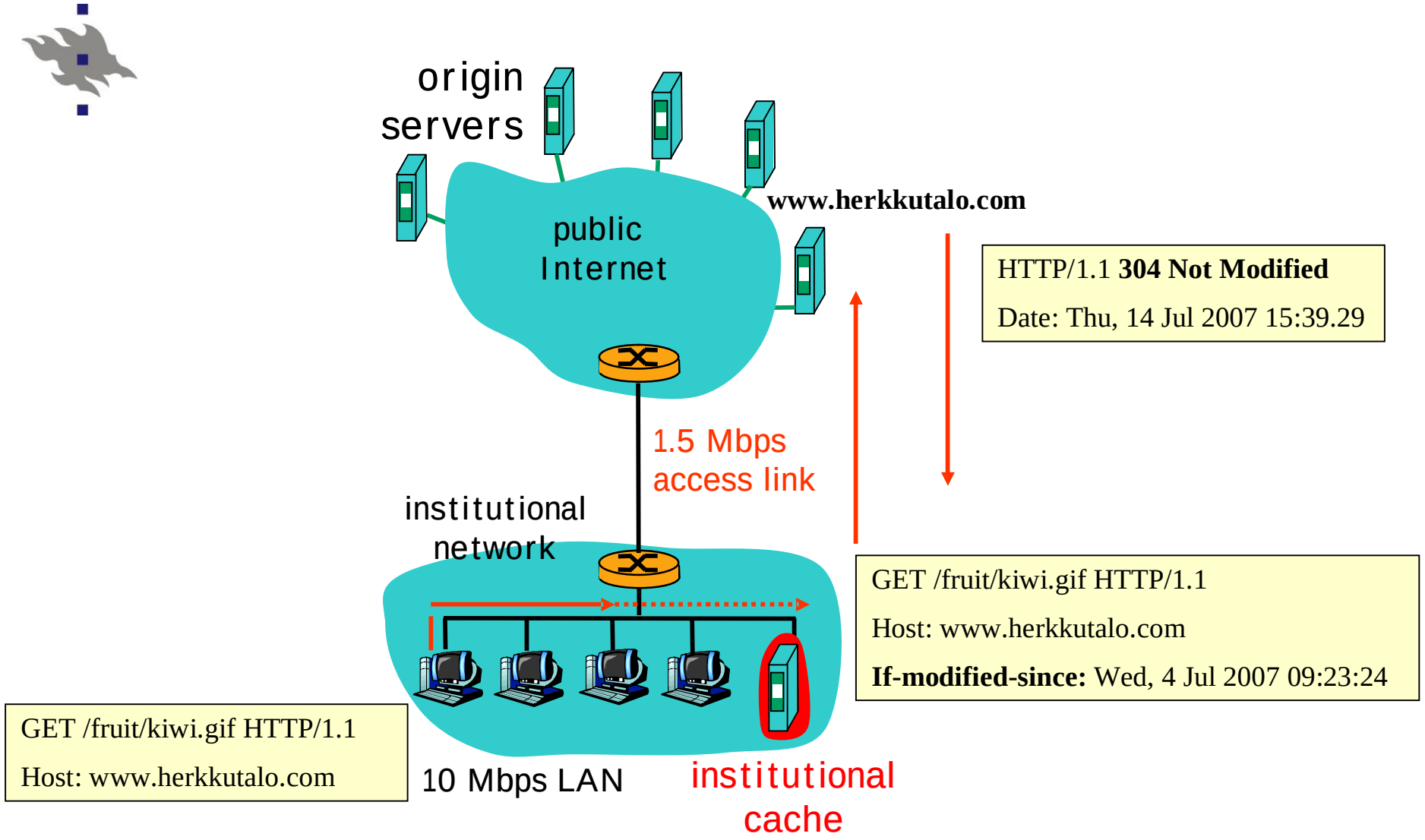
Kapselointi



HTTP (HyperText Transfer Protocol)

- **WWW:N sovellusprotokolla**
  - **Tekstimuotoiset sanomat**
    - pyyntö – vastaus
- **Asiakas**
  - Selain: **FireFox, Internet Explorer, Opera, Apple Safari, …**
  - pyytää, noutaa ja näyttää objektit
- **Palvelija**
  - etsii objektin (tiedoston) koneen hakemistosta
  - ja lähettää sen vastauksena asiakkaalle
- **Tilaton protokolla**
  - Palvelija ei muista mitään edellisistä pyynnöistä => evästeet (cookies)

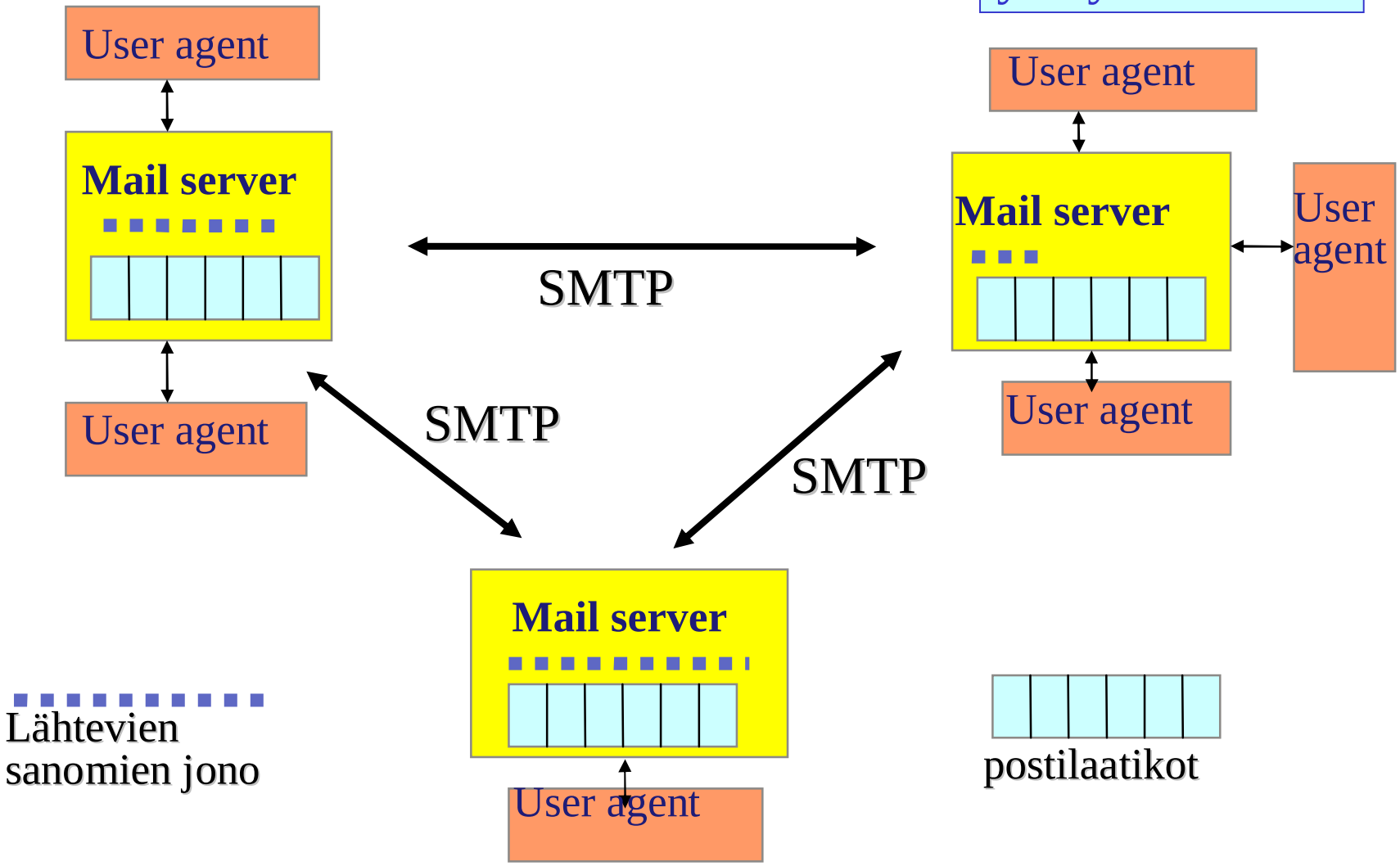



origin
servers
public
Internet
www.herkkutalo.com
HTTP/1.1 304 Not Modified
Date: Thu, 14 Jul 2007 15:39.29
1.5 Mbps
access link
institutional
network
GET /fruit/kiwi.gif HTTP/1.1
Host: www.herkkutalo.com
If-modified-since: Wed, 4 Jul 2007 09:23:24
GET /fruit/kiwi.gif HTTP/1.1
Host: www.herkkutalo.com
10 Mbps LAN
institutional
cache

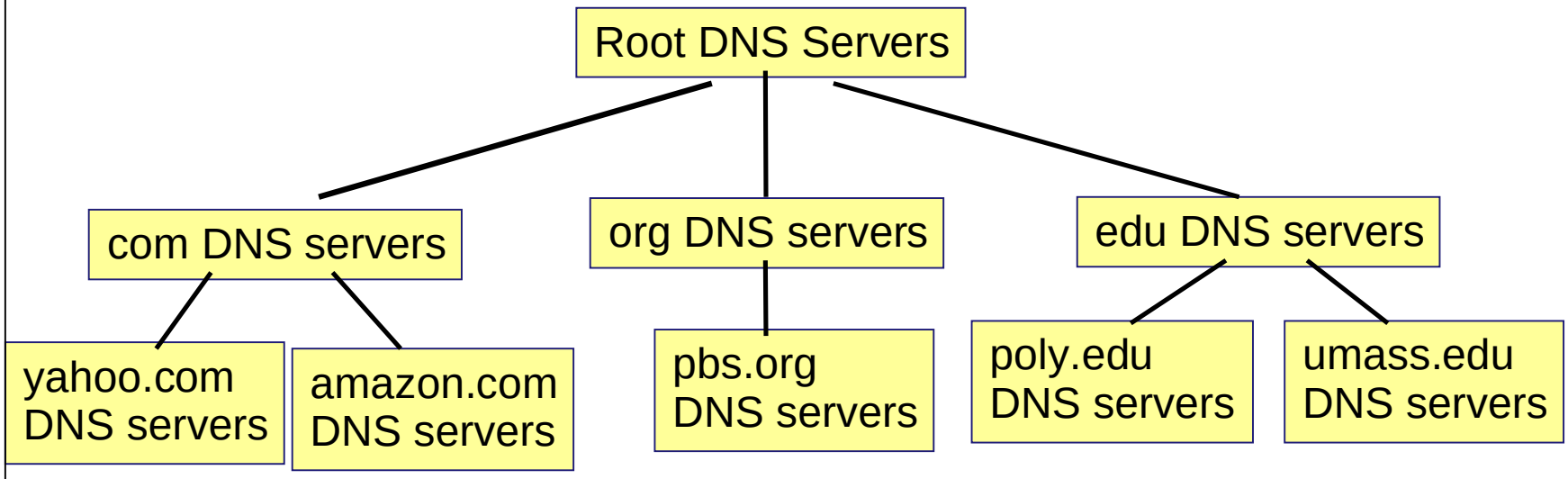
Sähköpostin komponentit

Push-protokolla

yhteydellinen



Hajautettu, hierarkinen tietokanta



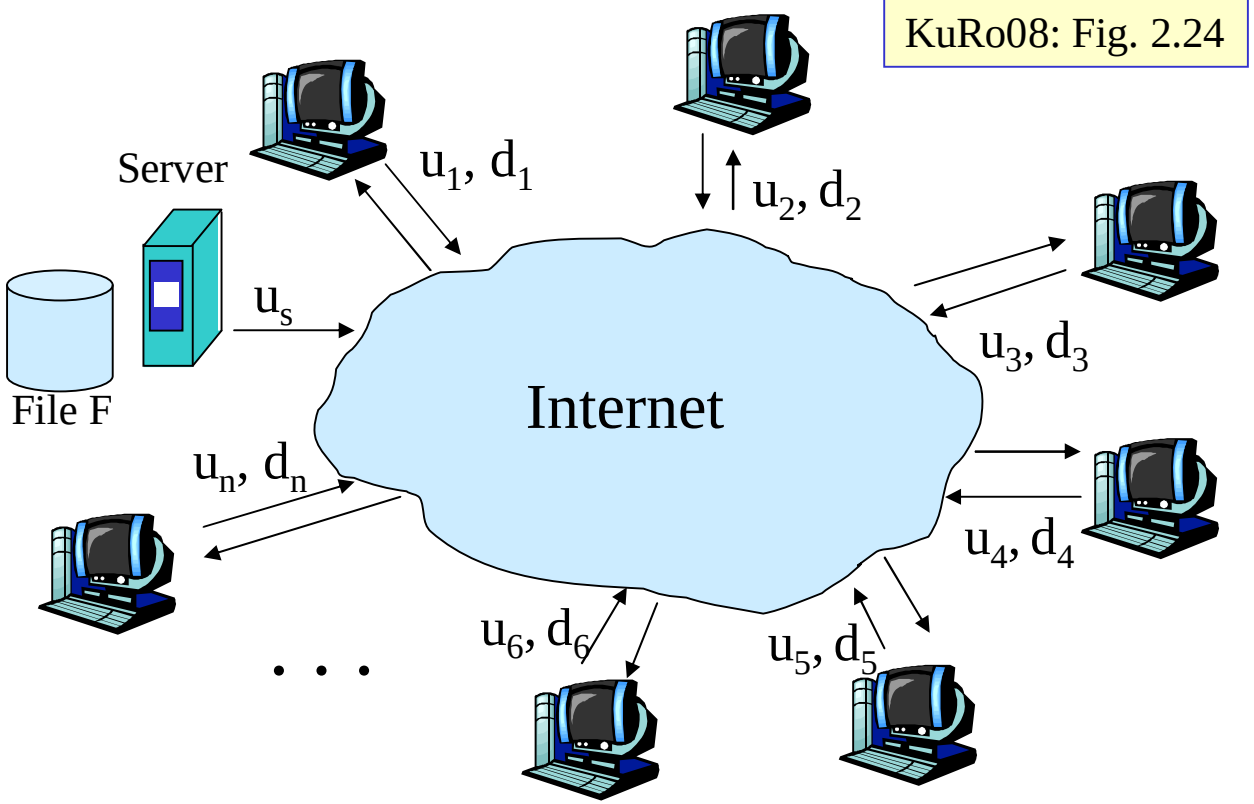
KuRo05: Fig 2.18

- **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

Skaalautuvuus

KuRo08: Fig. 2.24



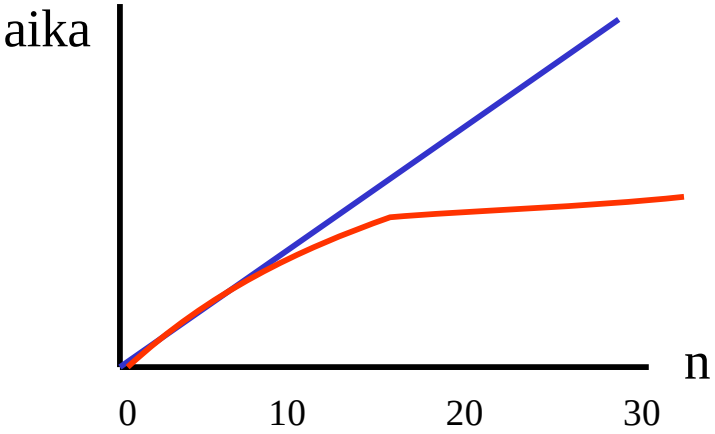
Asiakas-palvelinmalli:

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.



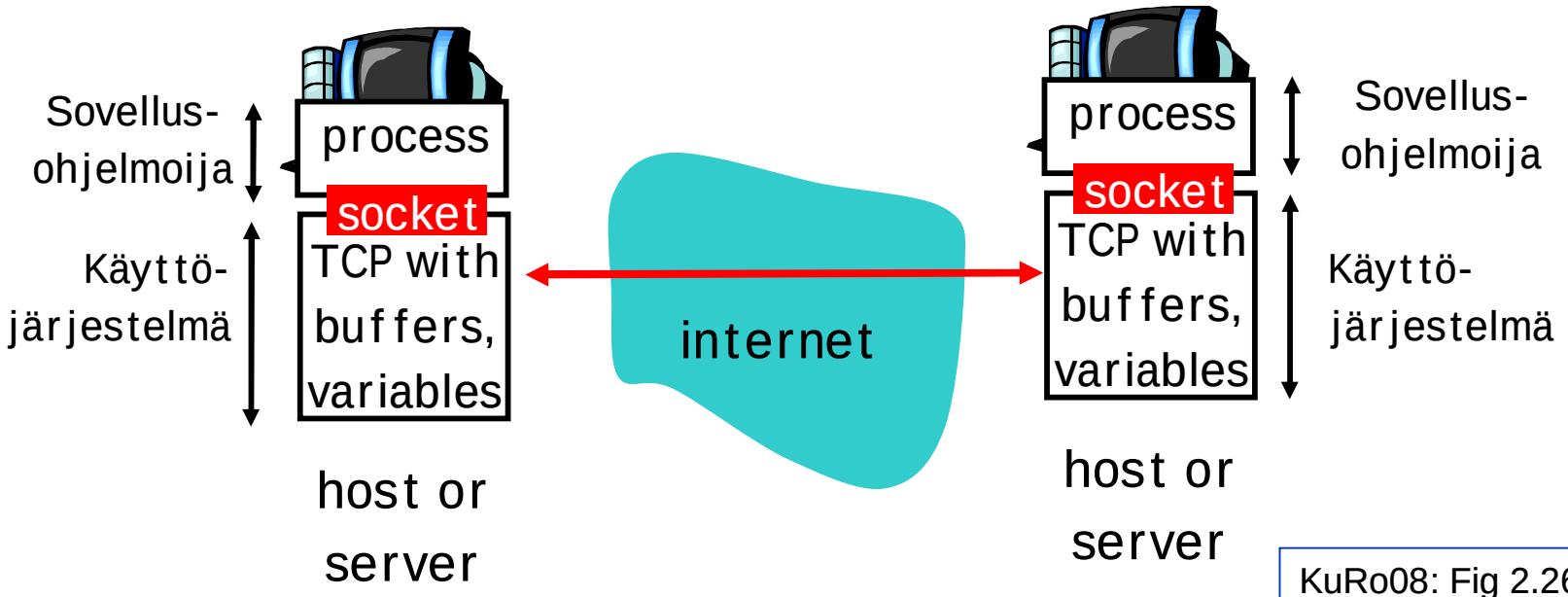
Vertaistoimijamalli (alussa tiedosto on palvelimella)

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

Summamerkki

Pistoke (socket)

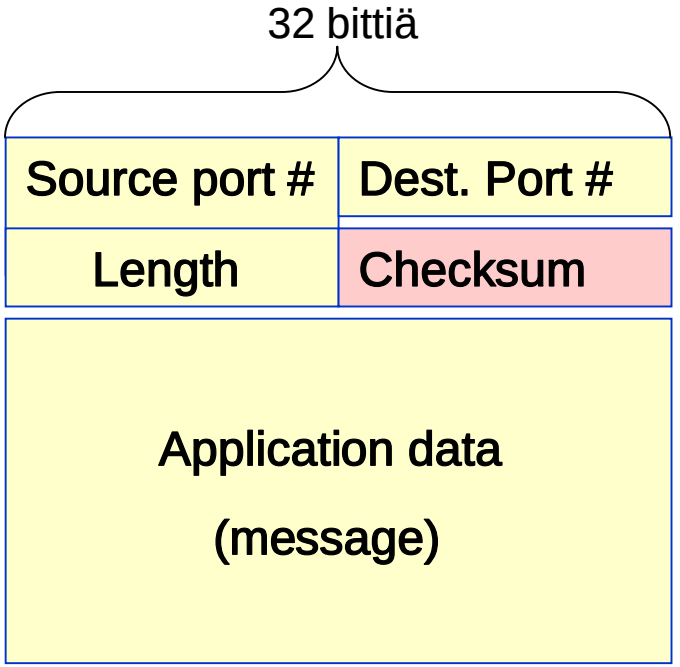
- Kuljetuspalvelun ja sitä käyttävän sovelluksen rajapinta isäntäkoneessa
  - Sovelluksen tietoliikenne = KJ:n palvelupyyntöjä
  - Pistoke on “palveluluukku”
- **Alunperin Berkeley UNIXin (BSD) mukana**



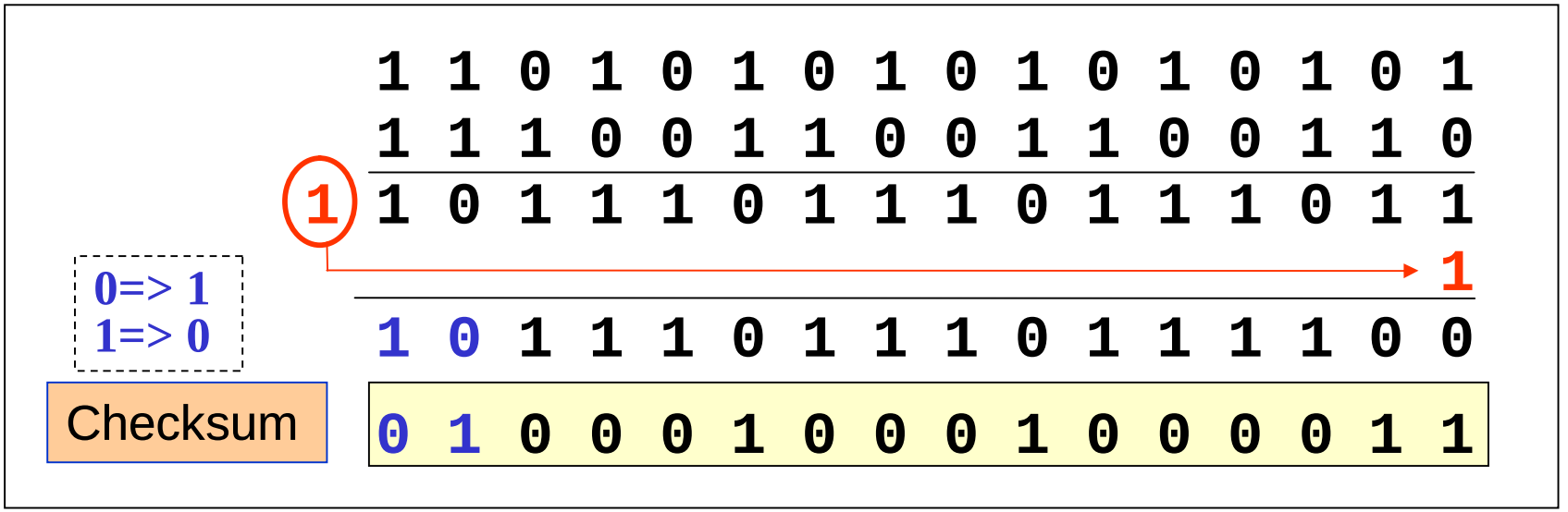
KuRo08: Fig 2.26

UDP: Tarkistussumma

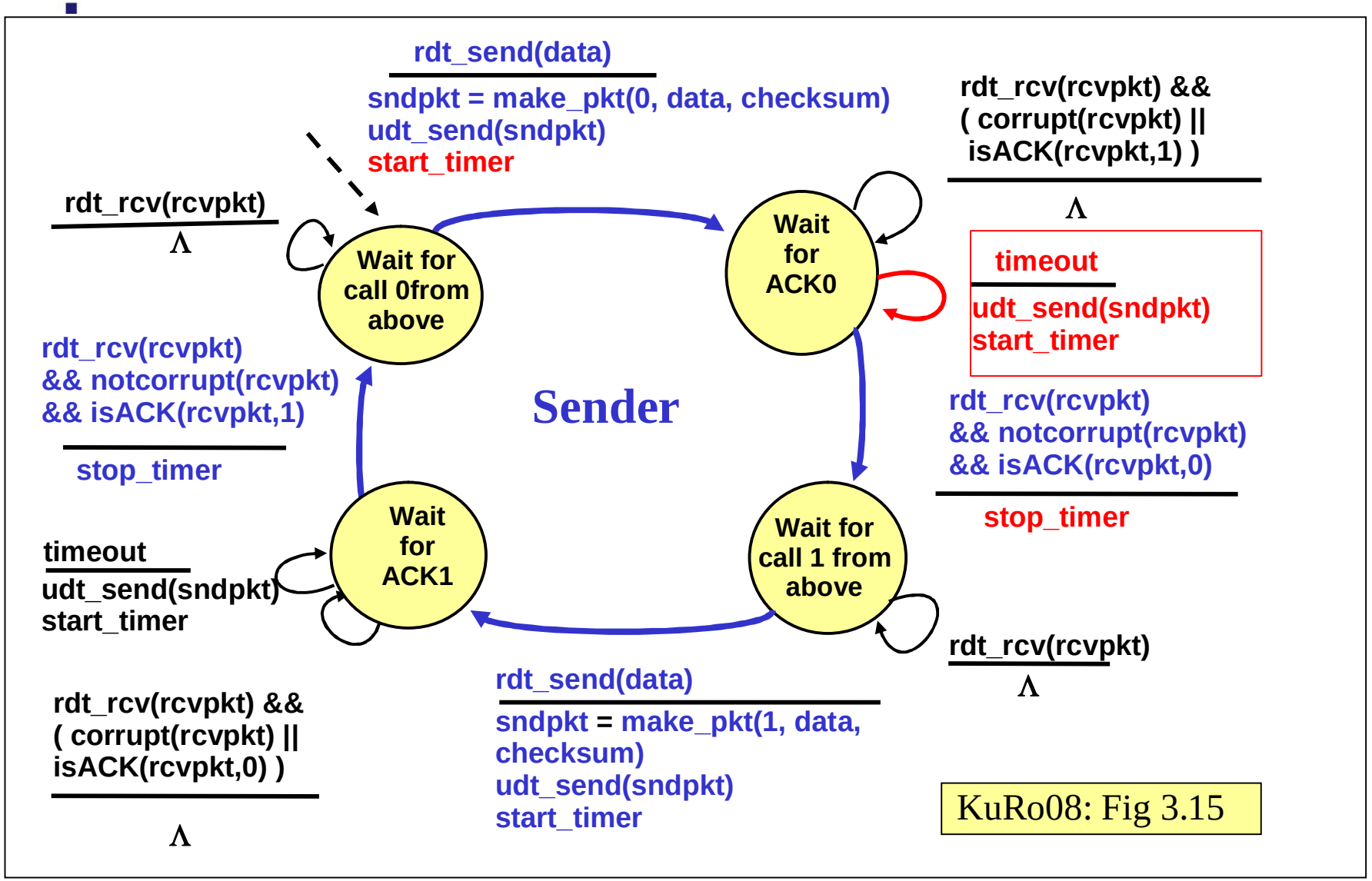
- Lähetys
  - Summaa 16 bitin kokonaisuudet (otsake + pseudo-otsake mukana), ylivuotobitit lasketaan mukaan, talleta yhden komplementtina
- Vastaanotto
  - Summaa 16 b kokonaisuudet (myös tarkistussumma).
  - Jos tuloksena on 16 ykköstä, niin OK!



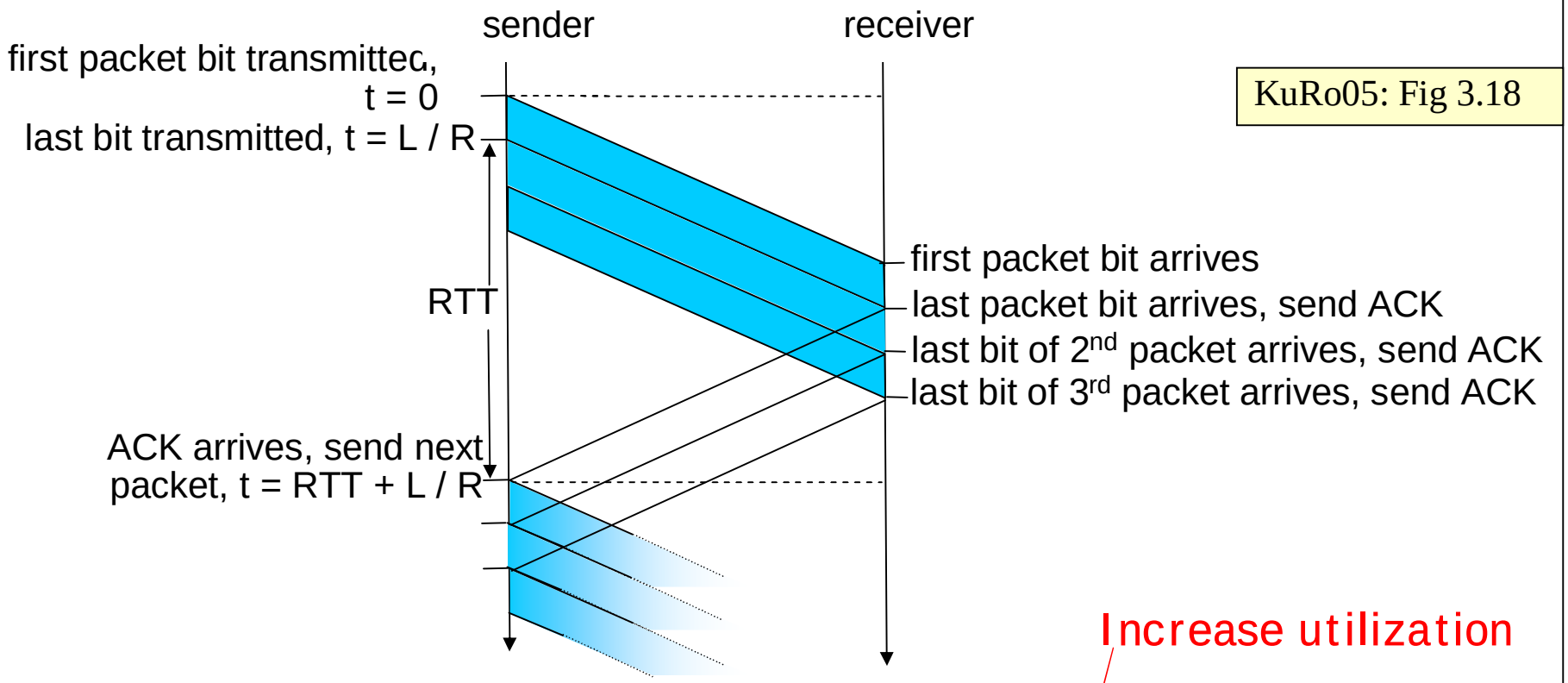
UDP-otsake



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

Liukuhihnoitus: käyttöasteen kasvattaminen

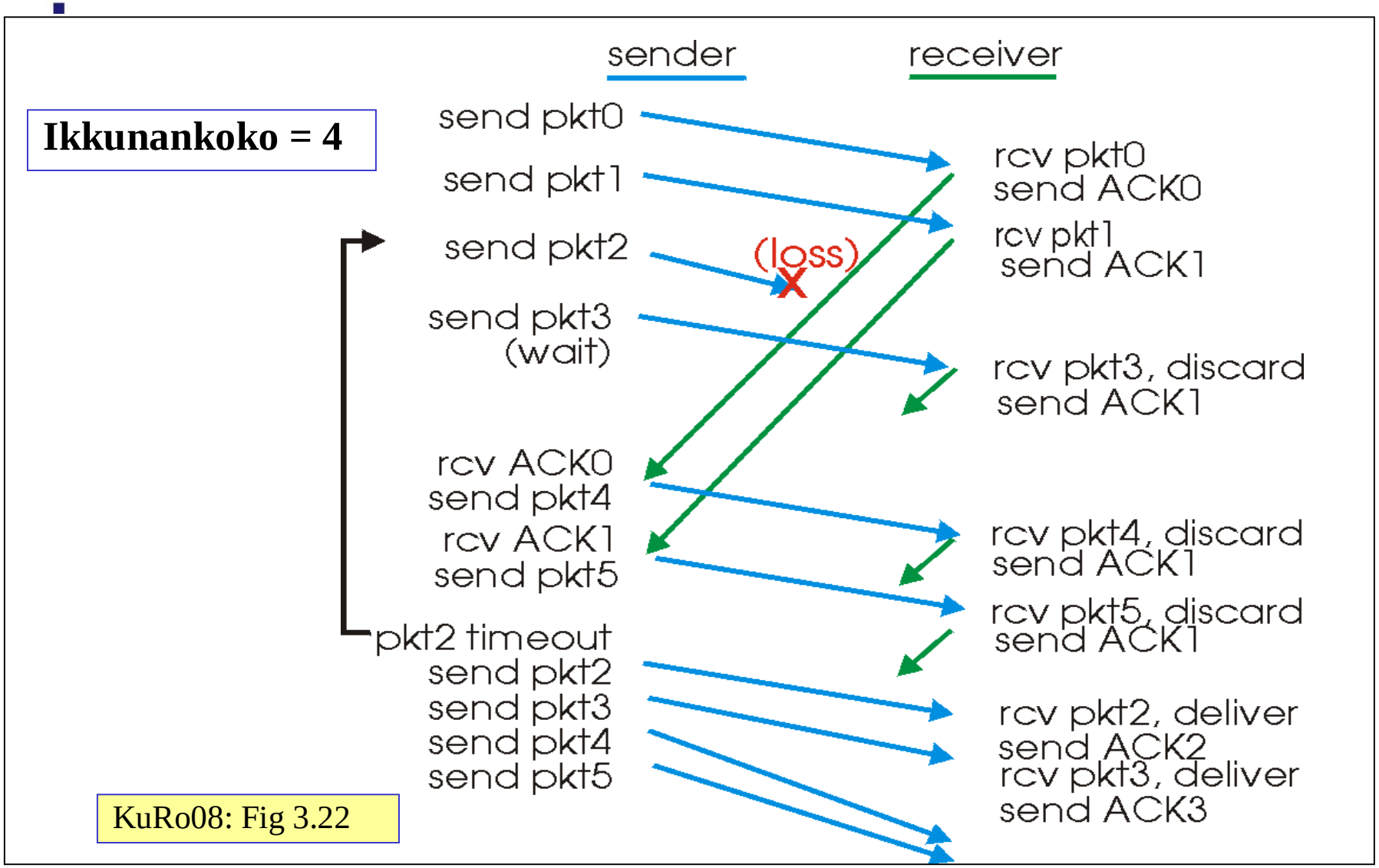


KuRo05: Fig 3.18

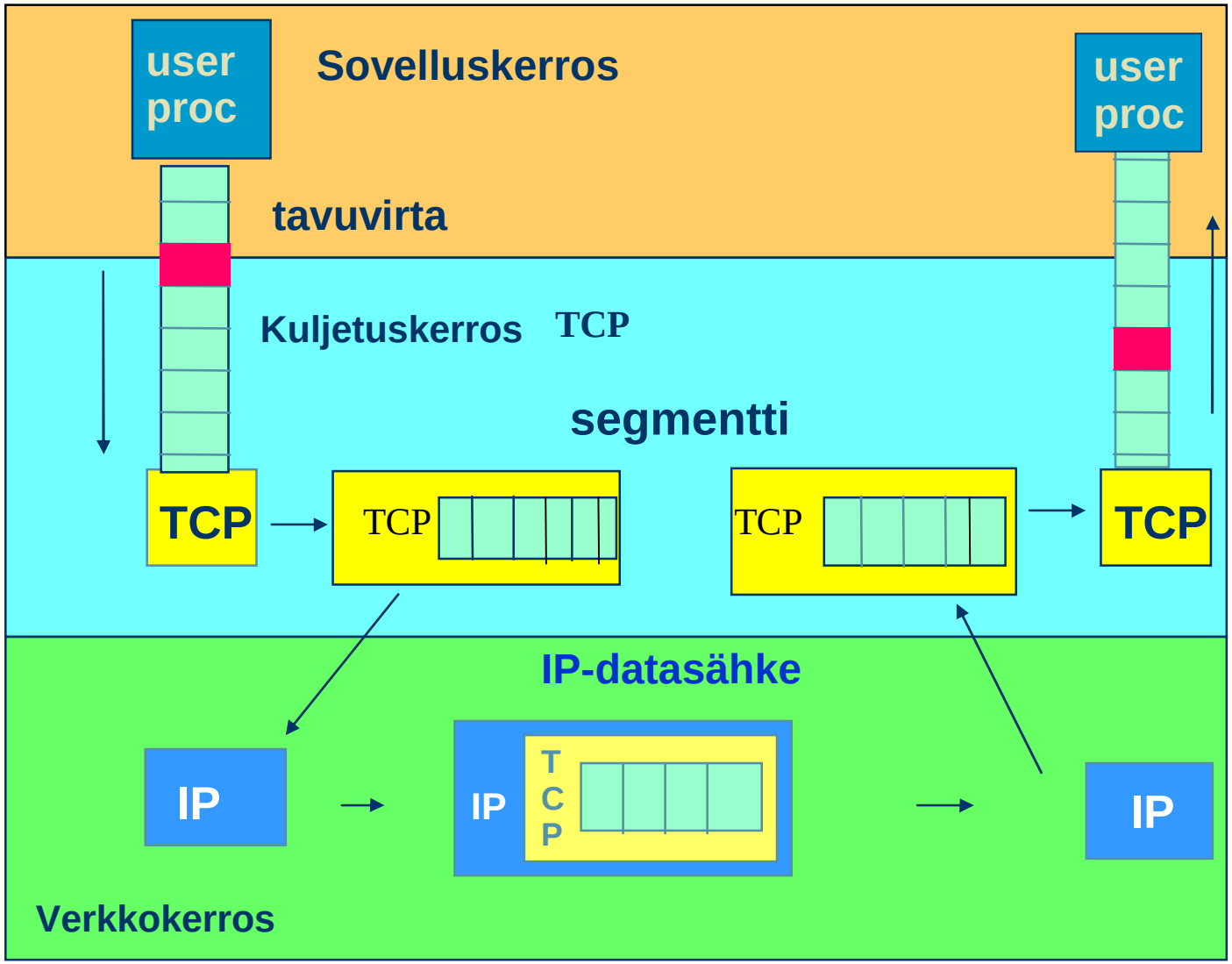
Increase utilization by a factor of 3!

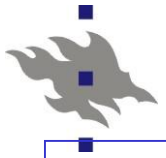
$$U_{sender} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

Go-Back-N: Esimerkki



TCP: prosessilta prosessille -tavuvirta





TCP:

- **Luotettava, järjestyksen säilyttävä tavuvirta**
  - Ei sanomarajoja
  - Tavunumerointi
  - Checksum-tarkistus
  - Puskurointi uudelleenlähetystä varten
  - Kumulatiiviset kuittaukset
- Yhteydellinen
  - Kolminkertainen kättely, yhteyden purku
- Vuonvalvonta, ruuhkanhallinta (-valvonta)
  - Lähettäjä ei saa tukahduttaa vastaanottajaa eikä reitittimiä
  - Vuonvalvonta: Receive window
  - Ruuhkanhallinta

Yhteyden muodostus

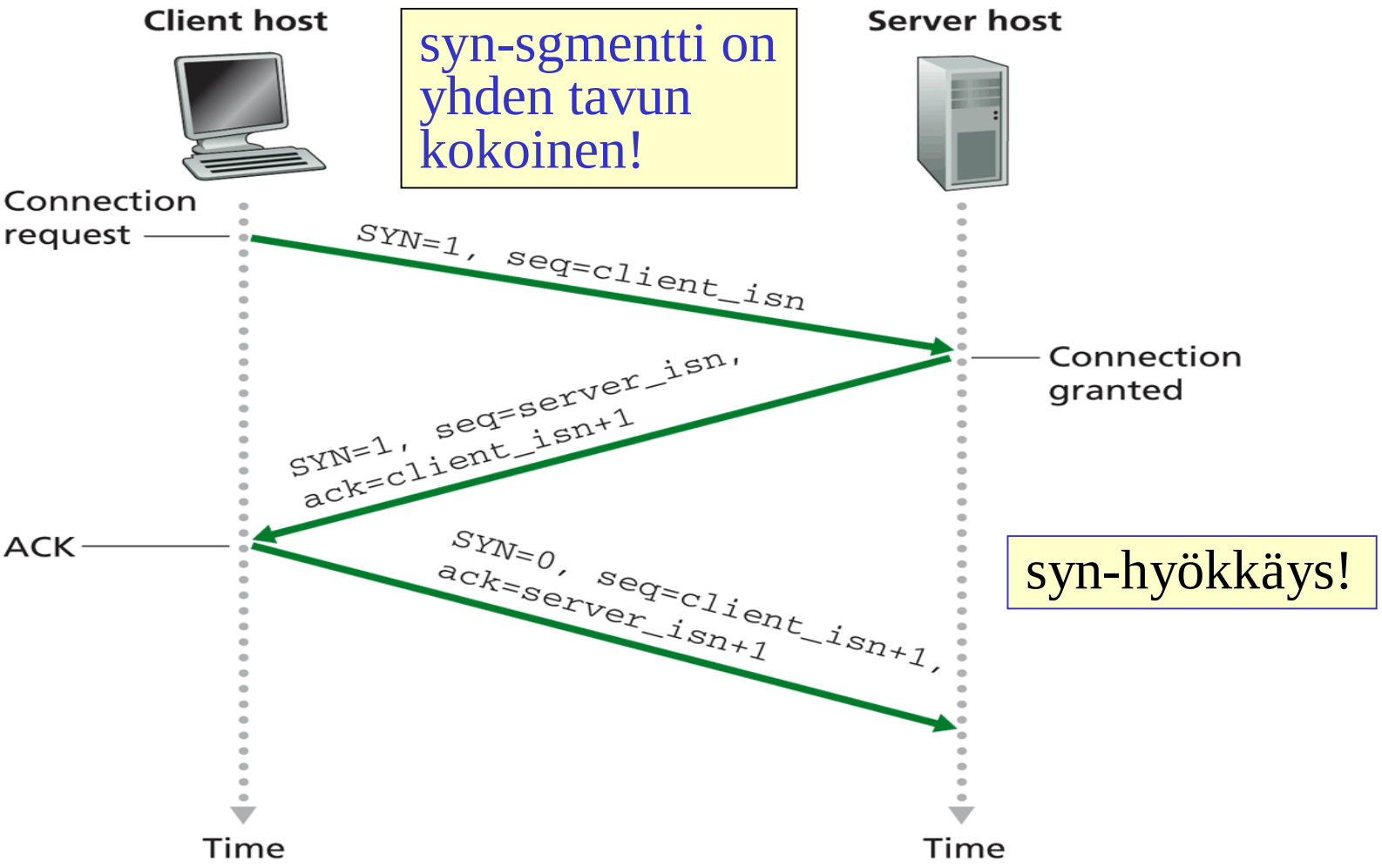


Figure 3.39 ◆ TCP three-way handshake: segment exchange

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

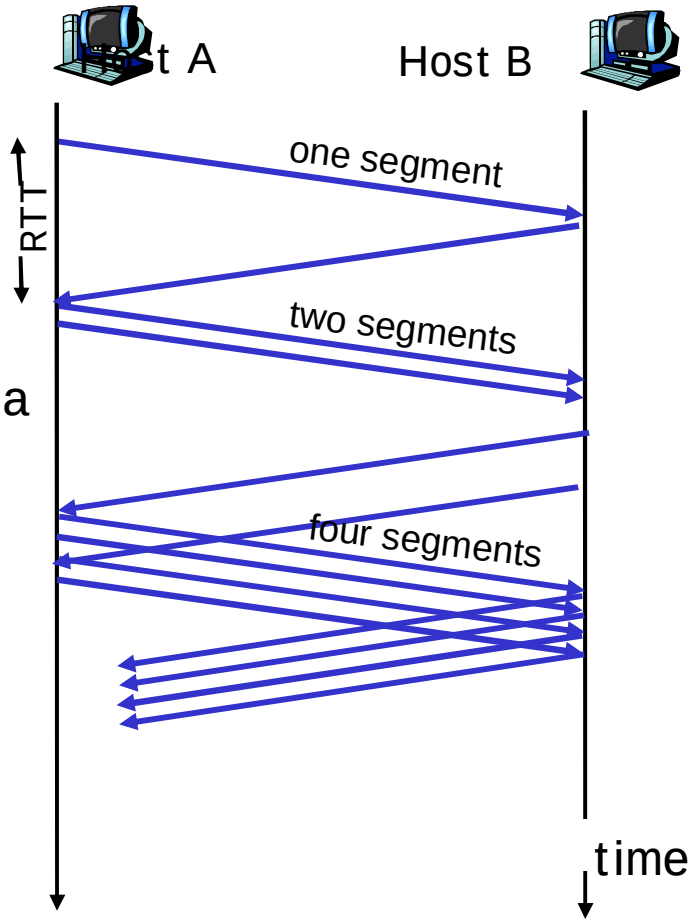
- **Aluksi ruuhkaikkuna = yksi segmentti**
  - Alussa hidas siirtonopeus= MSS/RTT
- **Kukin kuittaus kasvattaa yhdellä ruuhkaikkunan kokoa**
  - Eksponentiaalinen kasvu
  - Ikkuna kaksinkertaistuu yhden RTT:n aikana

hidas aloitus

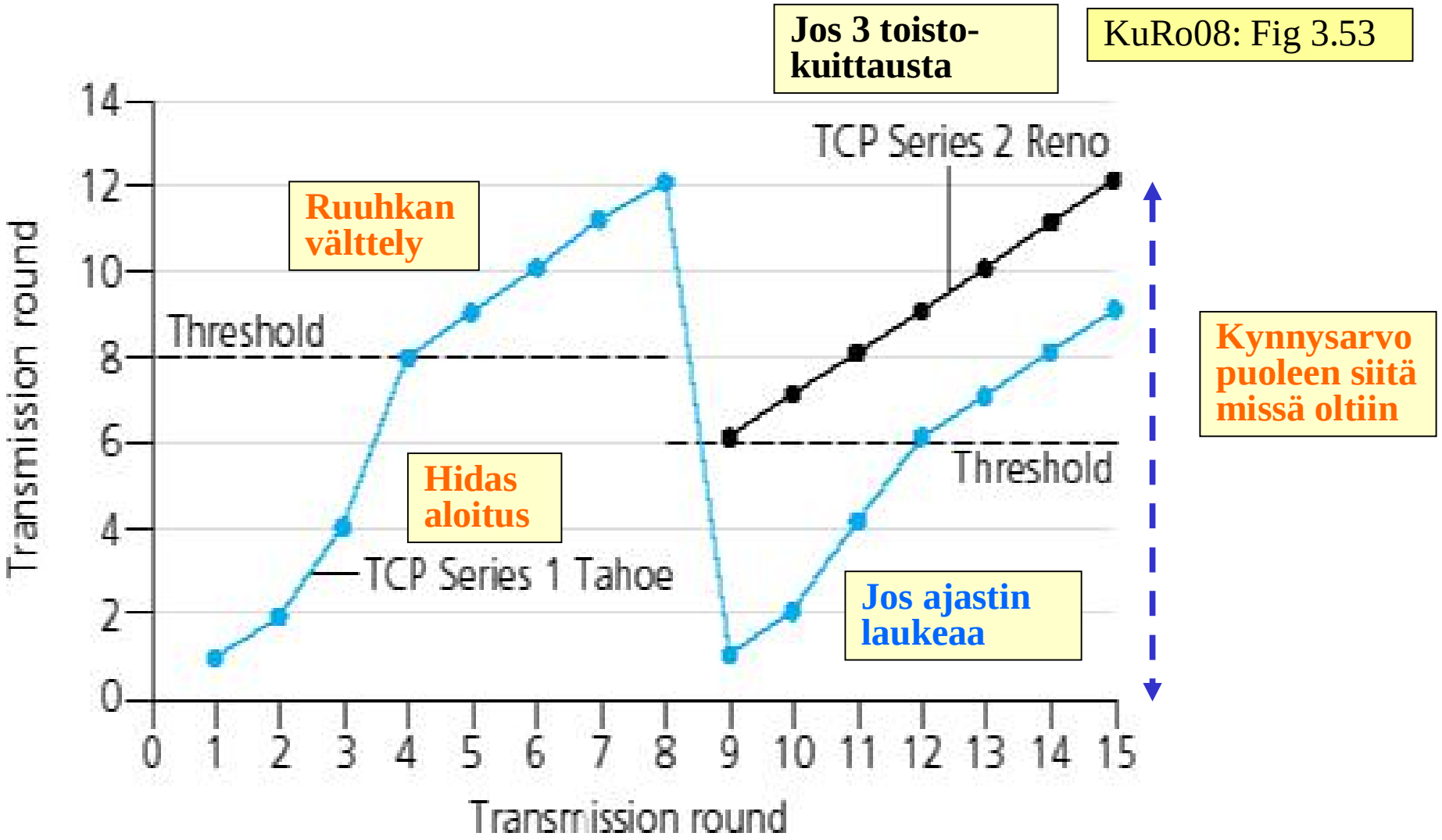
- **Jos uudelleenlähetys, puolita ruuhkaikkunan koko**
  - Multiplicative decrease
- **Sen jälkeen kasvata ikkunaa yksi segmentti/RTT**
  - Lineaarinen kasvu (Additive increase)
  - Ruuhkan välttely (congestion avoidance)

ruuhkanvälttely

- Siirtonopeus = CognWin / RTT tavua/sek

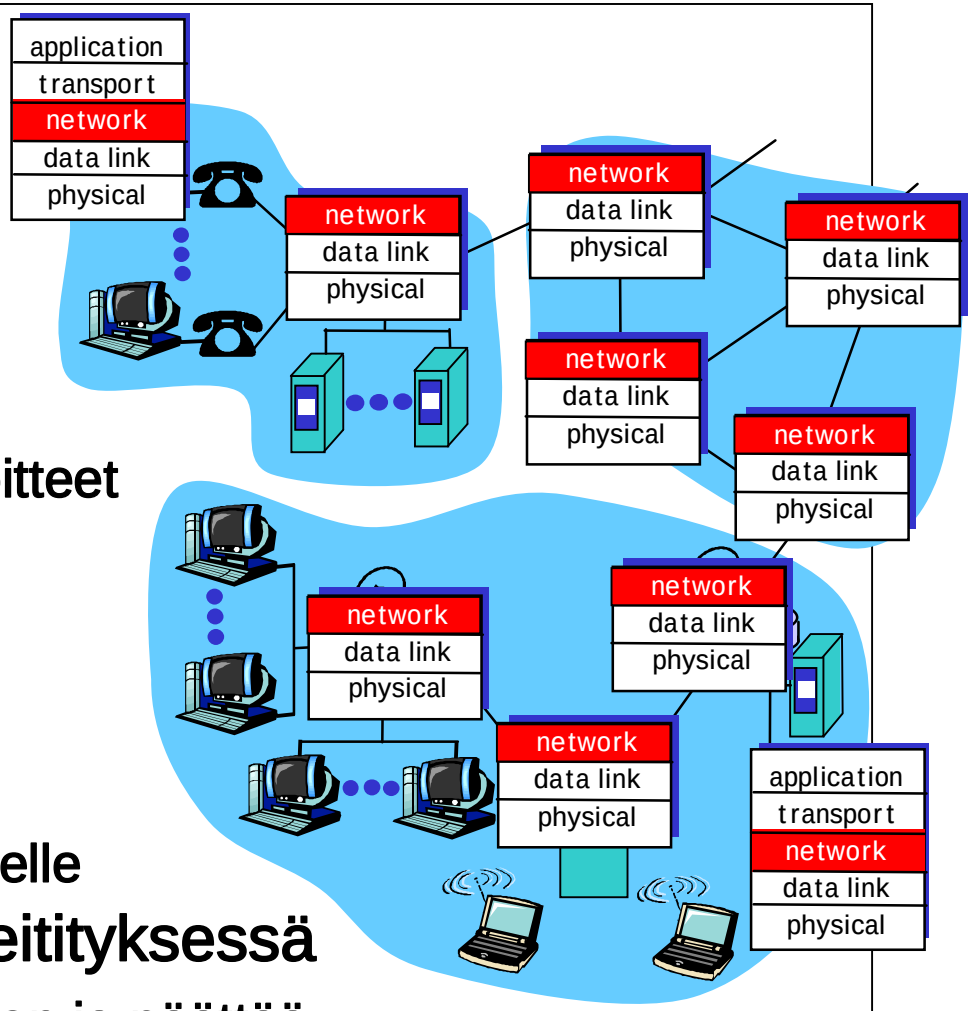


TCP Tahoe vs. TCP Reno



Verkkokerros

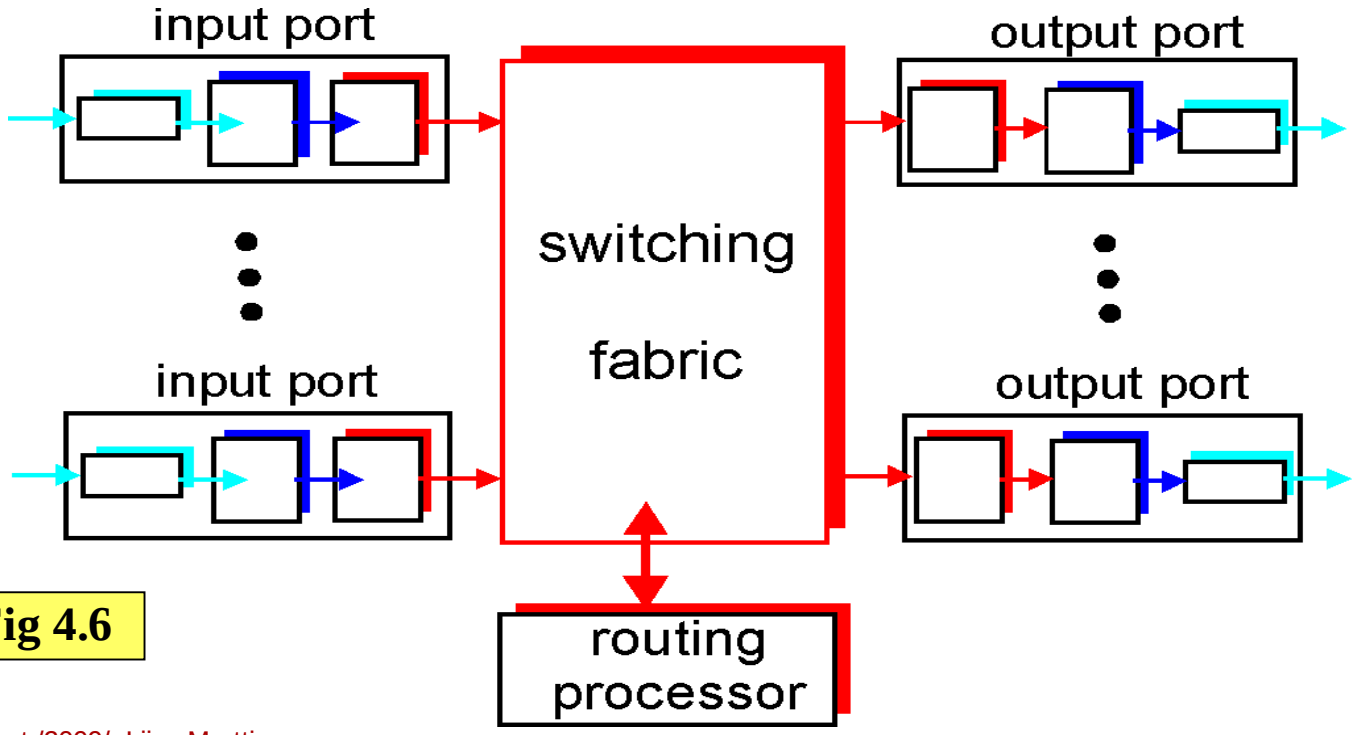
- Toimittaa kuljetuskerroksen segmentit vastaanottajalle
- Lähettäjä
  - luo segmenteistä verkkokerroksen IP-paketteja
  - Lisää otsaketietoja: mm. IP-osoitteet
- Reitittäminen
  - Isäntä – reititin ... reititin – isäntä
- Vastaanotto
  - Poista otsake
  - Anna segmentti kuljetuskerrokselle
- Verkkokerros toimii etenkin reitityksessä
  - Reititin tutkii IP-paketin otsakkeen ja päättää, mihin linkkiin se lähetetään seuraavaksi



KuRo08: Fig 4.1

Reitittimen arkkitehtuuri

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



KuRo08:Fig 4.6

IP-paketin rakenne (IPv4)

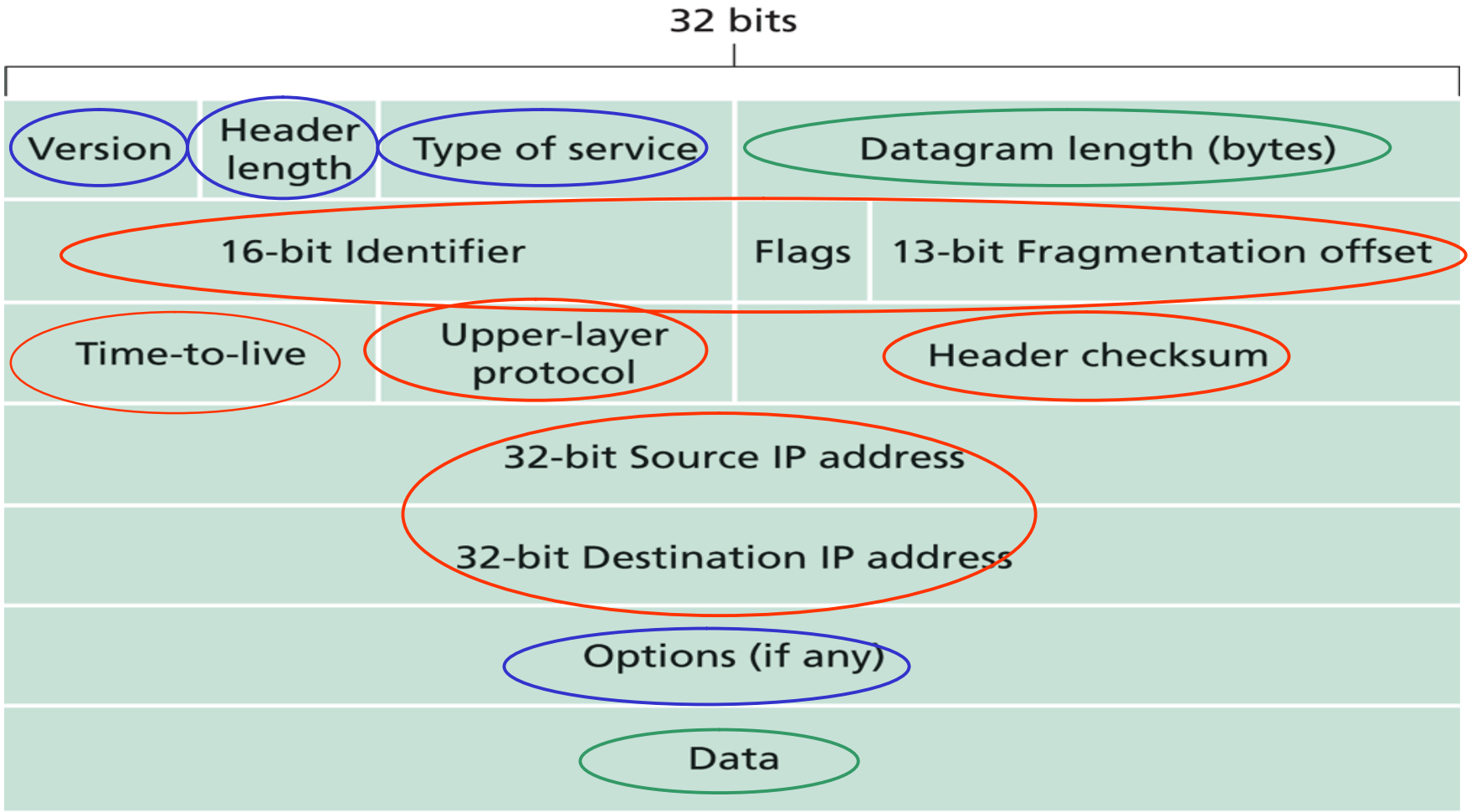


Figure 4.13 ♦ IPv4 datagram format

IP-pakettien paloittelu (fragmentointi)

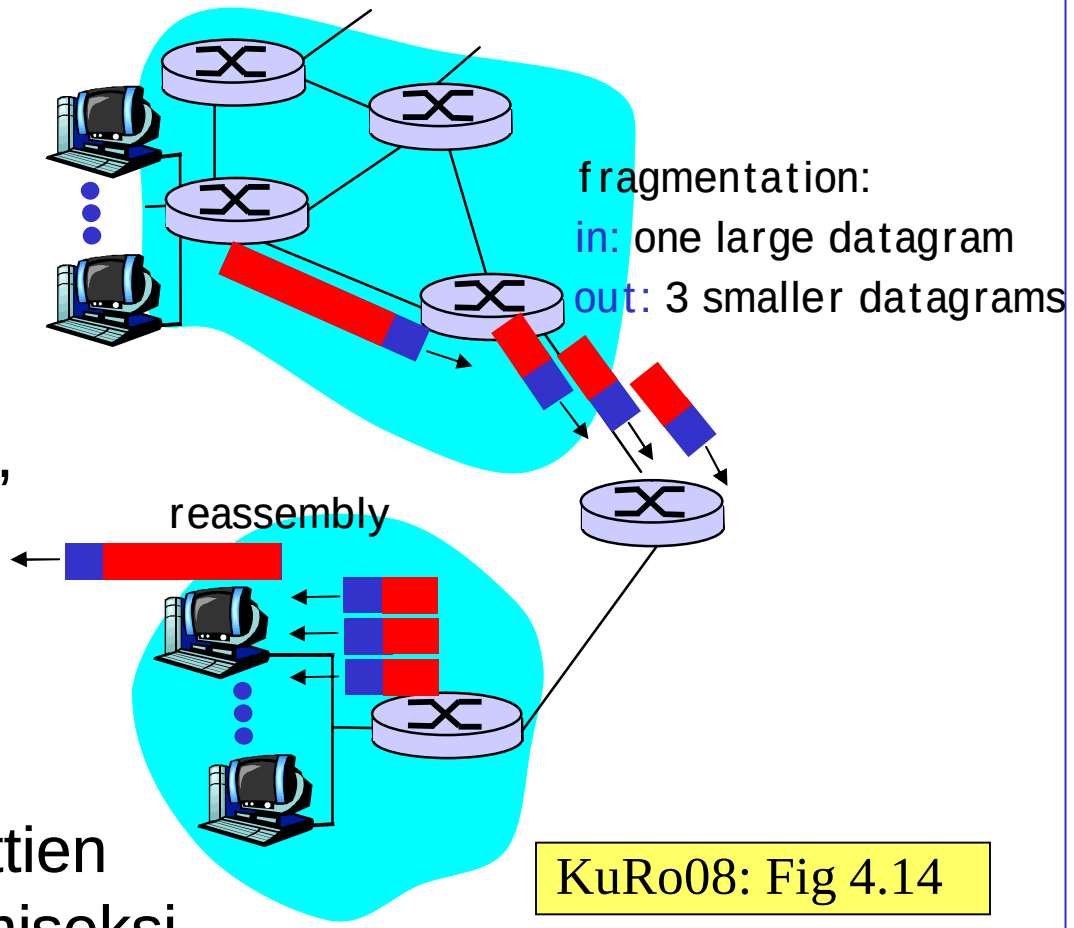
Maximum transfer Unit (MTU)

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

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

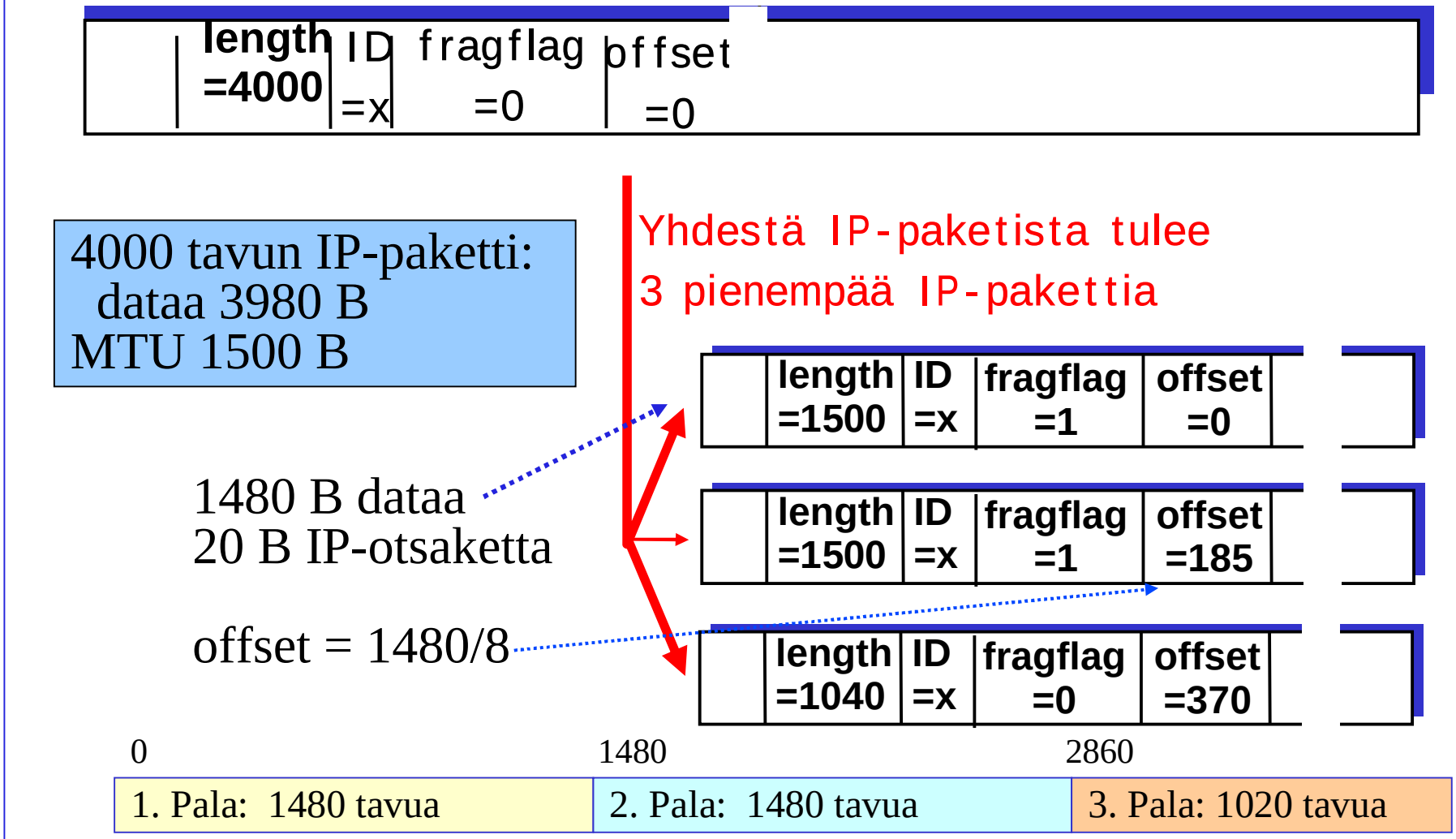
- voivat kukin kulkea eri reittiä

IP-otsakkeessa kentät yhteenkuuluvien fragmenttien tunnistamiseksi ja kokoamiseksi



KuRo08: Fig 4.14

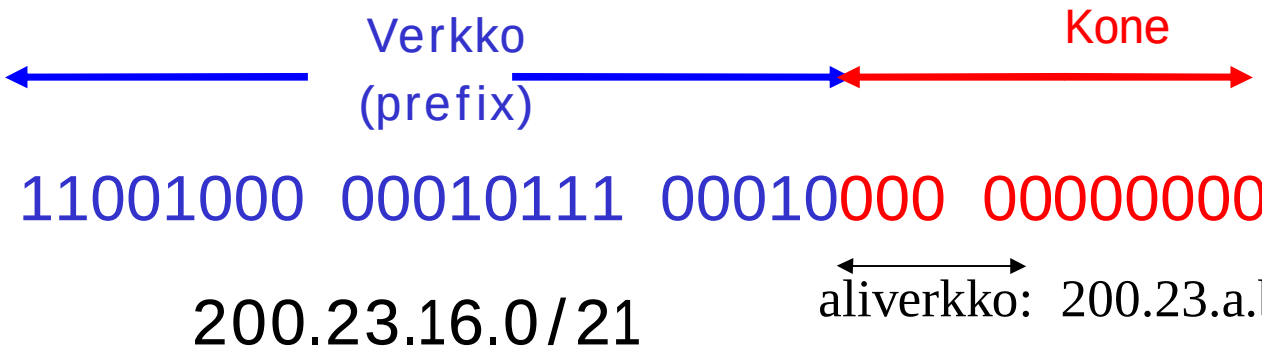
Esimerkki


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
2860
1. Pala: 1480 tavua
2. Pala: 1480 tavua
3. Pala: 1020 tavua

CIDR: Classless InterDomain Routing

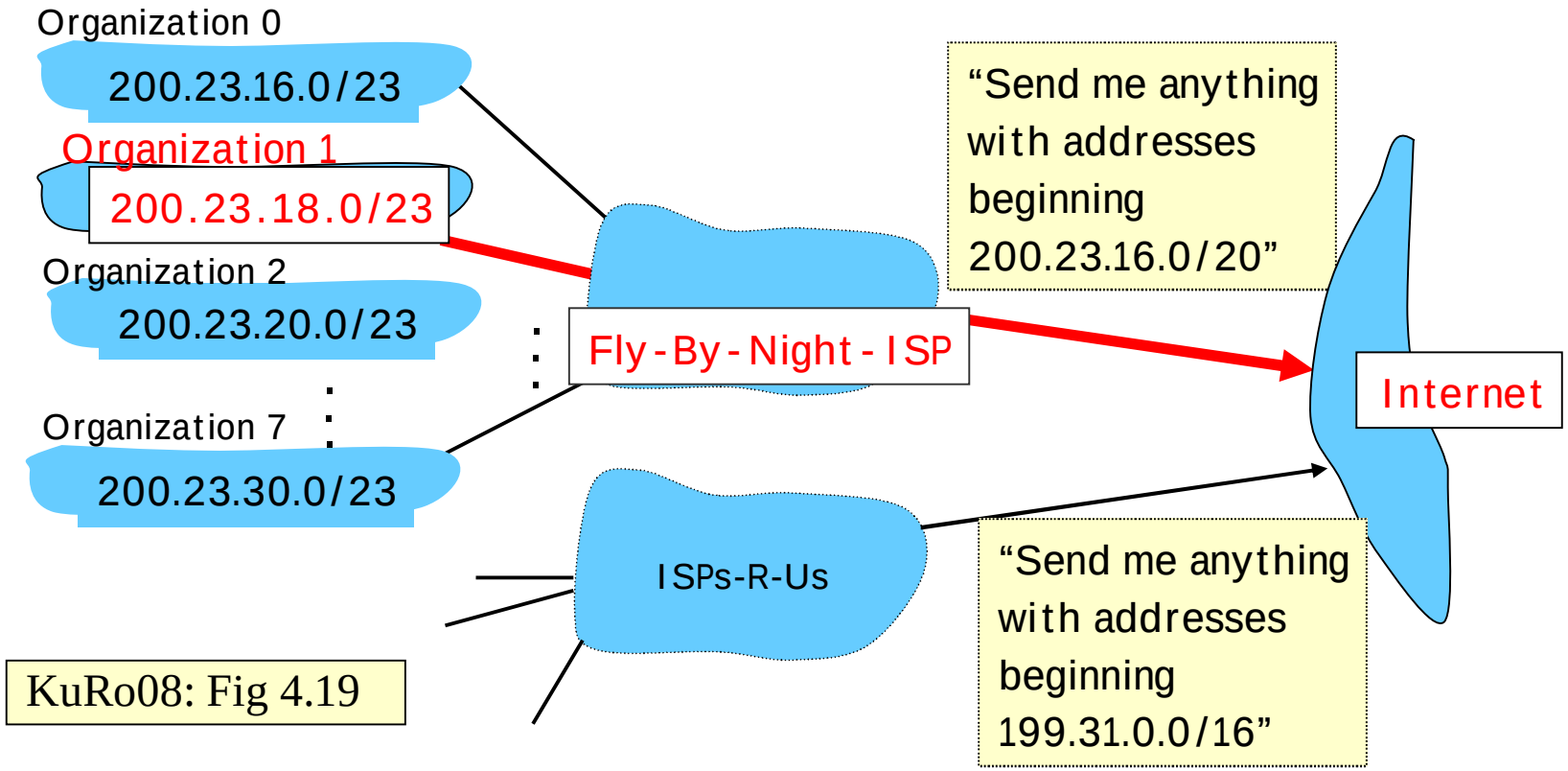
- Verkko-osa voi olla minkä tahansa kokoinen
  - Vanha luokallinen osoite: A-luokka 8 b, B-luokka 16 b,C-luokka 24 b
- Formaatti: a.b.c.d/x
  - x ilmoittaa verkko-osan bittienlukumäärän (prefix)

  Esim. Organisaatio, jolla 2000 konetta varaa 2024 = 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.



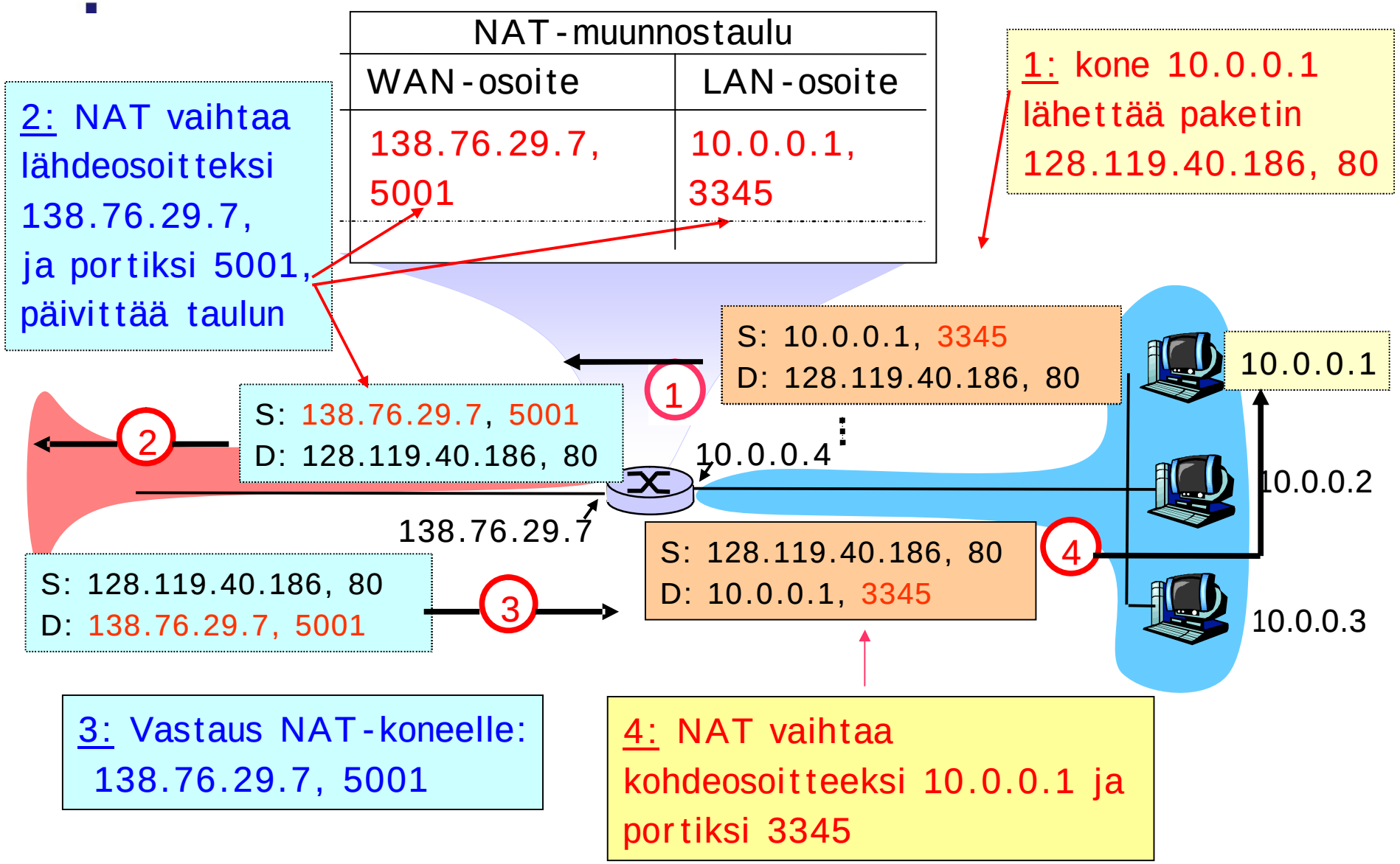
Hierarkkinen osoite

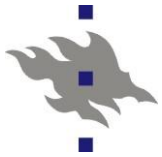
- CIDR luo reititystä helpottavan hierarkian
  - Aggregointi (yhdistäminen): yhteinen alkuosa => samaan suuntaan



KuRo08: Fig 4.19

NAT: Esimerkki





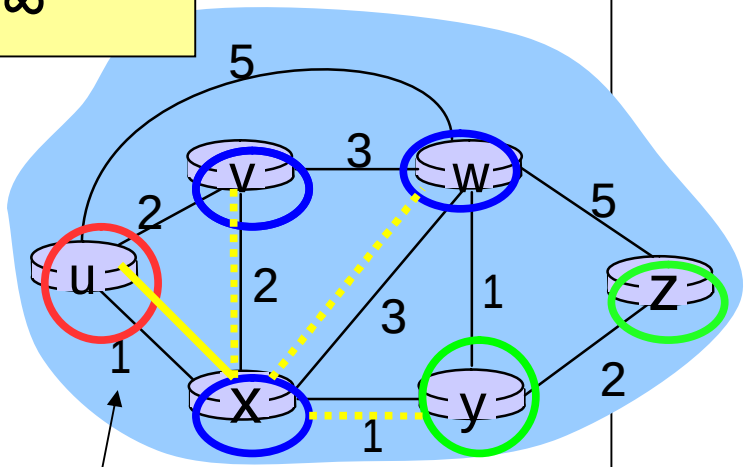
Reititysalgoritmi

- **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)

Dijkstran algoritmi

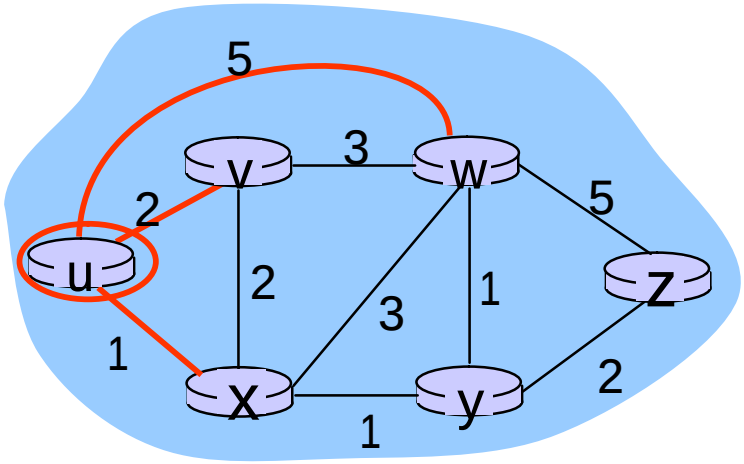
D(v)=2, D(w) = 5, **D(x)=1**
D(y) = ∞ , D(z)= ∞

```
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'
```



Etäisyysvektorireititys:

Esimerkki 1



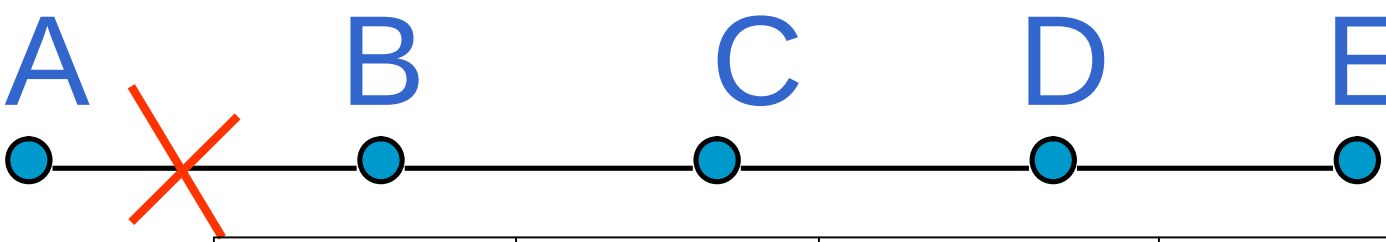
Kohde	kust.	linkki
Z	**4**	**X:ään**

Jos on jo saatu selville
(= naapurit kertoneet), 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)

Huono uutinen etenee hitaasti!



Linkki AB katkeaa => etäisyys äärettömäksi

Joka vaihdossa 'paras arvio' huononee vain yhdellä = reitityssilmukka

Count-to-infinity -ongelma

$D_B(A)$	$D_c(A)$	$D_D(A)$	$D_E(A)$
ääretön	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			

Etäisyys A:han

Linkkikerros

- Laitetoimintoa
- Siirtää paketin fyysistä linkkiä pitkin koneelta (solmulta (node)) toiselle
  - langallinen / langaton
  - bitit sisään, bitit ulos
- Kapseloi paketin siirtoon sopivaan muotoon
  - Siirtokehys (frame)
- Lähiverkossa linkkejä voi yhdistää keskittimillä tai kytkimillä
  - Käytetään fyysisiä osoitteita
  - 'reititystä' ilman IP-osoitteita

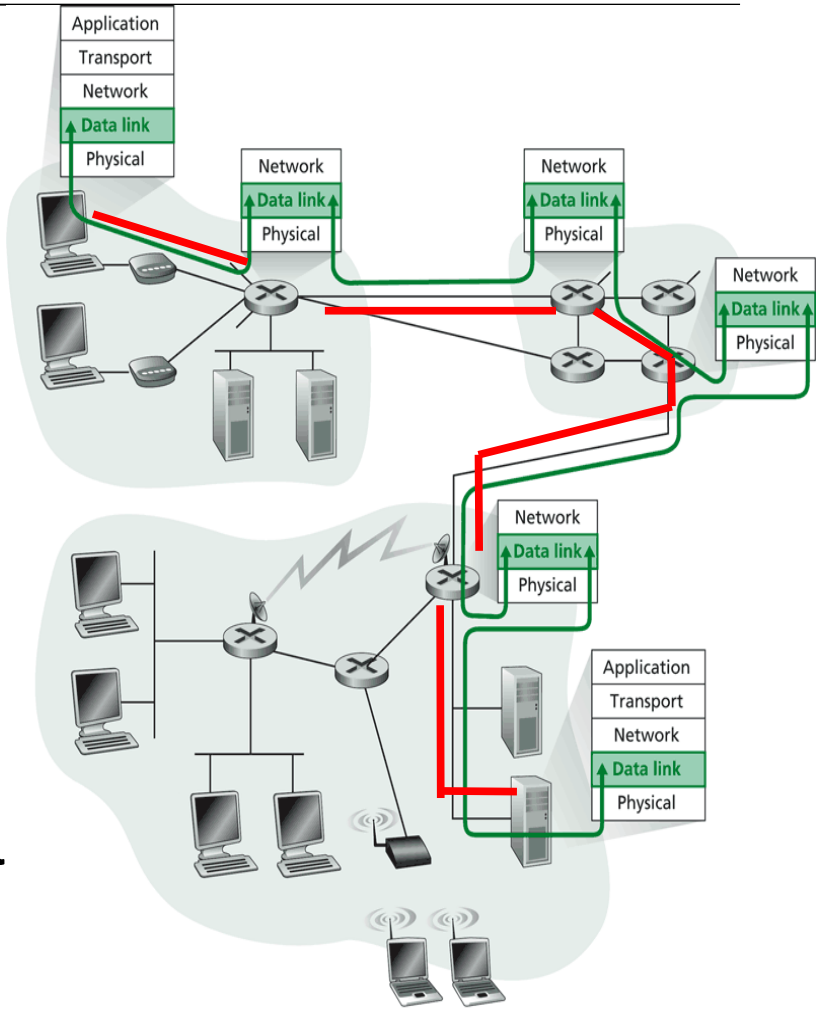


Figure 5.1 ♦ The link layer

CRC-esimerkki

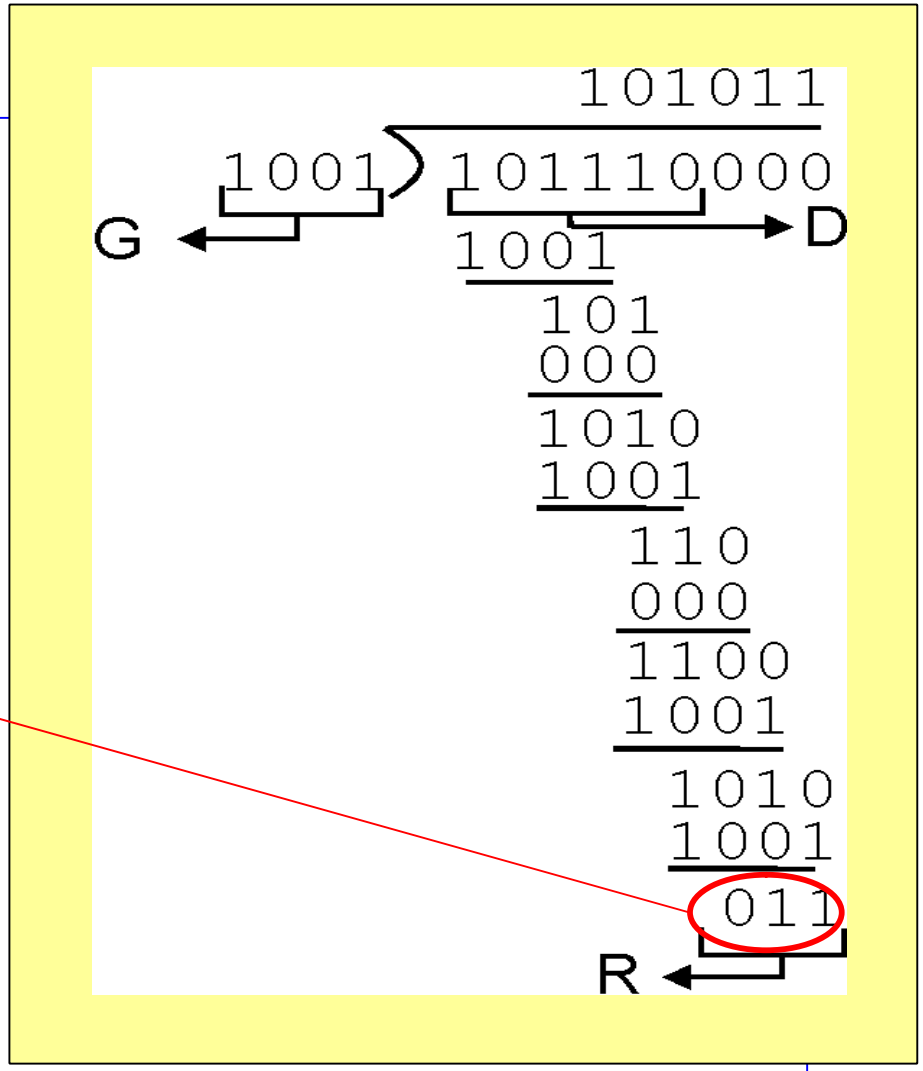
Data: 101110

G: 1001, polynomina

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

<D,R>: 101110???

Lähetä: 101110011



KuRo08:Fig 5.8

Modulo 2-aritmetiikka

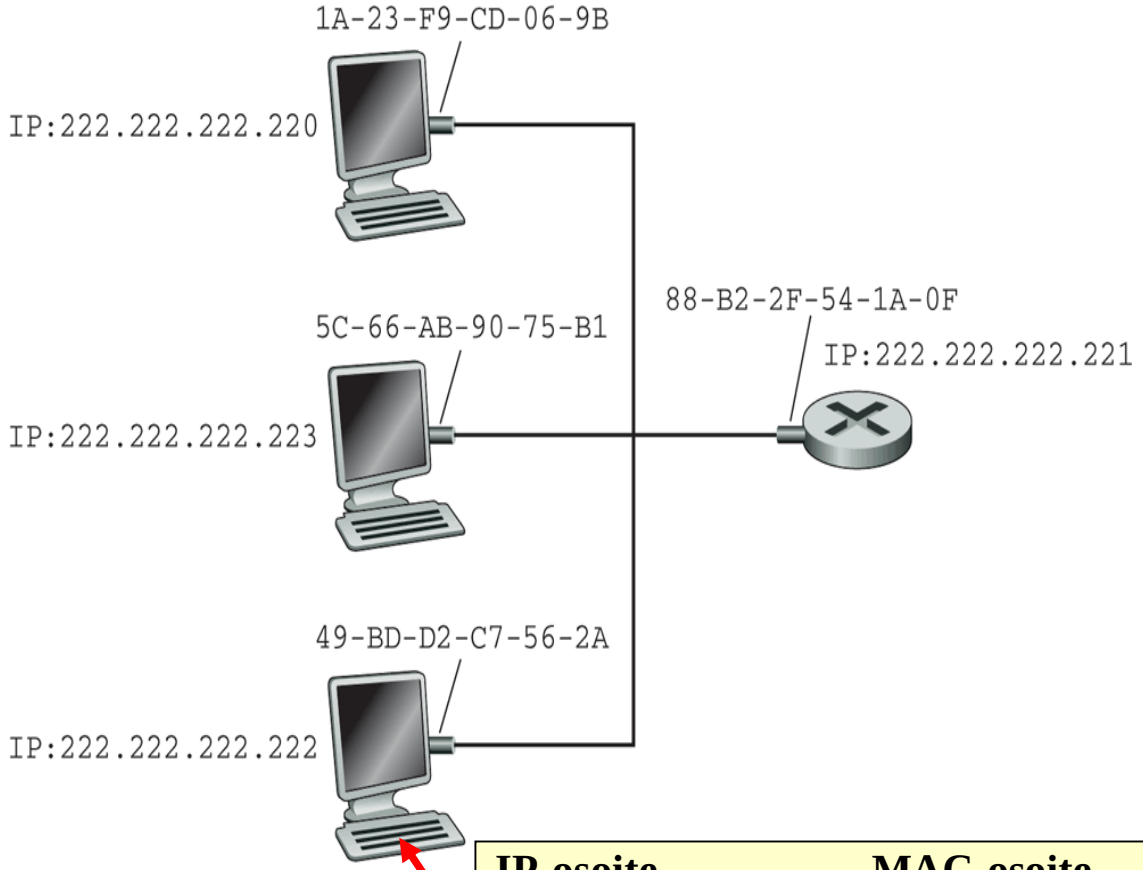
vähennyslasku yhteenlaskuja

ei lainaamista, ei muistinumeroita

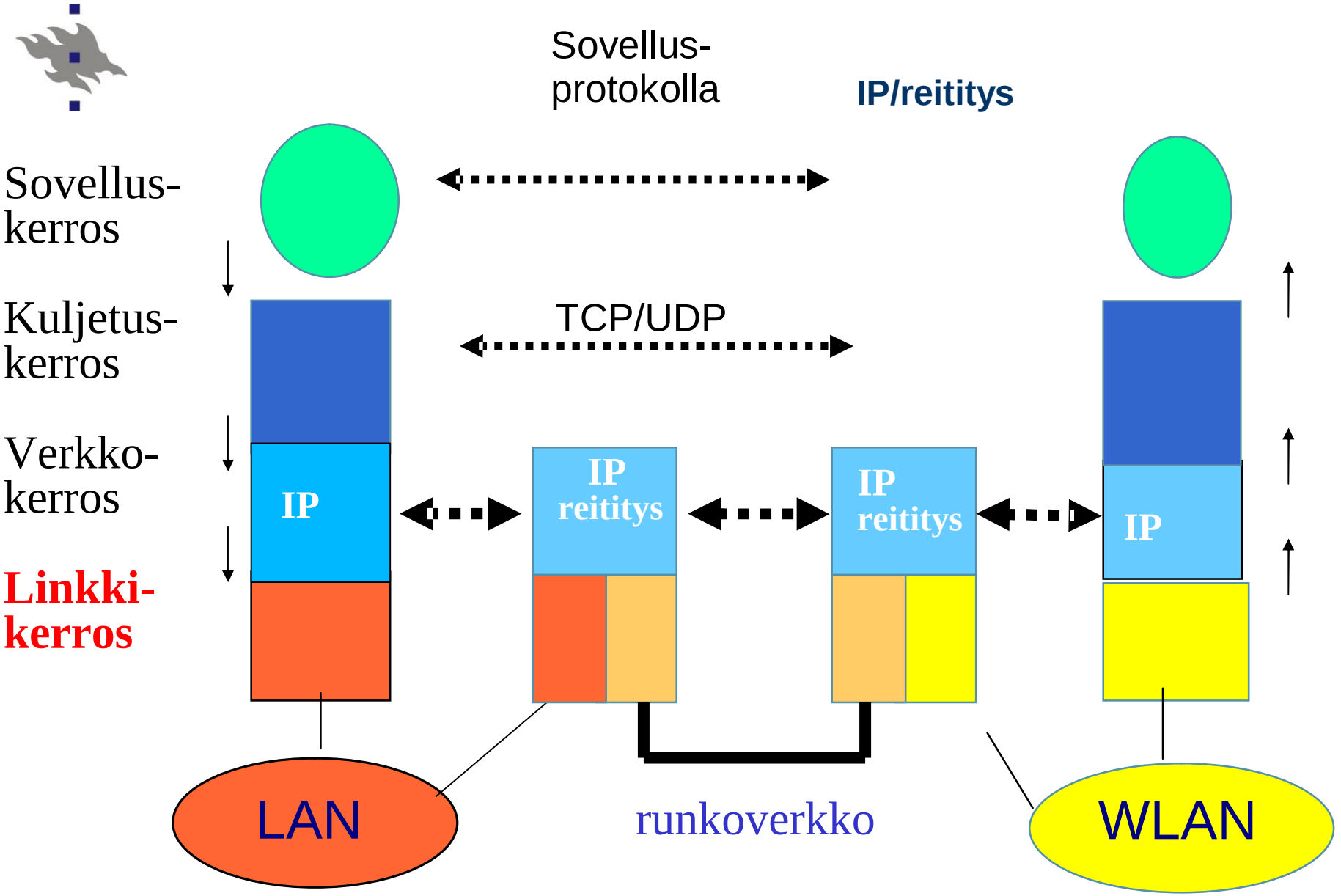
= bittitason XOR

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

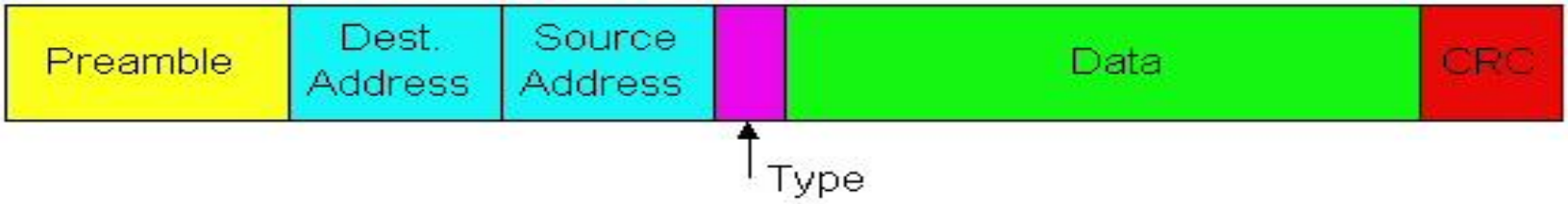
MAC-osoitteet ja ARP-taulu, ARP-protokolla



IP-osoite	MAC-osoite	TTL
222.222.222.220	1A-23-F9-CD-06-9B	13:24:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00


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

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)

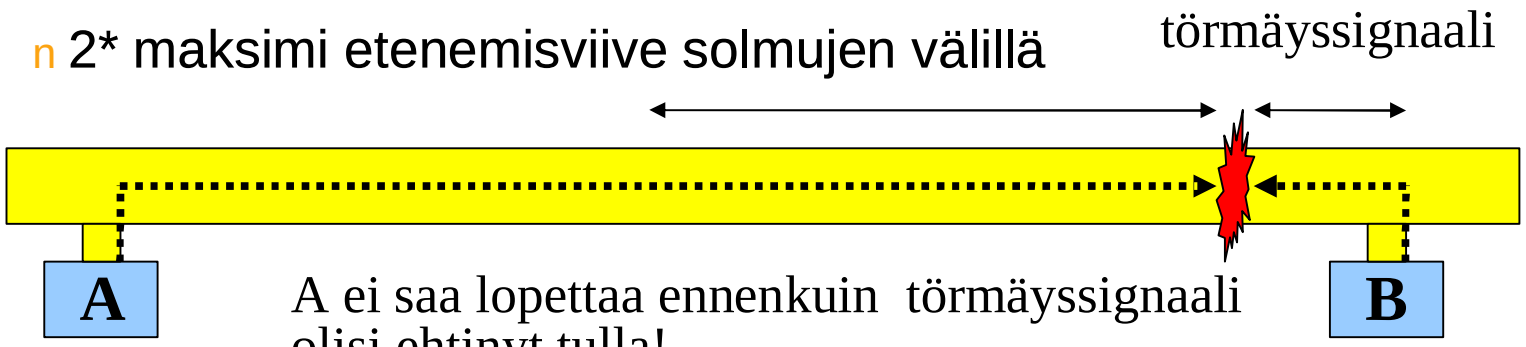
- Ethernet MTU = 1500 B

CRC (4 B)

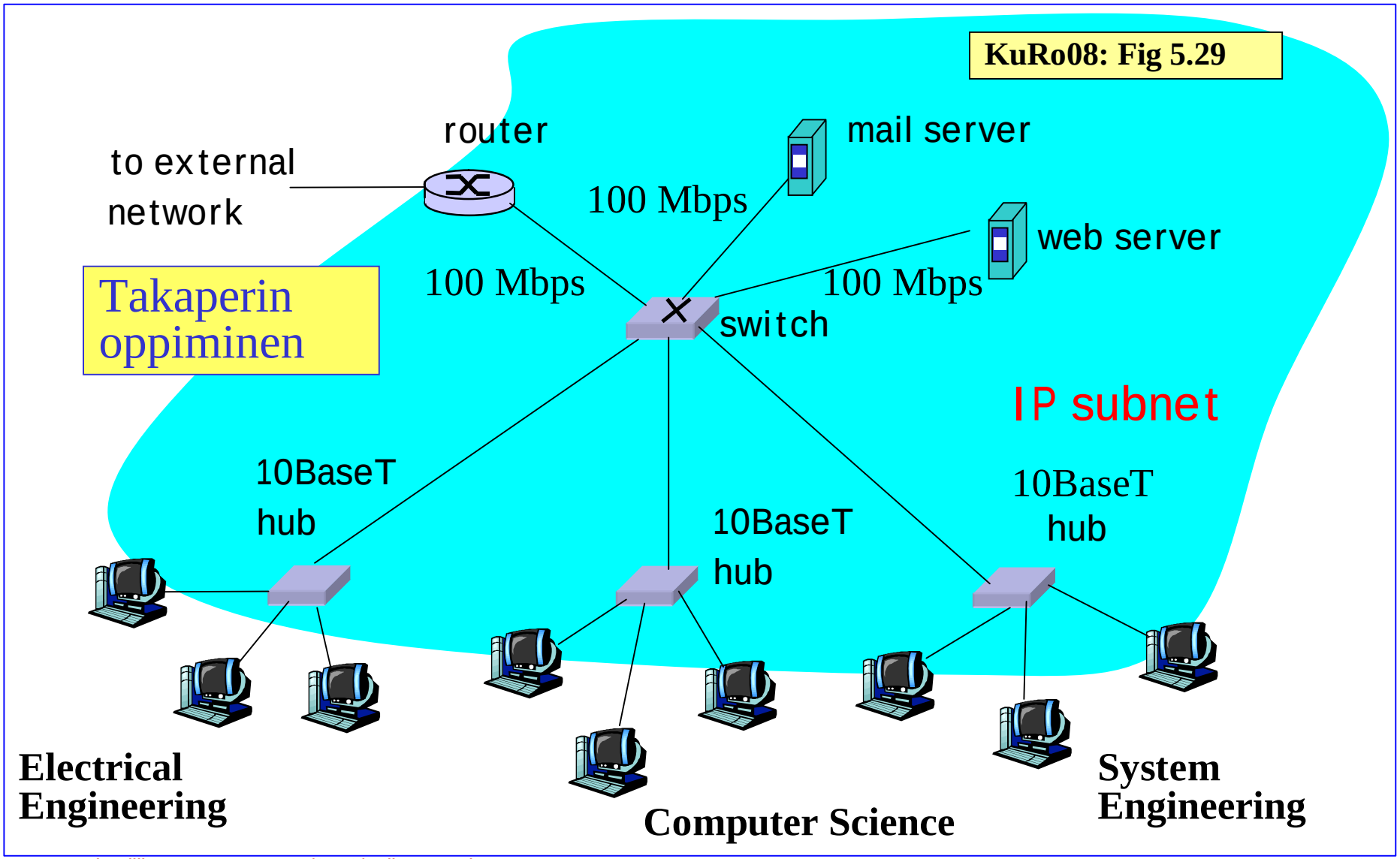
- tarkistusbitit, tahdistuskuvio mukana laskennassa

CSMA/CD (with Collision Detection)

- Asema kuuntelee myös lähettämisen jälkeen
  - Langallinen LAN: 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



LAN, verkkosegmentit



802.11: CSMA/CA

KuRo08: Fig 6.10

Lähetys

1. **Jos kanava vapaa**
   - Kuuntele DIFS aikayksikköä
   - Lähetä kehys kokonaan
2. **Jos kanava varattu**
   - Käynnistä peruutuslaskuri (backoff) random(max), jota vähennetään vain kun kanava on vapaa,
   - Lähetä, kun laskuri nollassa
   - Jos ei tule kuittausta, niin yritä uudestaan max = 2*max

Vastaanotto

Jos kehys OK

- Odota SIFS aikayksikköä
- Lähetä ACK (linkkikerroksen ACK)

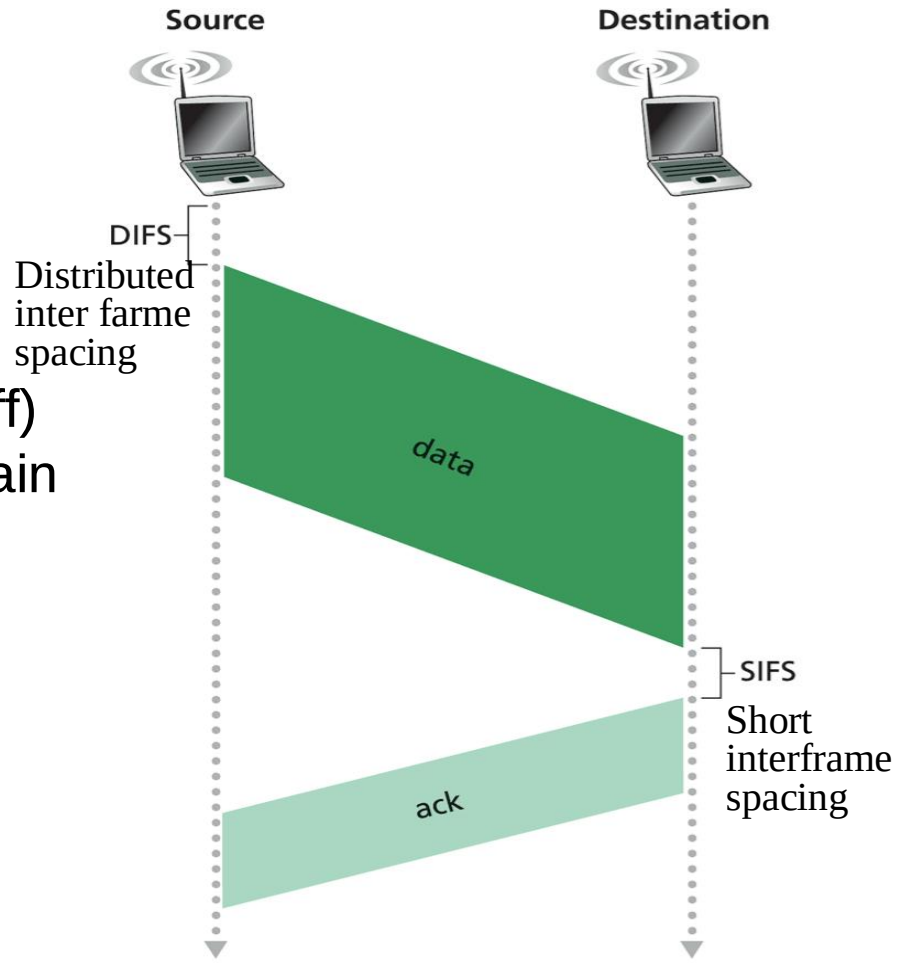


Figure 6.8 ♦ 802.11 uses link-layer acknowledgmer

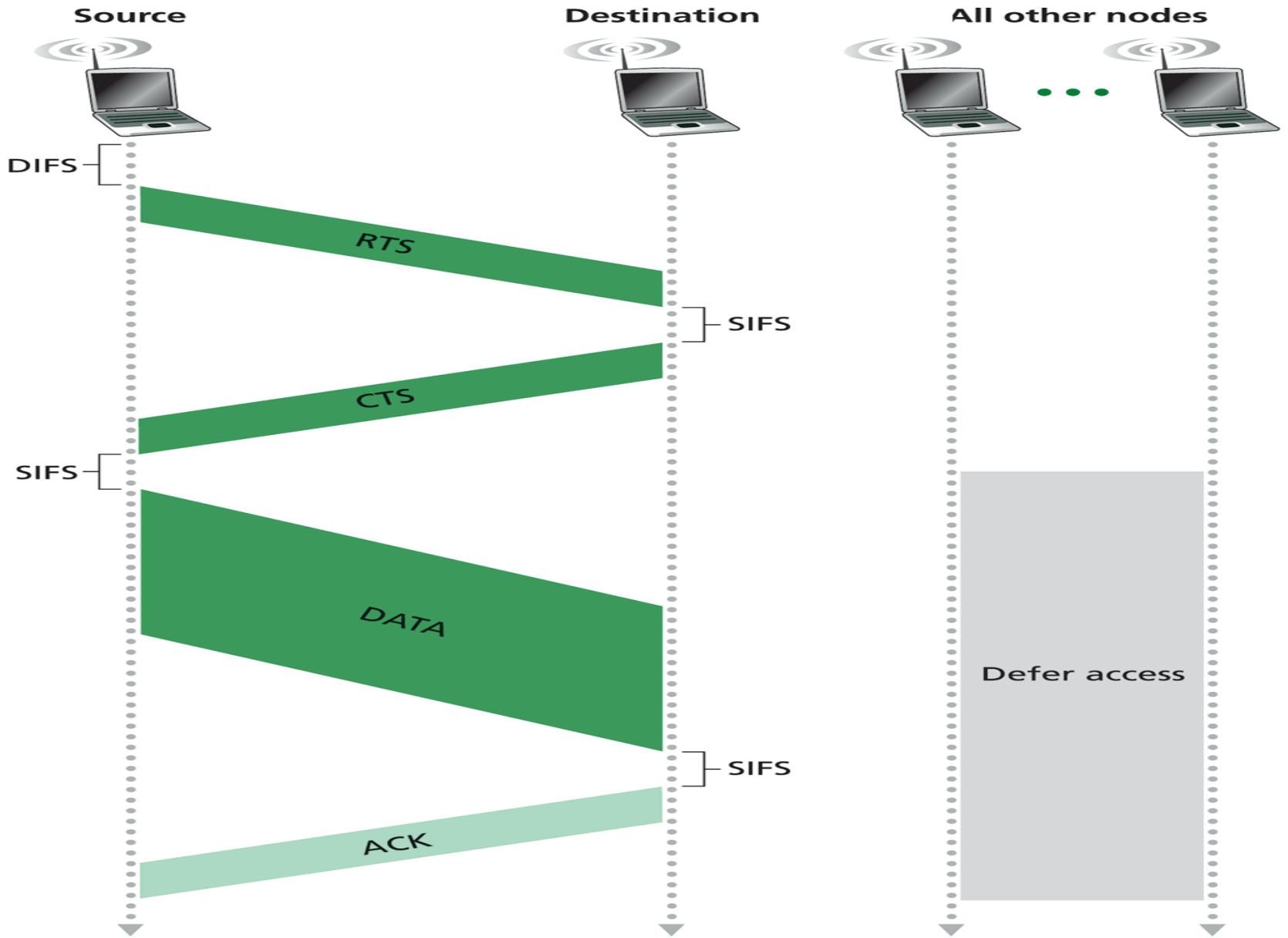


Figure 6.10 ◆ Collision avoidance using the RTS and CTS frames

Hajautettu DoS-hyökkäys (DDoS)

- Hyökkääjä ottaa ensin haltuun ison joukon koneita niiden omistajien huomaamatta
  - Koputtelee ja löytää turva-aukot
  - Asentaa hyökkäysohjelman, joka vain odottelee käskyä /kellolyömää
- Kaapatut koneet aloittavat samaan aikaan hyökkäyksen uhrin kimppuun
  - Hajautetusti
  - IP-osoitteet peukaloituina

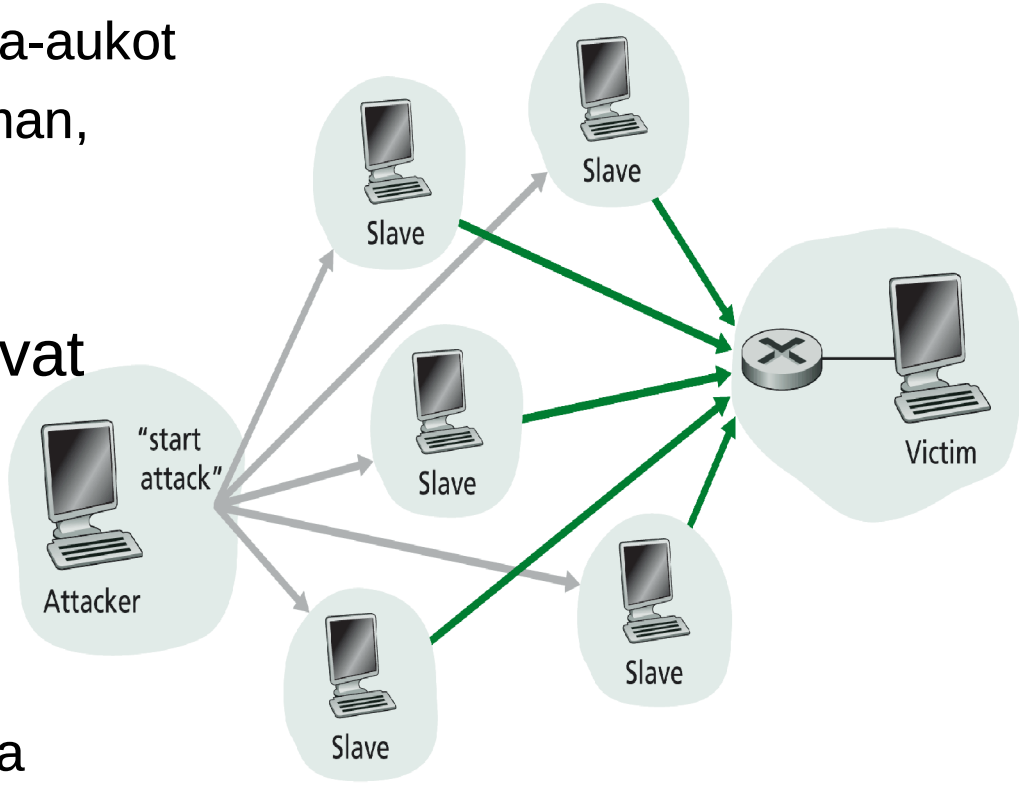
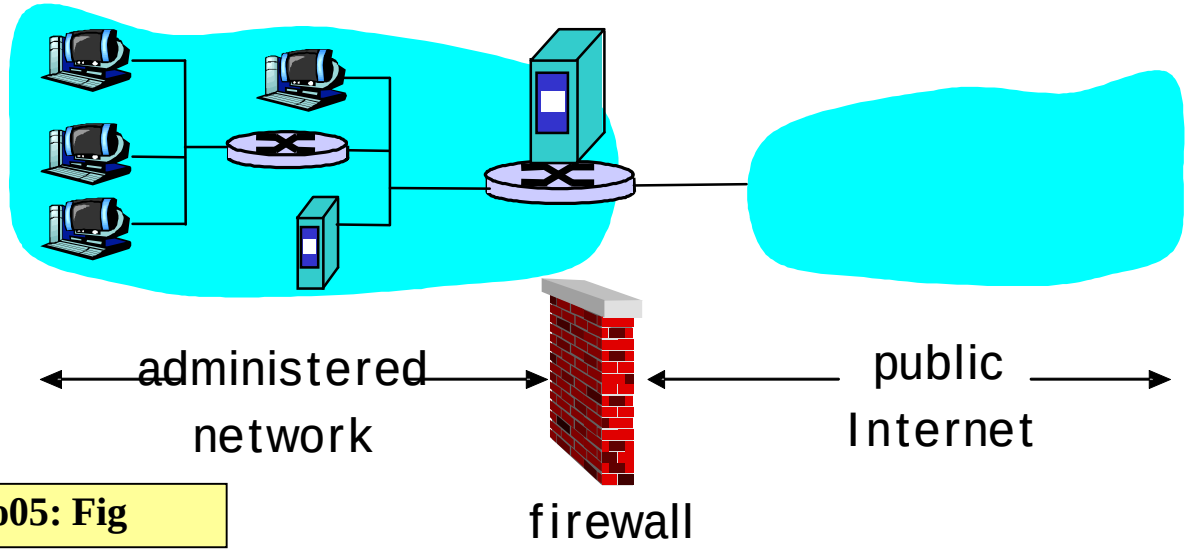


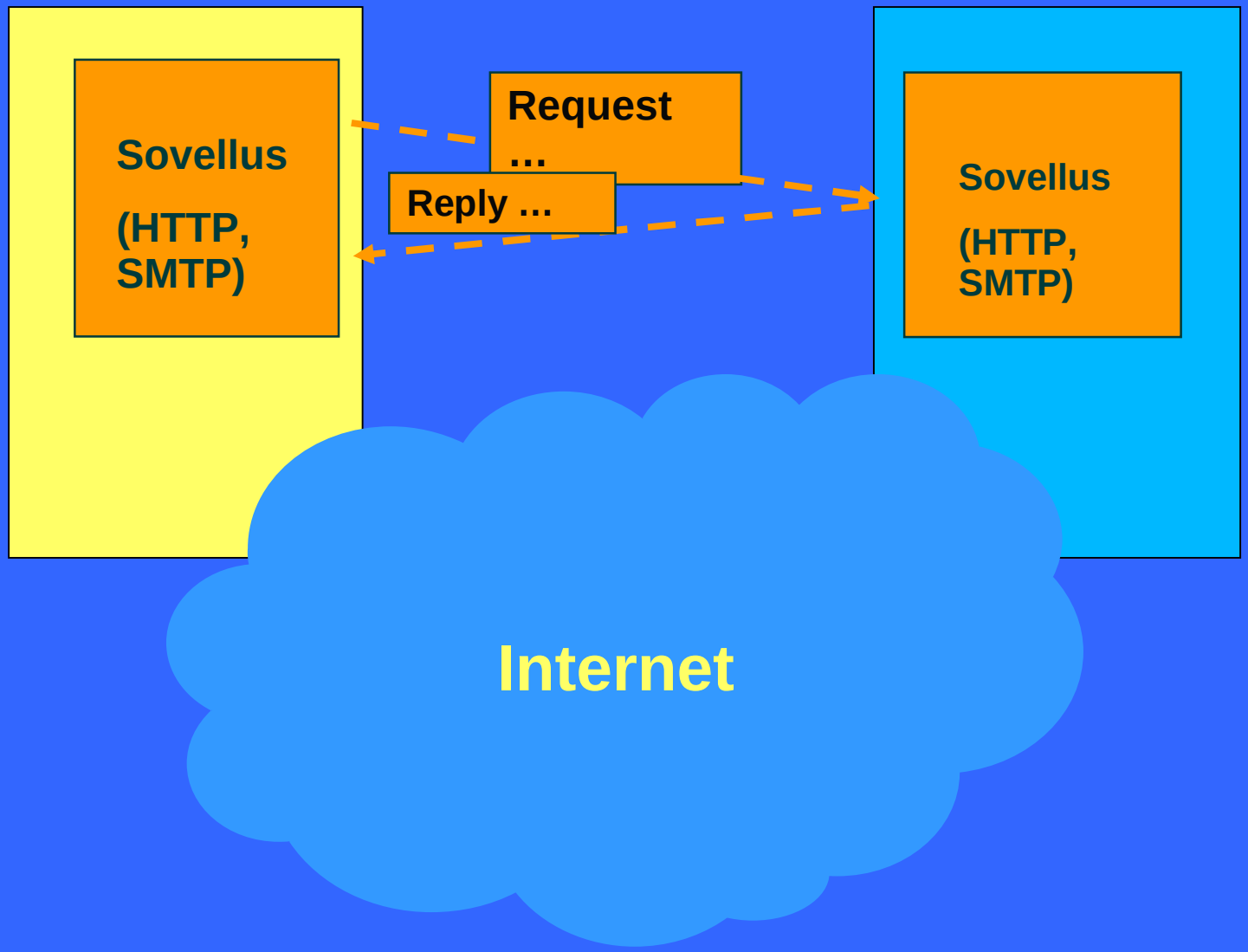
Figure 8.26 ♦ A DDoS attack

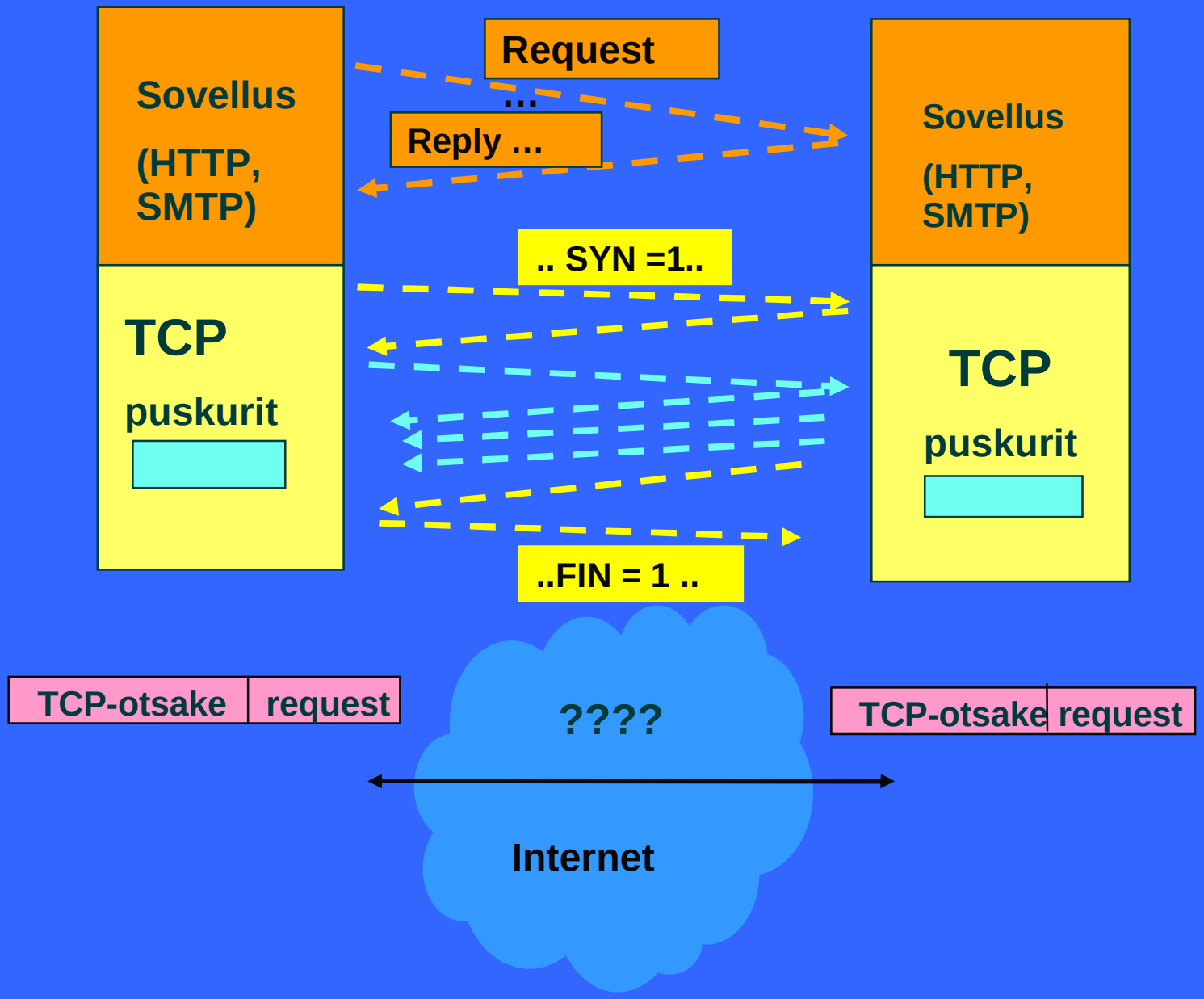
Palomuuri (firewall)

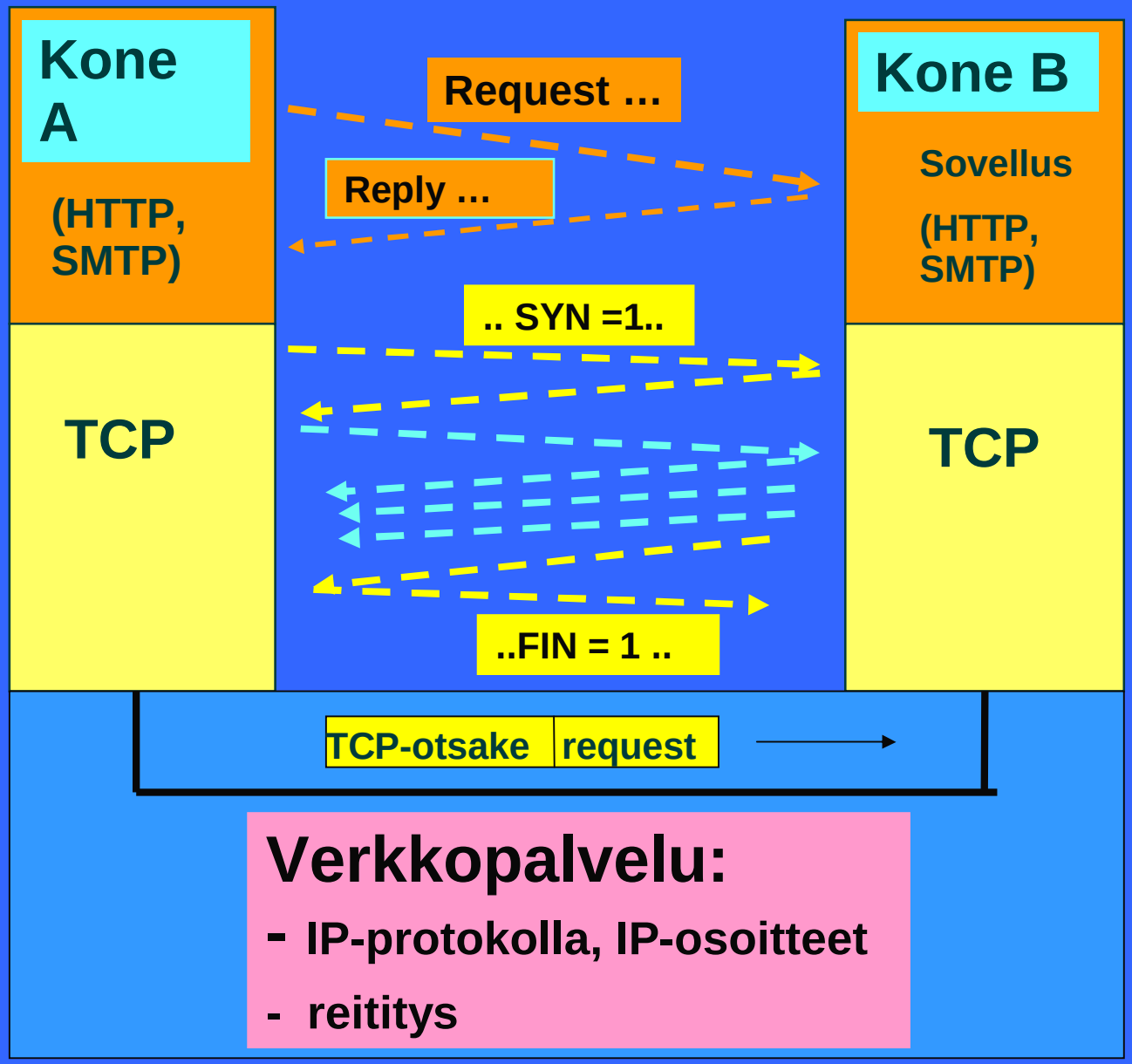
- Ohjelmisto + laitteisto
- Suodattaa (filteroi) liikennettä organisaation oman verkon (intranet) ja julkisen Interbetin välillä
  - Osa IP-paketeista pääsee palomuurin läpi, osa ei

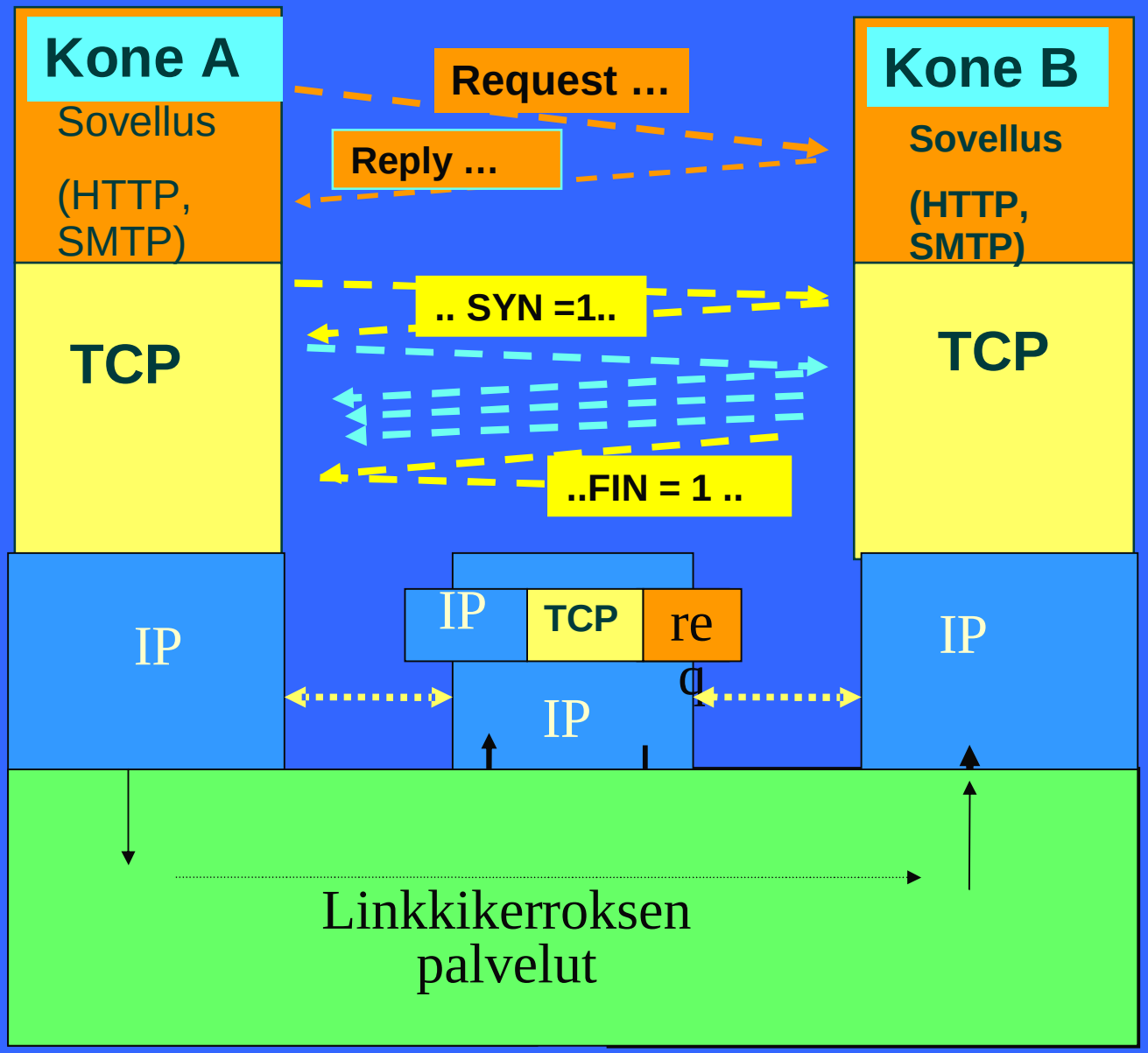


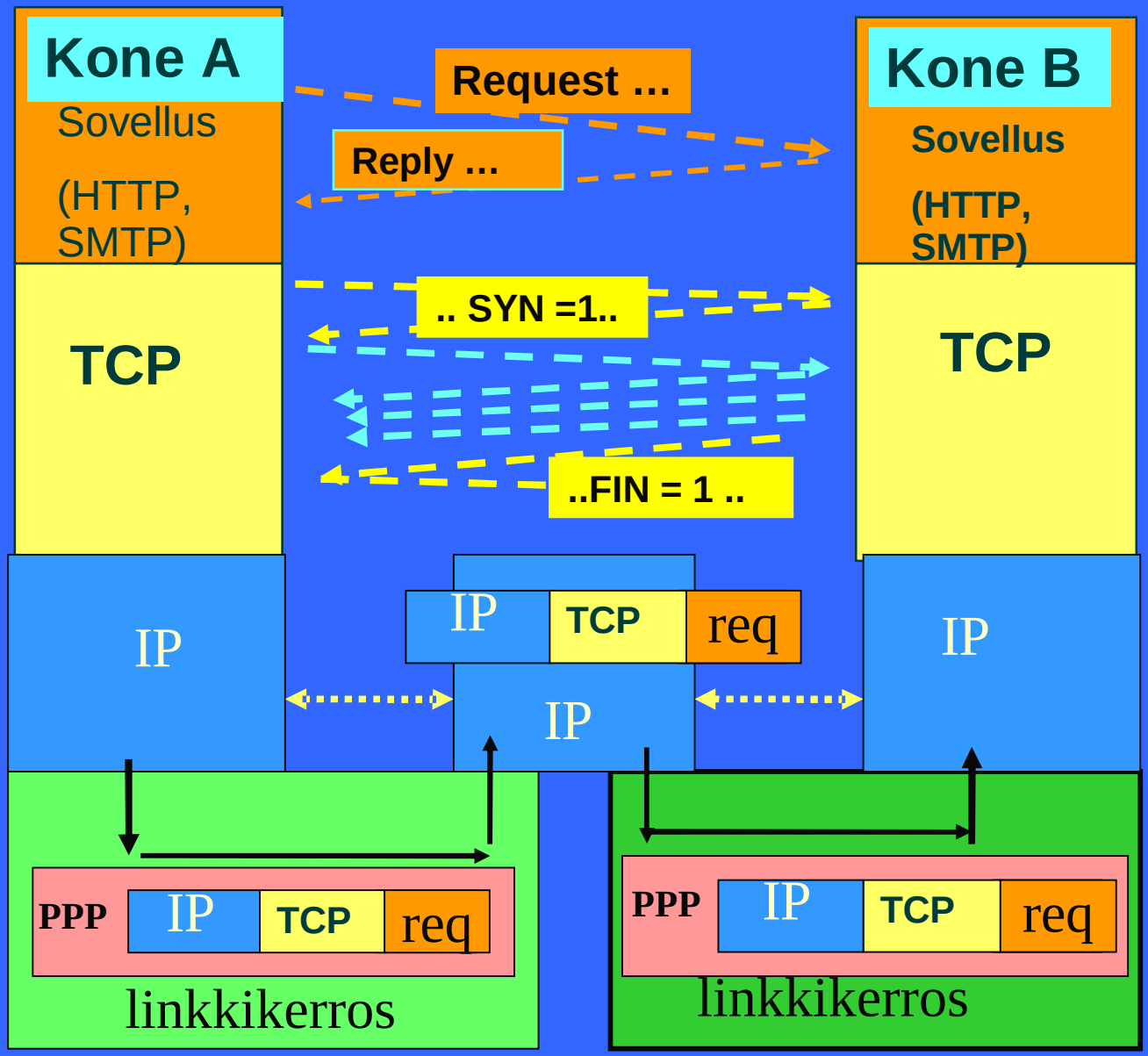
KuRo05: Fig 8.23


Sovellus
(HTTP, SMTP)
Request ...
Reply ...
Sovellus
(HTTP, SMTP)
Internet

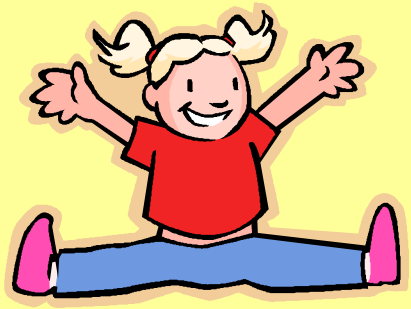

Sovellus
(HTTP, SMTP)
TCP
puskurit
Request
...
Reply ...
.. SYN =1..
..FIN = 1 ..
Sovellus
(HTTP, SMTP)
TCP
puskurit
TCP-otsake
request
????
TCP-otsake
request
Internet


Kone A
(HTTP, SMTP)
TCP
Request …
Reply …
.. SYN =1..
..FIN = 1 ..
Kone B
Sovellus
(HTTP, SMTP)
TCP
TCP-otsake
request
Verkkopalvelu:
- IP-protokolla, IP-osoitteet
- reititys


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


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

Tietoliikenteen perusteet



Siinäpä se!