Tietoliikenteen perusteet

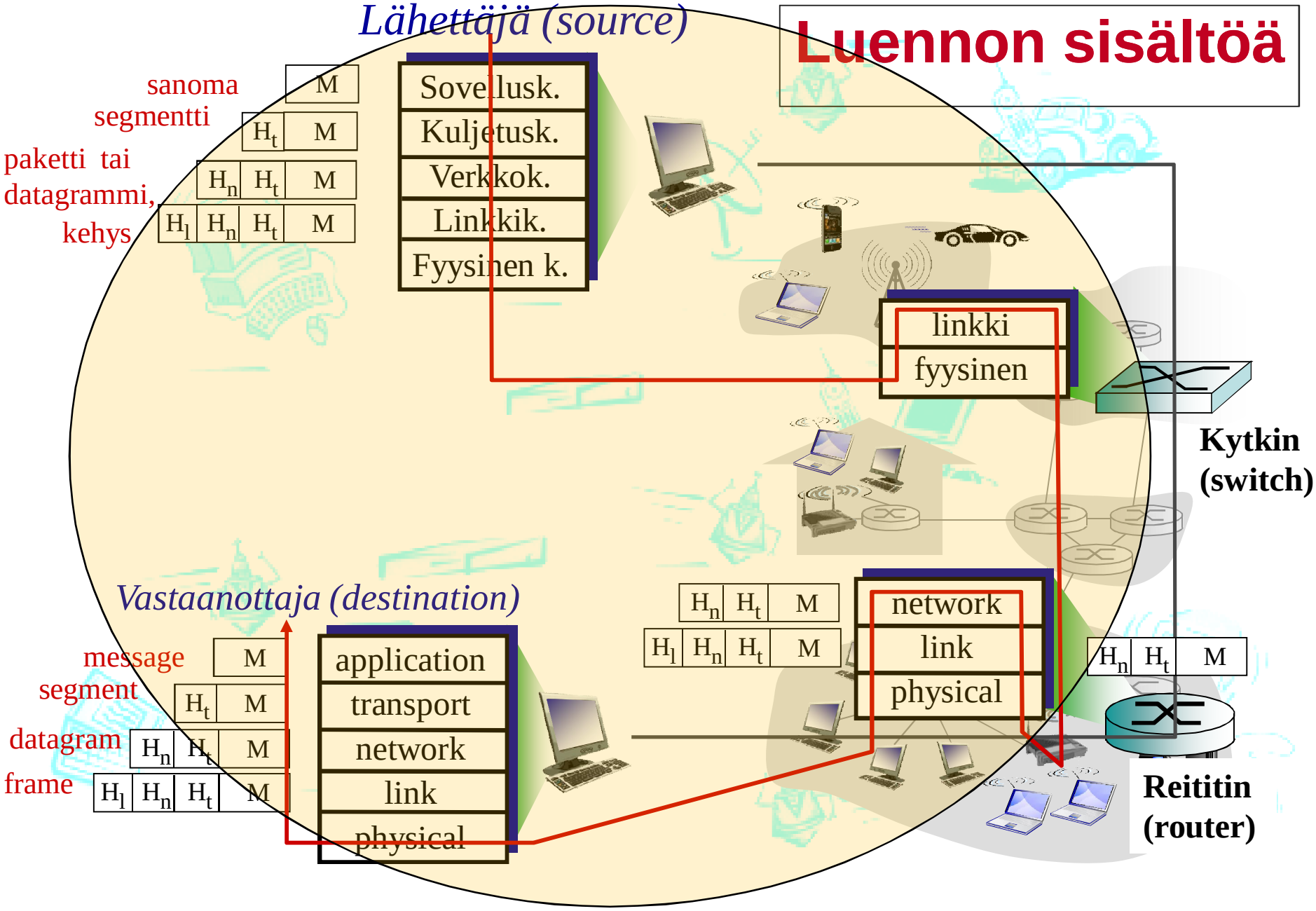
Luento 14: Kertausta

Syksy 2014, Tiina Niklander

Kurose&Ross: Ch 8

Pääasiallisesti kuvien

Luennon sisältöä



Sisältöä

Viime vuoden kurssikoe

Kertausta keskeisestä sisällöstä
- **aiempien luentojen kalvoja**
- **kysymyksiä**

Oppimistavoitteet:
- Kurssin yleiset oppimistavoitteet ja kalvoihin kerätyt (laajennetut) versiot

Kurssikoe

- Kurssin koe on keskiviikkona 17.12.2014 09.00 A111
- Ei tarvita laskinta, ei ’lunttipaperia’
- Vastaukset jokaiseen kysymykseen omalle arkille
- Koetilanteessa jaetaan (lomaketilauksen tehneille) QR-kooditarrat liimattavaksi koepapereihin
- Tarroitetut vastaukset saa arvostelun jälkeen tammikuussa skannattuna omaan postilaatikkoon
- Kurssikokeesta 50 pistettä ja harjoituksista 10 pistettä

Vuoden 2013 kurssikoe

- 3 kysymystä

- Internet protokollapino (15 p)
- Ruuhkanhallinta ja vuonvalvonta (18 p)
- Lähiverkoista (17p)

1. Internetin protokollapino (15 p)

- Piirrä Internetin protokollapino ja nimeä sen eri kerrokset. Kerro lyhyesti kunkin kerroksen tehtävistä. (5 p)

- Millaisia osoitteita eri kerroksilla käytetään? (5p)

- Sijoita seuraavat protokollat oikeisiin kerroksiin: DNS, UDP, ARP, DHCP ja IMAP. Kerro myös muutamalla virkkeellä, mihin näitä protokollia käytetään. (5p)

2. Ruuhkanhallinta ja vuonvalvonta (18 p)

Käytetään TCP-protokollaa. Segmentin maksimikoko on 2 kB dataa + otsakkeet. Kynnysarvo (threshold) on 18 kB. Vastaanottajan ikkunan koko on 20 kB. Yhteyden kiertoviive (round trip time) on 100 ms.

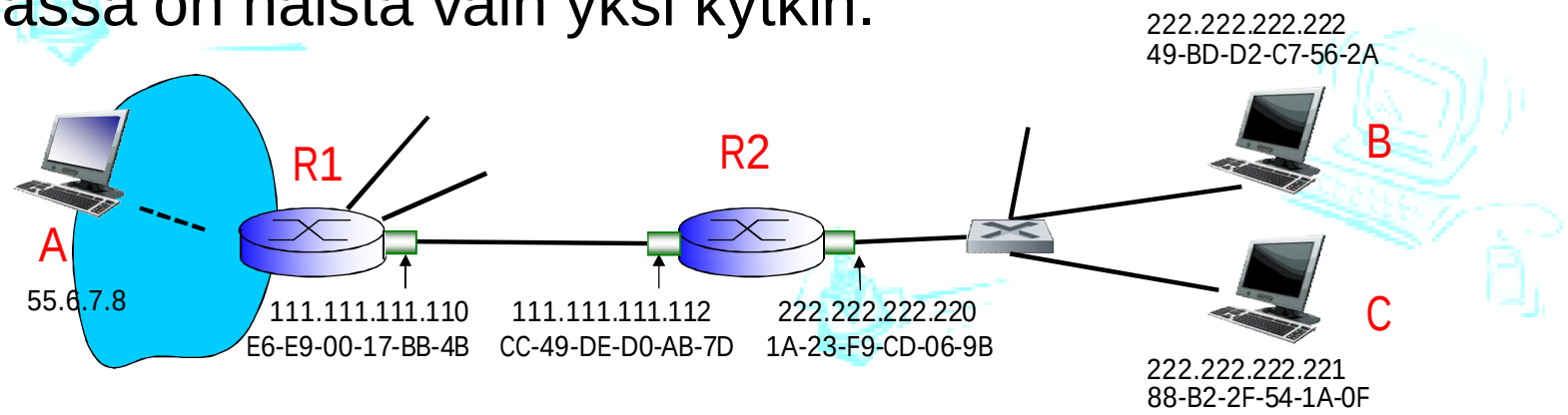
- Esitä kaaviokuvana, kuinka lähettäjä heti yhteyden alussa lähettää 100 kB dataa, kun kuittaukset lähetettyihin segmentteihin saapuvat ajoissa eikä vastaanottaja muuta vastaanottoikkunan arvoa. (6p)

- Entä jos 10. lähetetty segmentti katoaa, mutta muut segmentit ja kuittaukset kulkevat virheettöminä. Miten lähettäjä havaitsee katoamisen? Esitä kaaviokuvana, miten lähettäjä jatkaa lähetystä? (6p)

- Mitä tarkoitetaan vuonvalvonnalla (flow control) ja ruuhkanhallinnalla (congestion control)? Miksi niitä tarvitaan? Mitkä tehtävässä esitetyt numeroarvot liittyvät niihin ja miten? (6p)

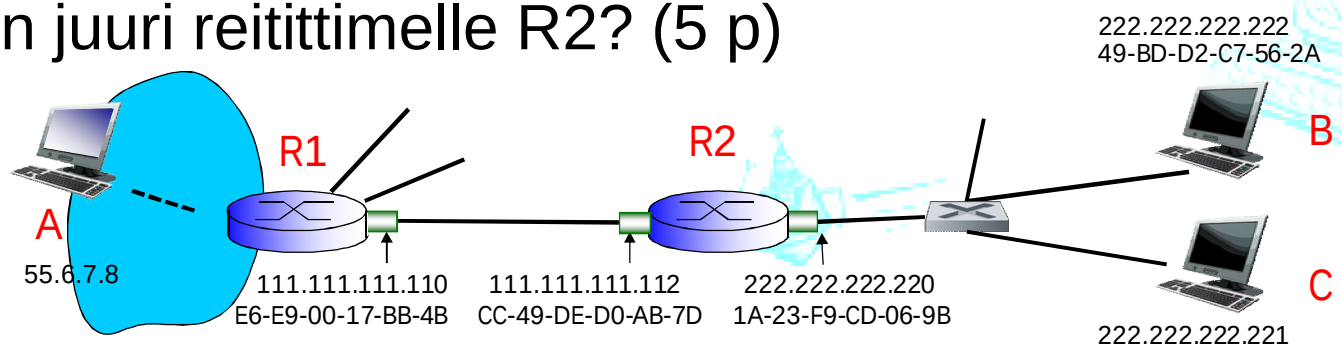
Lähiverkoista (17 p) - kalvo 1/2

- Kuvan reititin (router) R2 vastaanottaa toiselta reitittimeltä R1 oman Ethernet-lähiverkkonsa koneelle B osoitetun paketin (datagrammin), joka sisältää HTTP-vastauksen koneen B lähettämään HTTP-kyselyyn. Vastauksen lähettäjä on kone A jossain Internetissä. Reitittimen R2 oma lähiverkko koostuu kytkimillä (switch) ja keskittimillä (hub) yhdistetyistä lähiverkoista. Kuvassa on näistä vain yksi kytkin.



Lähiverkoista (17 p) - kalvo 1/2

- Minkä eri protokollien otsakkeita ja dataa reitittimen R2 vastaanottama paketti sisältää? Piirrä kuva. (4 p)
- Minkä muotoisena reititin R2 lähettää paketin eteenpäin? Piirrä kuva, josta selviää erityisesti eri kerroksilla käytetyt osoitteet. (3 p)
- Miten aliverkkoja yhdistävä kytkin (switch) osaa ohjata saamansa kehyksen oikeaan aliverkkoon? (5 p)
- Miten reititin R1 osaa lähettää koneelle B osoitetun paketin juuri reitittimelle R2? (5 p)



Muita tyypillisiä kysymyksiä

- Kalvojen ja kirjan lopussa olevat kertauskysymykset!!!!
- Osoitteet, kerrosten yhteistoiminta, verkon rakenne
- Yksittäiset protokollat: HTTP, DNS, ARP, SMTP, UDP, TCP, DHCP, IMAP, IP, …
- Reitittin, kytkin - toimintaperiaatteet, paketin välitys
- Tänä vuonna lisäksi: tietoturvaa, avaimet, todennus, sähköpostin suojaus ja allekirjoitus, SSL, palomuurit, …

KERTAUSTA

Paketin sisältö

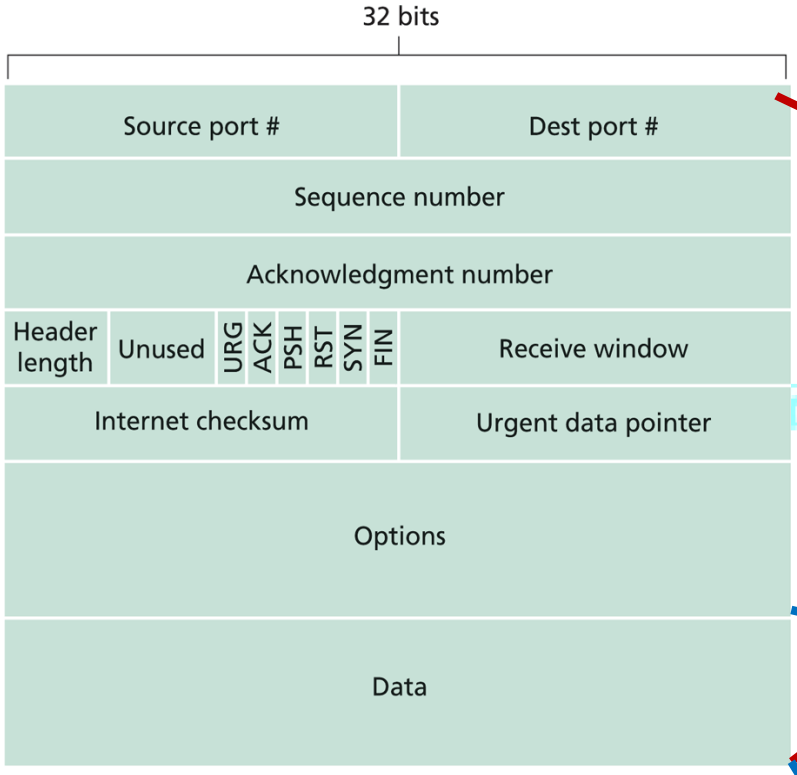
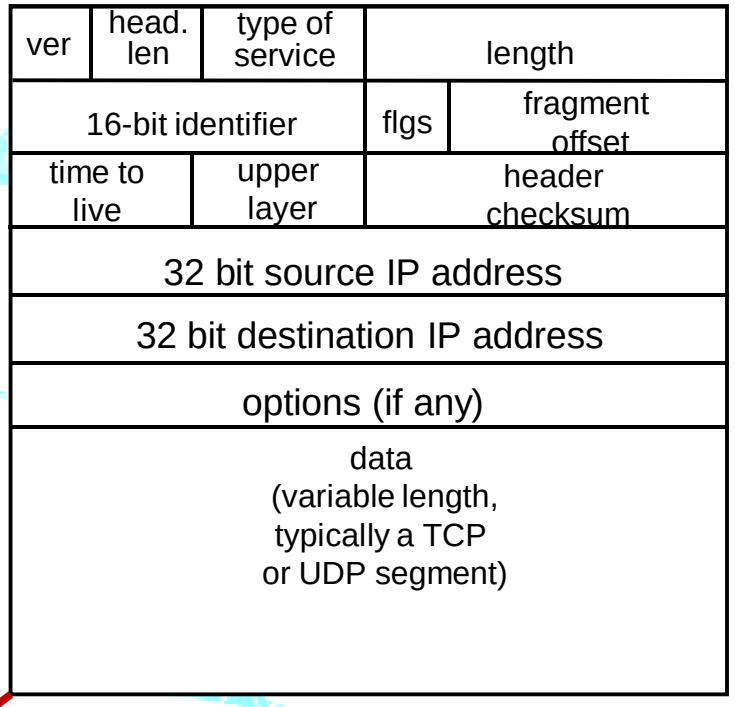
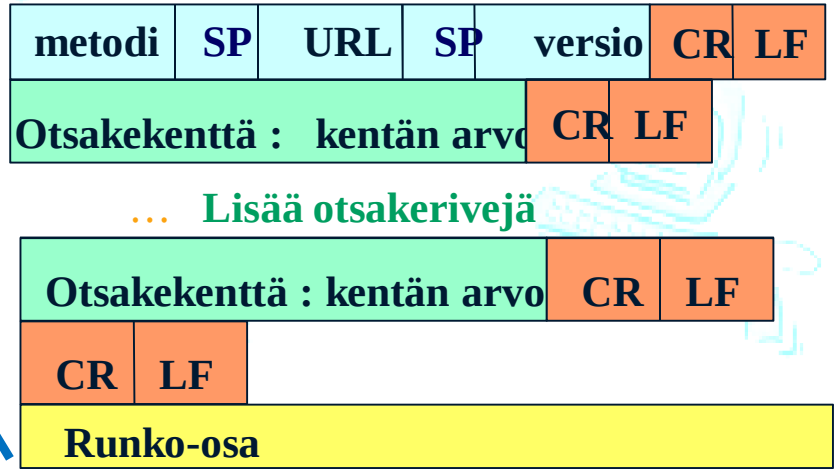


Figure 3.29 ♦ TCP segment structure





Internet: eri näkökulmia

Rakenne:

- verkkojen verkko (löyhä kytkentä)
  - Internet-palveluntarjoajien (Internet service provider, ISP) verkot yhdistetty
  - Julkinen Internet vs. rajattu intranet ja extranet
- Päästä-päähän suunnittelumalli
  - tila ja toiminnot reunoilla
- Protokollat - kommunikointisäännöt
- Standardeja
  - RFC – request for comments
  - IETF – Internet Engineering Task Force

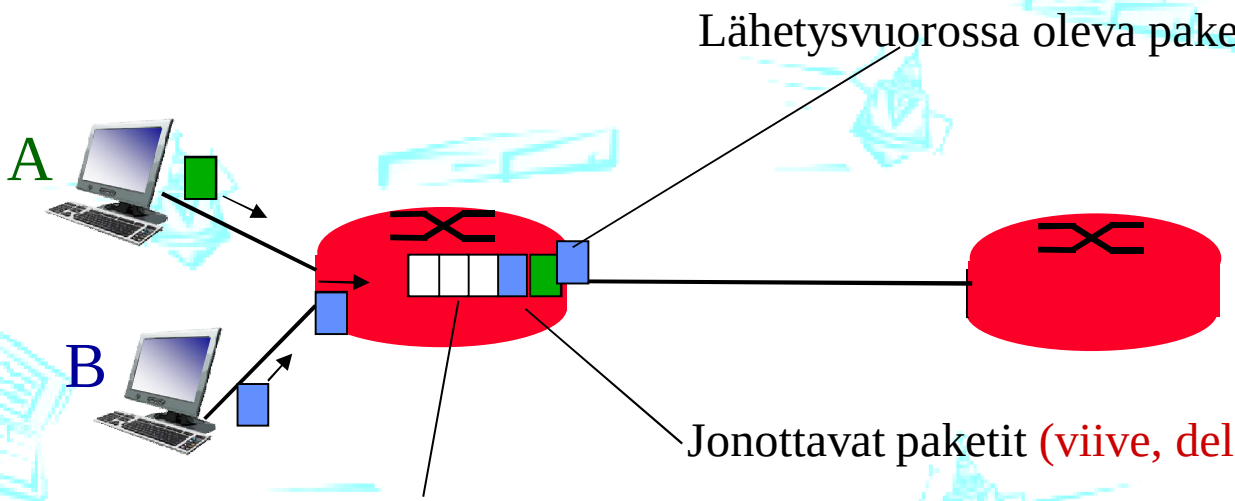
Palvelu:

- Infrastruktuuri, joka tarjoaa palveluja sovelluksille:
  - Web, VoIP, sähköposti, some, verkkopelit, verkkokauppa, …
- Tarjoaa ohjelmointirajapinnan sovelluksille (application programming interface, API)
  - koukkuja, joiden avulla sovellus voi lähettää ja vastaanottaa viestejä
- Tarjoaa viestintäpalvelua, hyvä analogia: posti
  - kirje postilaatikkoon

Mistä pakettien viivästyminen ja katoaminen johtuu?

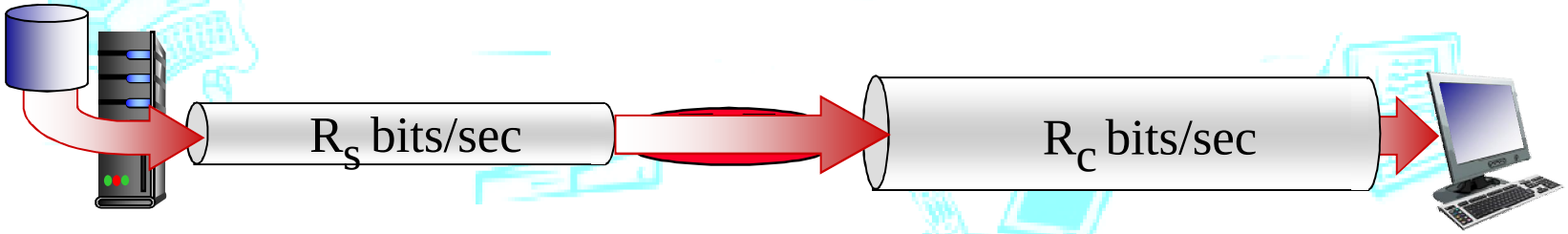
Paketit jonottavat (*queue)* reitittimien puskureissa (*buffers*)

- Pakettien saapumistiheys (*arrival rate*) ylittää (tilapäisesti) ulosmenevän linkin kapasiteetin
- Paketit jonottavat ja odottavat lähetysvuoroaan

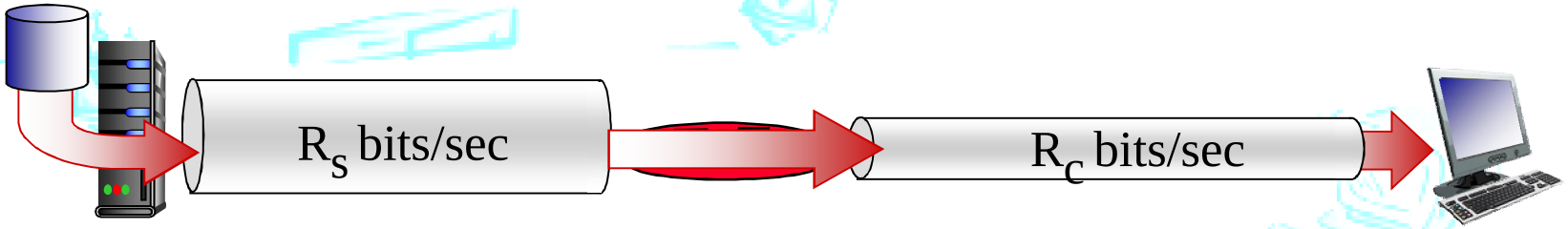


Läpäisy (Throughput)

- $R_s < R_c$ Mikä on keskimääräinen läpäisy päästä-päähän?



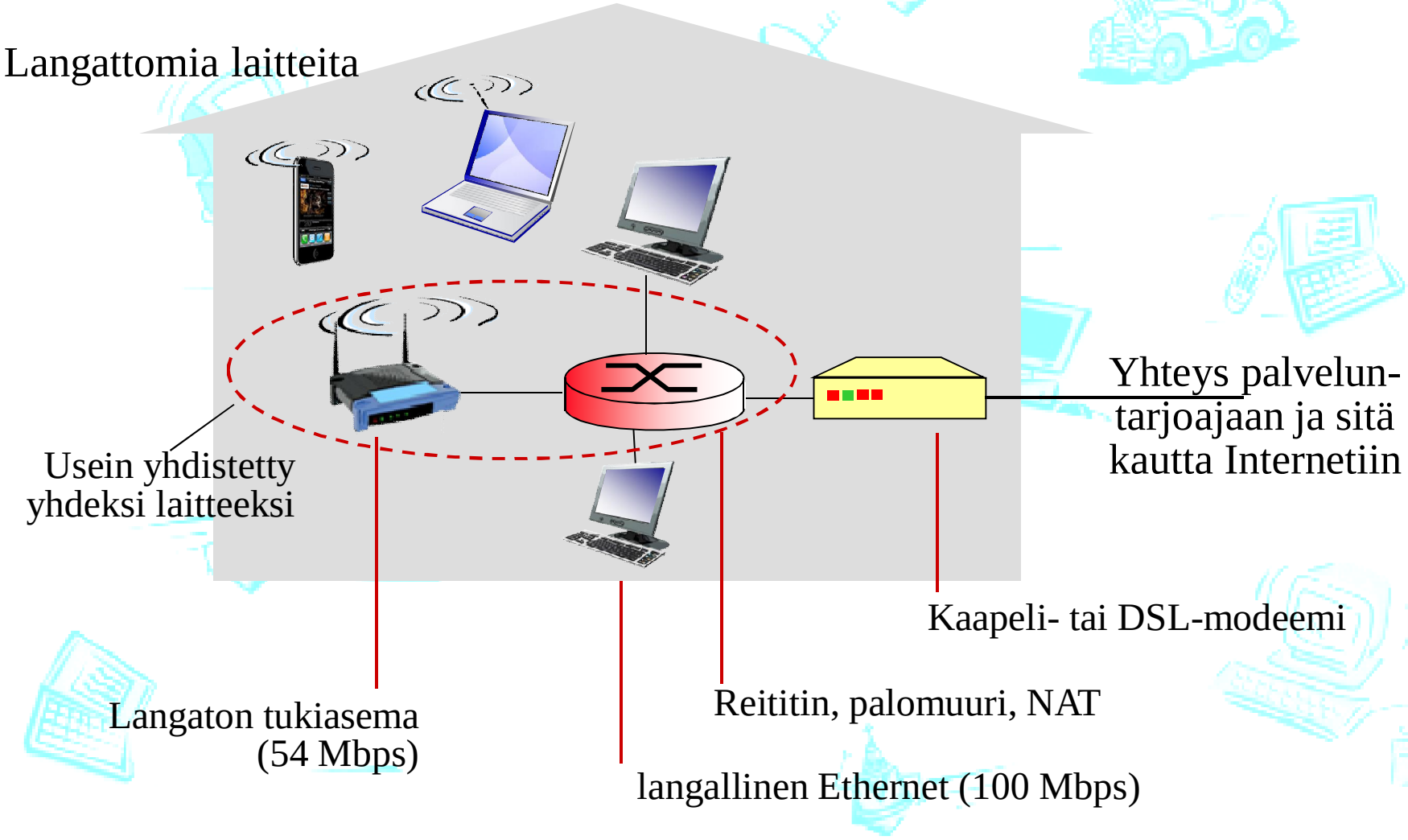
- $R_s > R_c$ Entä nyt?



Pullonkaula (bottleneck)

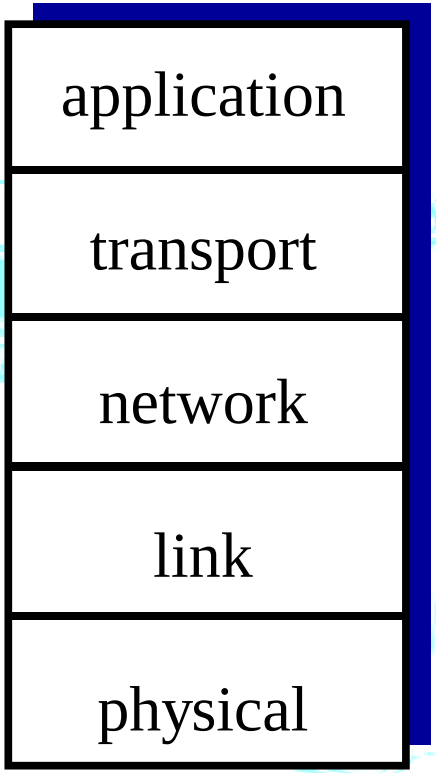
Linkki, joka rajoittaa läpäisyä päästä-päähän yhteydelle

Kotiverkko (home network)



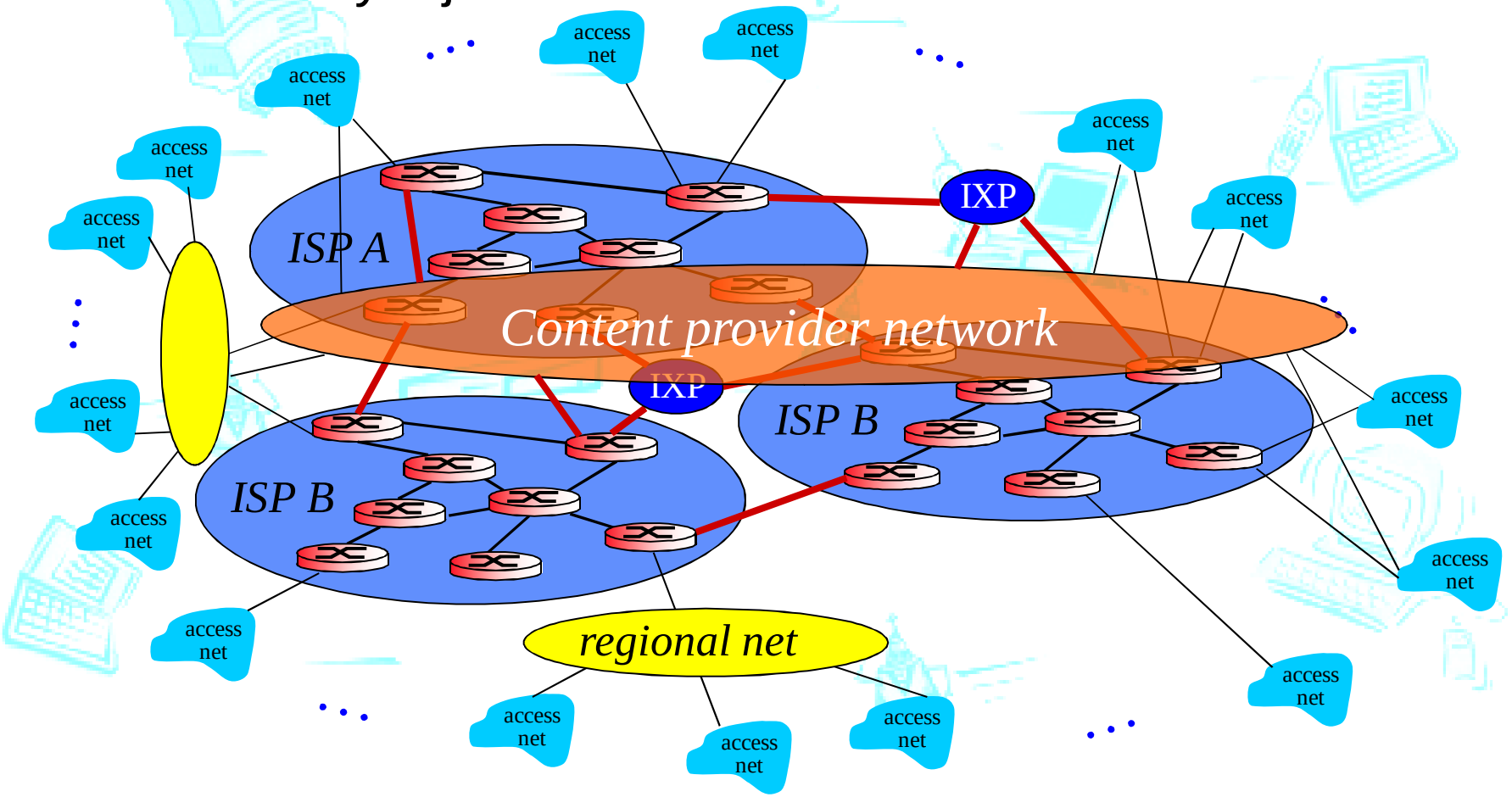
Internet-protokollapino

- *Sovellus:* verkkosovellusten omat protokollat
  - FTP, SMTP, HTTP, DNS,...
- *Kuljetus:* segmenttien siirto prosessilta toiselle (“päästä-päähän”)
  - TCP, UDP
- *Verkko:* pakettien reititys ja siirto verkossa lähettäjältä vastaanottajalle
  - IP, routing protocols
- *Linkki:* siirtää paketit kehyksinä kahden verkkolaitteen välillä
  - Ethernet, 802.111 (WiFi), PPP
- *Fyysinen:* generoi, siirtää ja vastaanottaa bittejä koneelta toiselle



Internetin rakenne: Verkkojen verkko

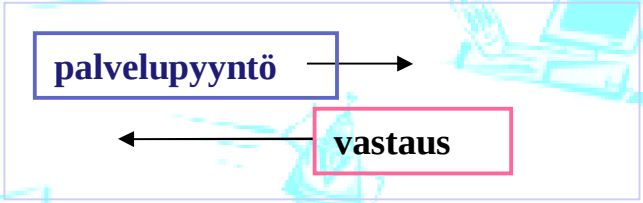
… ja sisällön tuottajien (content provider) (e.g., Google, Microsoft, Akamai) omia verkkoja, joilla palvelut lähemmäs käyttäjiä



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

Sovellusarkkitehtuuri

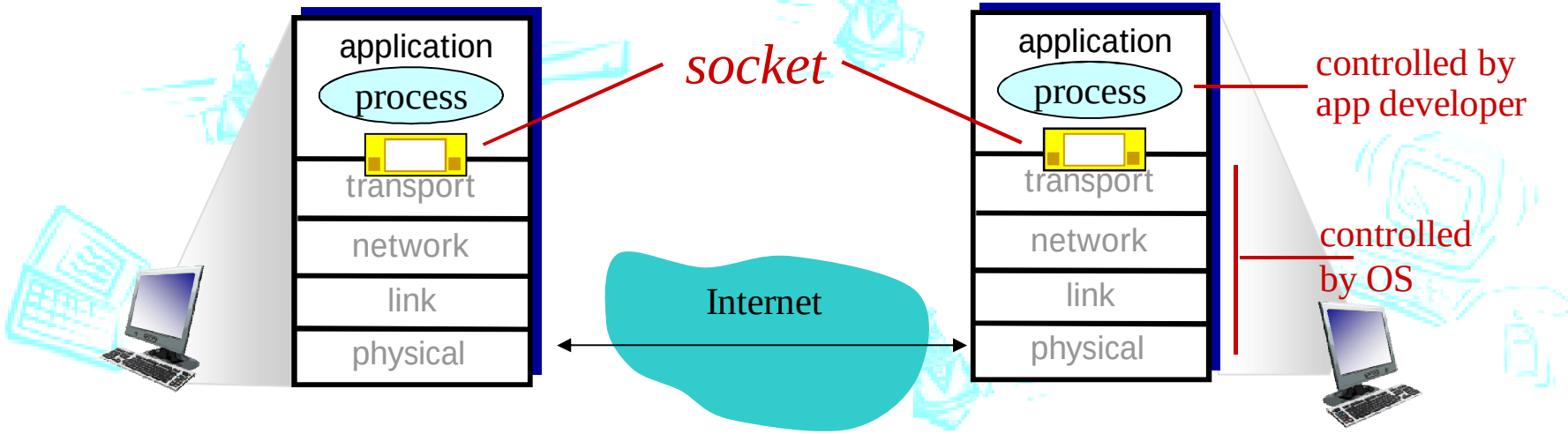
- **Asiakas-palvelija-malli** (esim. selain ja www-palvelin)
  - **Aina toiminnassa oleva palvelinohjelma, jolla kiinteä, tunnettu IP-osoite**
  - Asiakasohjelmat ottavat yhteyttä palvelimeen ja pyytävät siltä palvelua



- **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)

Sovelluksen rajapinta tieto-liikenteeseen: Pistokkeet (sockets)

- Prosessi lähettää ja vastaanottaa sanomia (messages) pistokkeen (socket) kautta
- Pistoketta voi ajatella ovena
  - Lähettävä prosessi tyrkkää sanoman ulos ovesta
  - Lähettävä prosessi luottaa että oven takana oleva kuljetuspalvelu toimittaa sanoman vastaanottajan ovelle



Osoittaminen

- Sanomissa oltava lähettäjän ja vastaanottajan IP-osoite ja porttinumero
- IP-osoite → oikea kone
  - koneen (verkkokortin) yksilöivä 32-bittinen tunniste
  - osoitteen verkko-osa yksilöi verkon
  - osoitteen koneosa yksilöi koneen verkossa
- Porttinumero → oikea prosessi
  - Yleisillä palveluilla standardoidut tunnetut porttinumerot:
    - www-palvelin kuuntelee porttia 80,
    - Postipalvelin kuuntelee porttia 25
  - KJ osaa liittää porttinumeron prosessiin

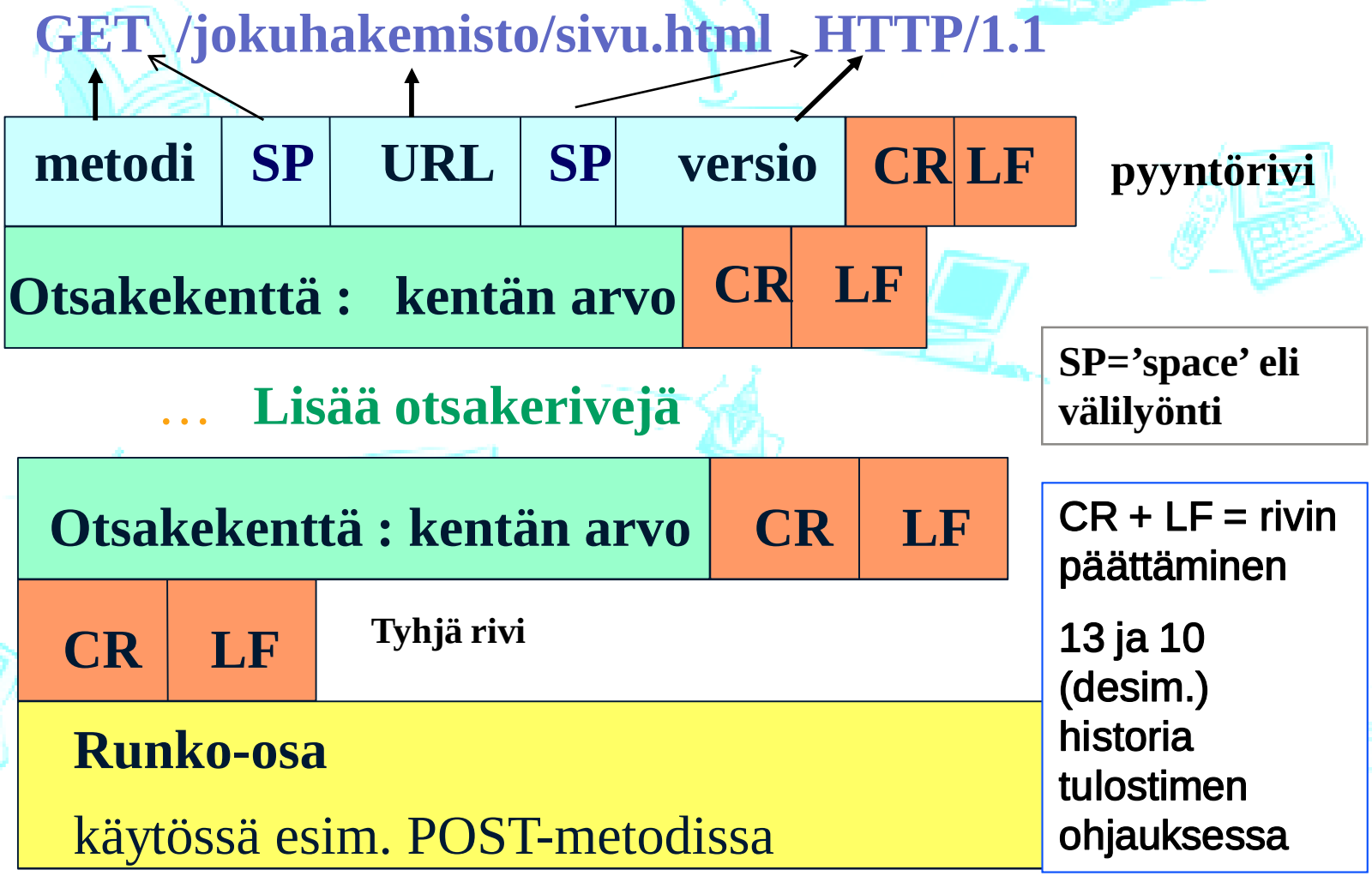
www.iana.org

HTTP (HyperText Transfer Protocol) [RFC 1945, RFC 2616]

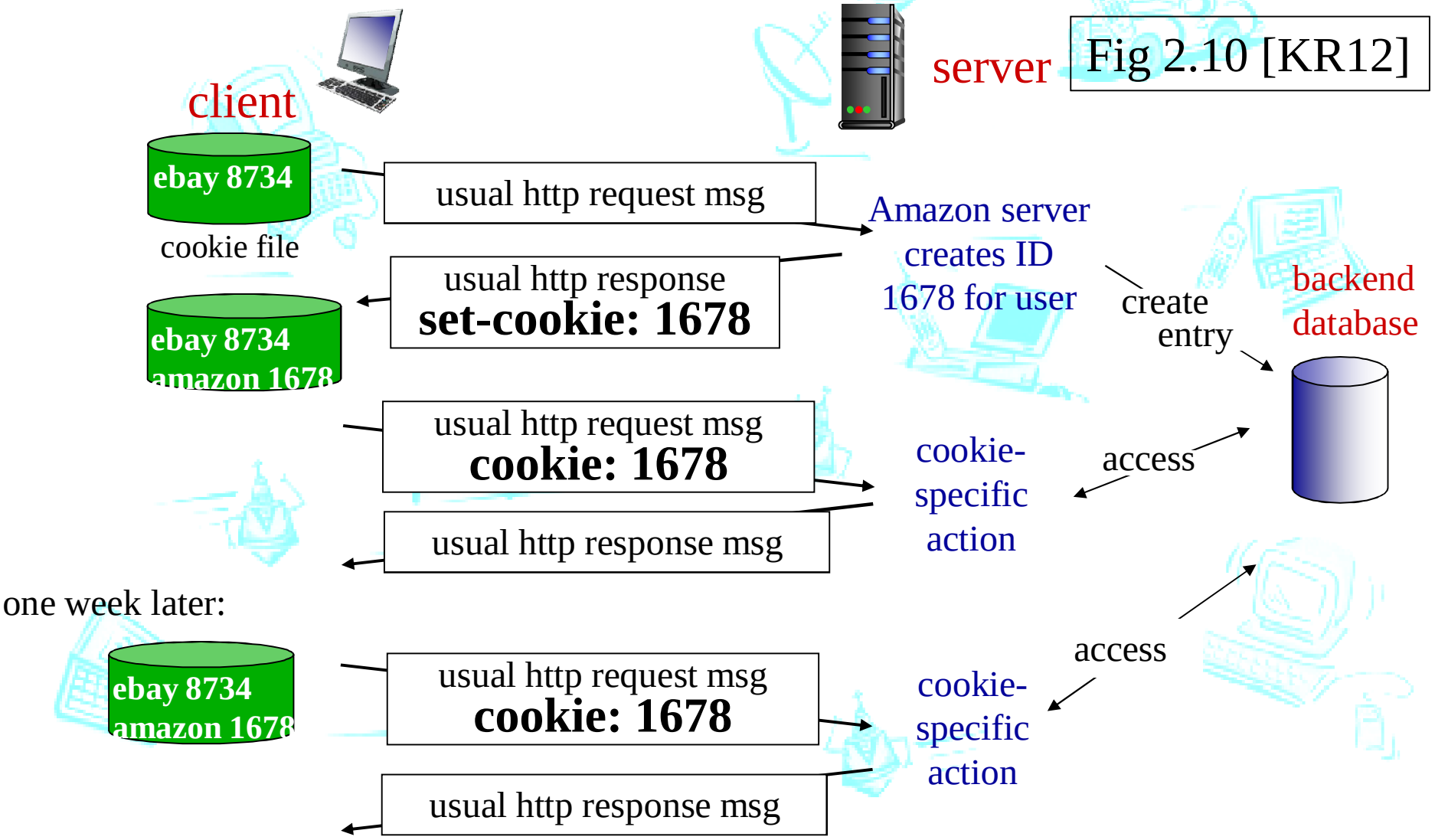
- WWW:n sovellusprotokolla
  - Tekstimuotoiset sanomat
  - pyyntö – vastaus
- Asiakas = joku selain
  - pyytää, noutaa ja näyttää objektit
- Palvelija = joku web-palvelu
  - etsii objektin (tiedoston) koneen hakemistosta
  - ja lähettää sen vastauksena asiakkaalle

Fig 2.6 [KR12]

HTTP-pyyntö: yleinen rakenne



Evästeet: esimerkki



Domain Name System (DNS)

- Hakemistopalvelu ja sovelluskerroksen protokolla
  - Isäntäkoneet ja nimipalvelimet käyttävät
  - Käyttää UDP-kuljetuspalvelua DNS-sanomien kuljettamiseen
- Hajautettu, hierarkinen tietokanta (hakemisto)
  - Toteutettu useiden replikoitujen nimipalvelimien yhteistyönä
  - skaalautuvuus, kuormantasaus, ylläpito, vikasietoisuus, ..
  - Jos oma nimipalvelija ei tunne, se kysyy muilta.
- Nimien muuttaminen IP-osoitteiksi (ja päinvastoin)
  - POSIX: gethostbyname

```
gethostbyname (hydra.cs.helsinki.fi)
218.214.4.29
```

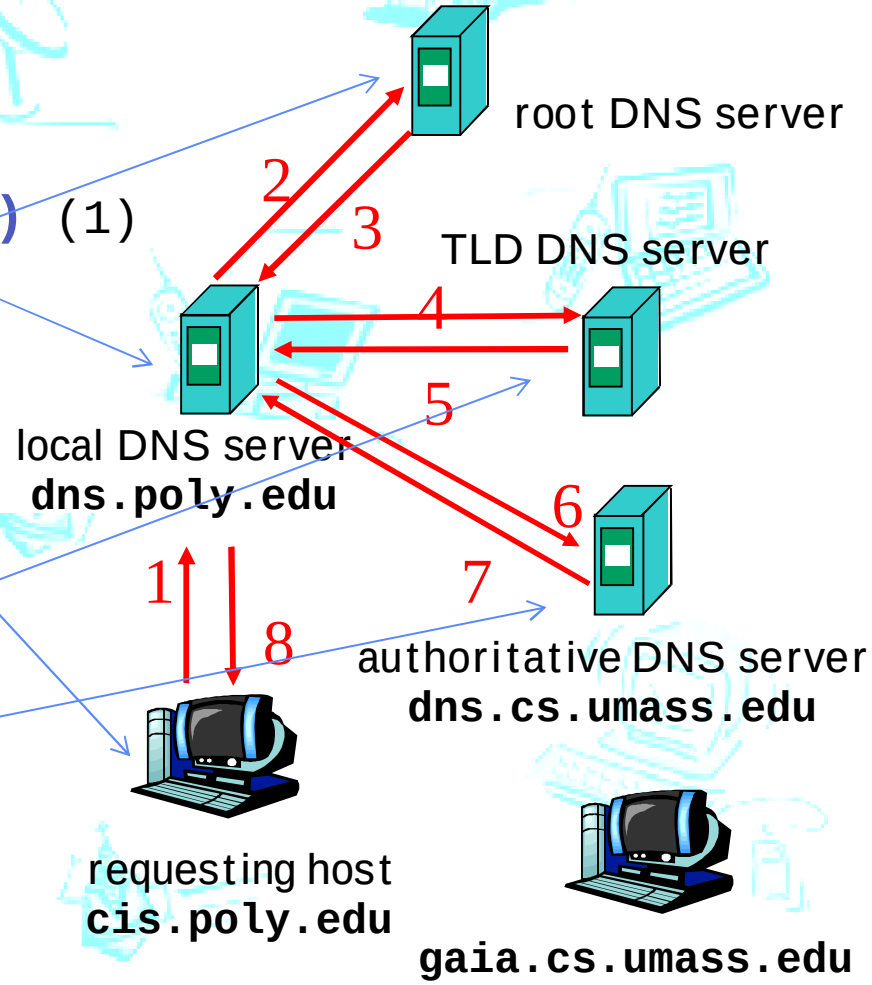
  - Kone = hydra =29, verkko= cs.helsinki.fi = 218.214.4.0
- Sallii aliasnimet, palvelijan replikoinnin/toisintamisen
  - Esim. WWW.cs.helsinki.fi ja cs.helsinki.fi ovat aliasnimiä
  - Esim. www-palvelijaan voi liittyä useita IP-osoitteita, rotaatio

Iteratiivinen kysely: “kerro keneltä pitää kysyä?”

Mikä on gaia.cs.umass.edu:n IP-numero?

Fig 2.21 [KR12]

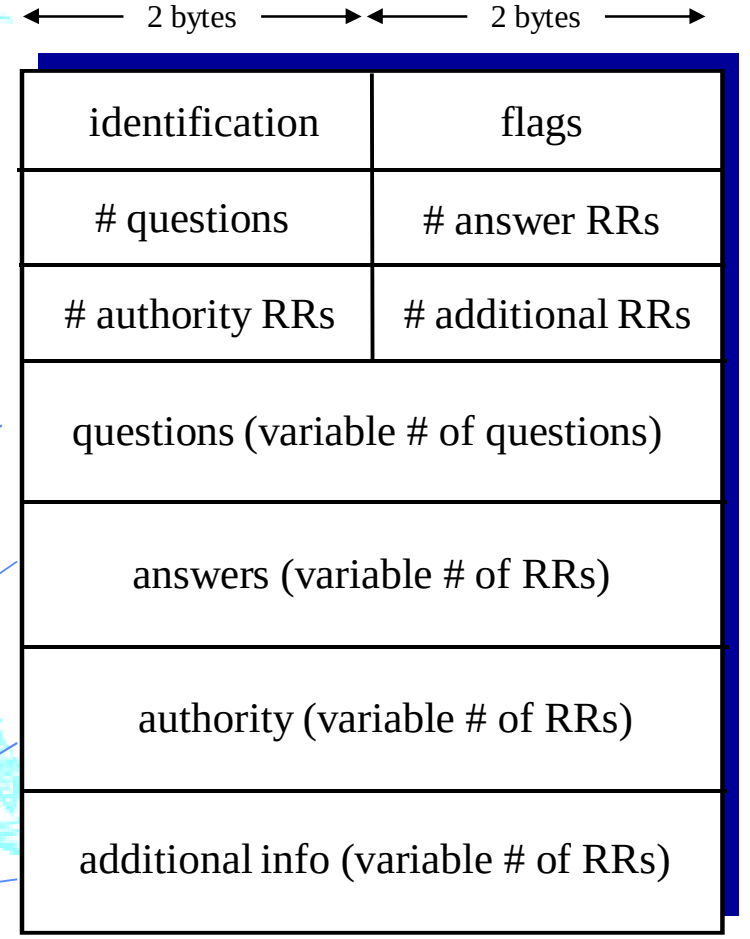
- Isäntäkone
  - Kysy omalta aluepalvelijalta
- Paikallinen nimipalvelija **(poly)** (1)
  - Ratkaise isäntäkoneen puolesta
- Juuripalvelin (3)
  - Kerro, mistä löytyy ylätason palvelin **edu**-tunnuksille
- Ylätason palvelin **(edu)** (4,5)
  - Kerro, mistä löytyy aluepalvelija **umass.edu**-tunnuksille
- Aluepalvelija (6,7)
  - Tuntee **cs**-verkon koneet.
  - Kerro koneen IP-osoite



DNS-sanoma

- Kyselystä voi generoitua vastaus, jossa on useita resurssitietueita
  - Esim. Palvelijafarmien kuormantasaaminen: vastauksessa on useita IP-osoitteita (rotaatio)

Fig 2.23 [KR12]



Kyselyalueella etsittävän nimi ja tyyppi

Vastausalueella (useita) resurssitietueita, jotka liittyvät kysyttyyn nimeen

Tietueita muihin autorisoituihin palvelijoihin

Ylim. hyödyllisiä resurssitietueita

rdt3.0: vuorottelevan bitin protokolla

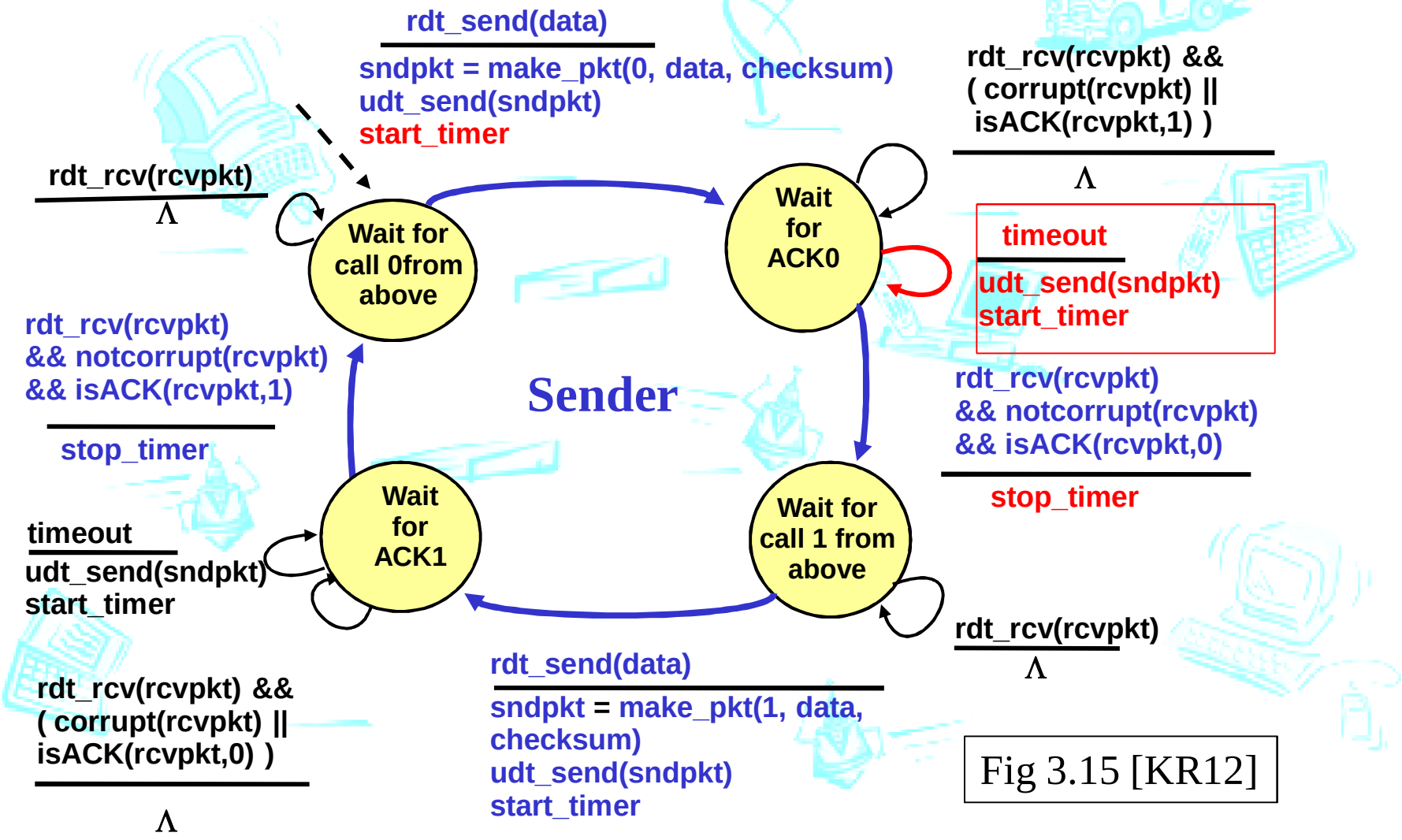
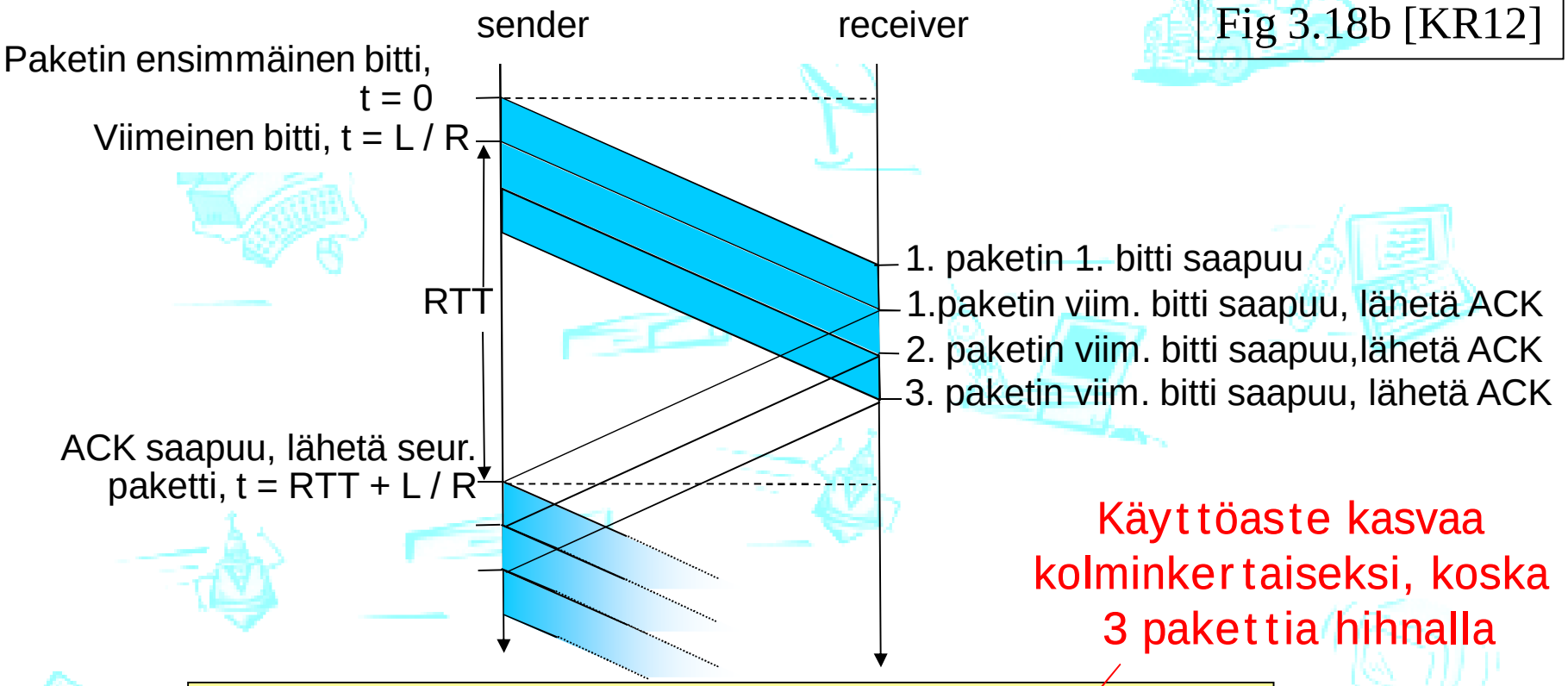


Fig 3.15 [KR12]

Liukuhihnoitus: käyttöasteen kasvattaminen

Fig 3.18b [KR12]



Käyttöaste kasvaa kolminkertaiseksi, koska 3 pakettia hihnalla

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

ilman liukuhihnaa: 0.00027

Liukuva ikkuna

- 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
- Vastaanottoikkuna (receiver window)
  - Mitkä pakettinumerot otettu vastaan, mutta kuittaamatta
  - Mitä pakettinumeroita lähettäjä saa vielä käyttää eli mitkä hyväksytään
- Jos saadussa paketissa on ikkunan viimeinen numero
  - Ikkuna pysäyttää pakettien lähetyksen vastapäästä
  - Ikkuna estää uusien pakettien vastaanoton
- Paketin kuittaus liu'uttaa myös vastaanottajan ikkunaa
  - Hyväksytään uusia pakettinumeroita

Ikkunankoko

- Pakettinumeroille varatun kentän koko vaikuttaa myös ikkunankokoon
  - Yleensä jokin kakkosen potenssi
  - Kentän koko k bittiä => käytössä 2**k pakettinumeroa
- Kun paketit numeroidaan 0, 1, …, n, niin ikkunan koko saa olla korkeintaan:

n+1 kpl

MIKSI NÄIN?

Harjoitus-tehtävinä!

Go-Back-n: **n** (k bit sekvenssinumeroilla < 2^k-1)

Valikoiva toisto: **(n+1)/2** (k bit sekv.num. < 2^{k-1})

Vastaanottajan pitää tietää onko kyse uusista paketeista vai uudelleenlähetyksistä

Yhteenveto menetelmistä

- Tarkistussumma
- Ajastin
- Järjestysnumero
  - Uudelleenlähetys vai uusi paketti
- Kuittaukset
  - Positiiviset ACK, tuplakuittaukset
  - Negatiiviset NAK
- Ikkunat, liukuhihnoitus

Kts Table 3.1 [KR12]

ACK vai NAK

- ACK ja NAK signaaleja voidaan yhdistellä
- TCP perustuu vain ACKiin
- TCP: Kumulatiivinen ACK
  - Tupla ACK X = NAK X+1
- Selektiivinen ACK
  - Parempi toteuttaa oma NAK signaali, tehokkaampi
  - Ei välttämättä tarvita NAKia
- Pelkkä NAK

UDP (User Datagram Protocol) (RFC 768)

- **Yhteydetön**
  - KJ ei pidä tallessa mitään sovellusten väliseen keskusteluun liittyvää.
  - Sovellus antaa aina sanoman lisäksi kohdeosoitteen ja kohdeportin
- **Ei takaa luotettavuutta** (~'epäluotettava'?)
  - vain minimipalvelu: mille koneelle, mihin porttiin, voi kadottaa sanoman
- **Ei säilytä sanomien järjestystä**
  - Sovellus saa sanomat siinä järjestyksessä kuin ne tulevat perille
- **Vähän yleisrasitetta**
  - Aikaa ei kulu yhteyden muodostukseen eikä purkuun
  - Ei resursseja yhteyden tilatietojen ylläpitoon
  - Pieni otsake (eli vähän itse protokollaan liittyviä tietoja)
  - Ruuhkanhallinta ei säännöstele liikennettä

Kuljetusprotokollat: TCP

(Transmission Control Protocol) [RFC 793]

- Yhteydellinen palvelu (connection-oriented)
  - Yhteyden muodostus ennen datan siirtoa (handshaking)
  - Kaksisuuntainen TCP-yhteys (full-duplex)
  - Yhteyden purku (shutdown)
- Luotettava kuljetuspalvelu
  - Järjestyksen säilyttävä tavuvirta sovellukselle
    - segmenttinumerot, kuittaukset, uudelleenlähetykset
- Vuonvalvonta (flow control)
  - Lähettäjä hiljentää vauhtia, jos **vastaanottaja** ei ehdi käsitellä
- Ruuhkanvalvonta (congestion control)
  - Lähettäjä hiljentää vauhtia, jos **reitittimet** eivät ehdi käsitellä

TCP-segmentti

Fig 3.29 [KR12]

Otsake aina vähintään 20 B
Optio-osa tarvittaessa

Järjestys- ja kuittaus-
numerot tavunumeroina

Ikkunankoko: paljonko tilaa
vastaanottopuskurissa (tavua)

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

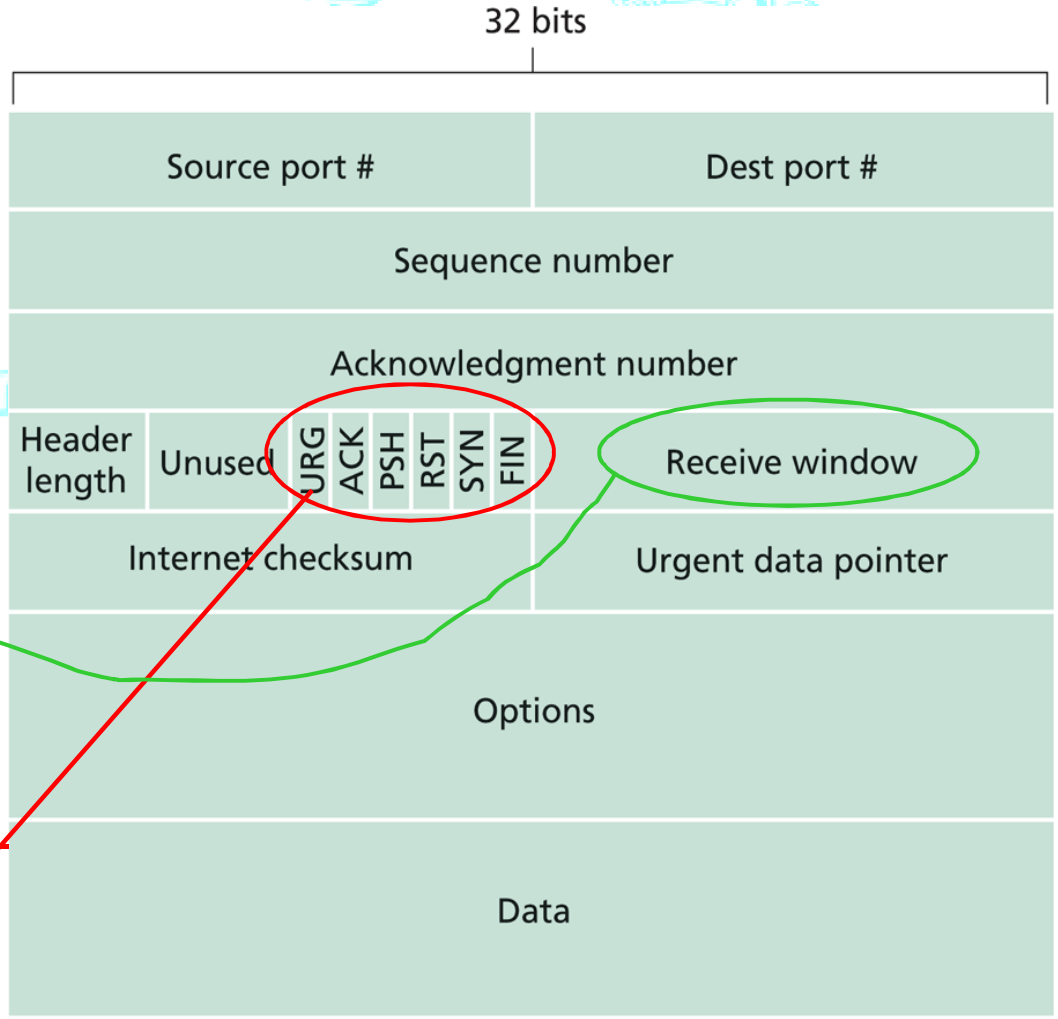


Figure 3.29 ♦ TCP segment structure

Tavunumerointi

Kuittauksia ei kuitata ja ne eivät siirrä numerointia!

Niissä ei siirretä tavuvirtaa.

Tavuvirtaa …

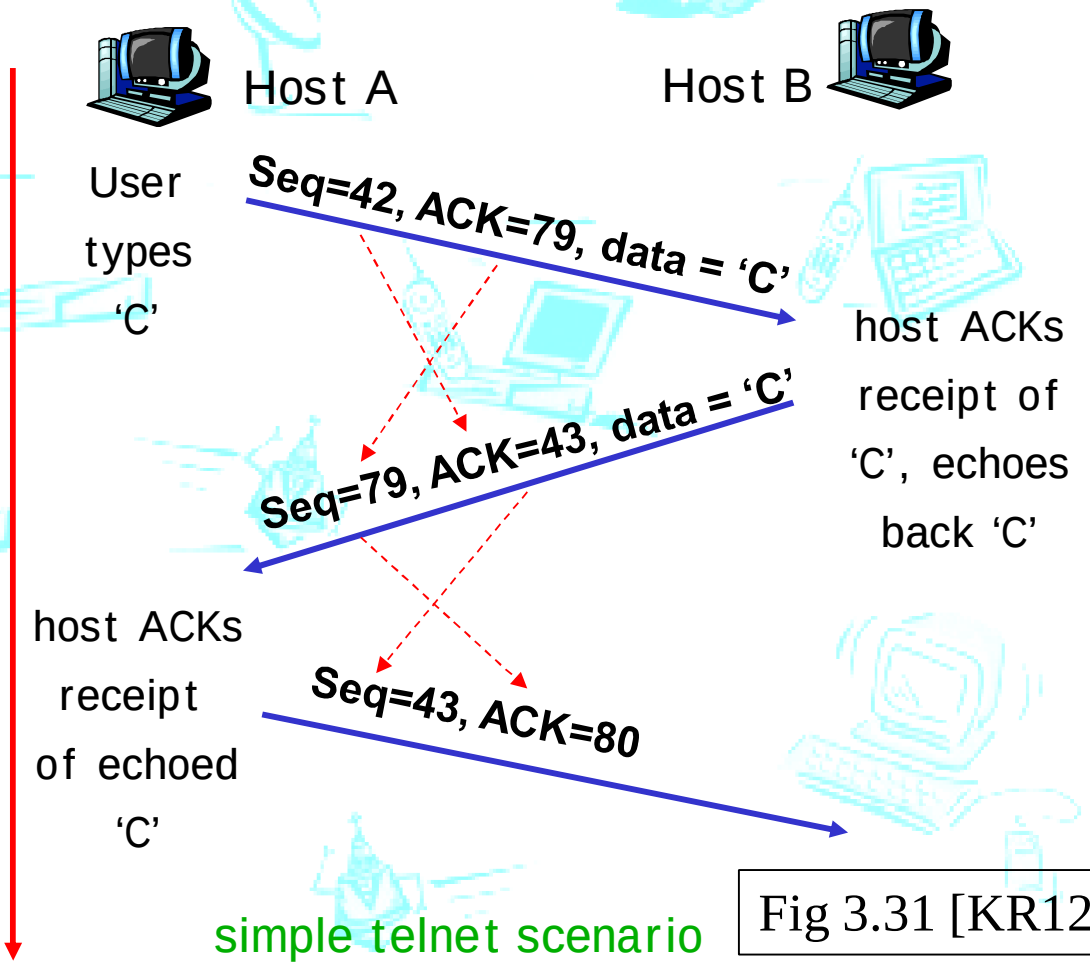
Segmentit voivat olla erikokoisia (=< MSS)

Järjestysnumero=

- ensimmäisen tavun numero
- alkuarvot sovitaan yhteyttä muodostettaessa

Kuittaus

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



simple telnet scenario

Fig 3.31 [KR12]

TCP: uudelleenlähetys

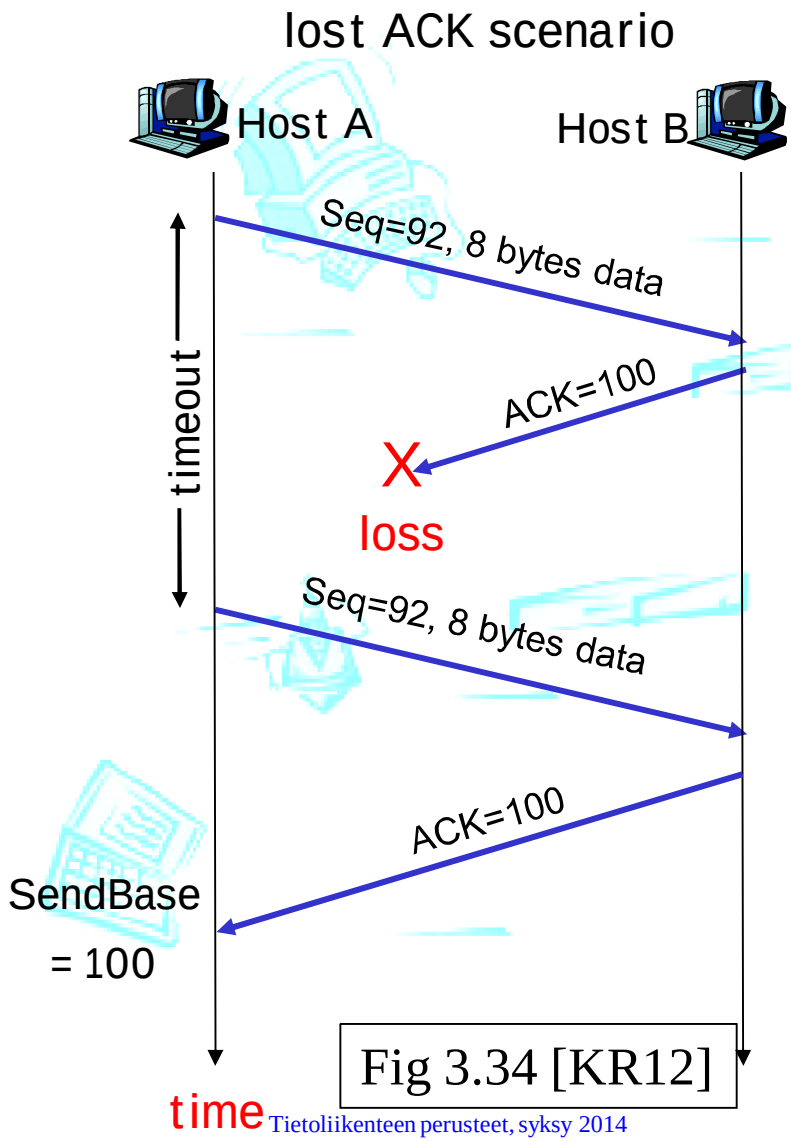


Fig 3.34 [KR12]

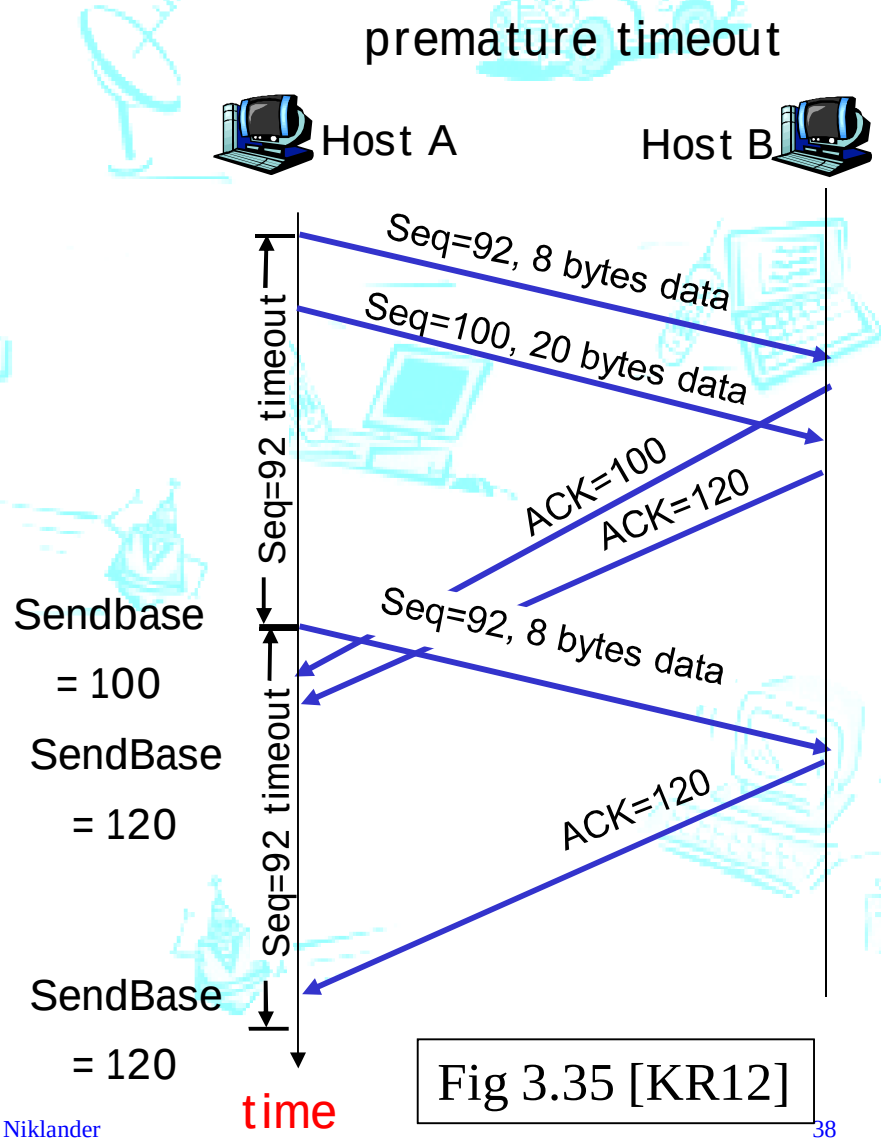
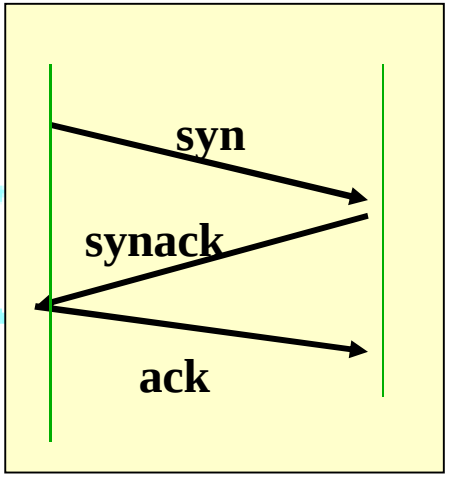


Fig 3.35 [KR12]

Yhteyden muodostus

- Kolmivaiheinen kättely (three-way handshake)
  - 3 segmenttiä: SYN – SYNACK – ACK
  - Otsakkeen bittikentät
  - Viimeinen voi sisältää dataa (piggybacked)
  - Jos porttiin ei liity prosessia, vastaukseksi segmentti eli yhteyttä ei voida muodostaa



- **Varaa puskuritilaa**
  - Lähettäjä puskuroi uudelleenlähetystä varten
  - Vastaanottaja saatujen pakettien järjestämistä varten
- **Sovitaan tavunumeroinnin alkuarvoista**
  - Kuittauksia varten
- **Ilmoita oma vastaanottoikkunan koko**
  - Vuonvalvontaa varten

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

- **Aluksi ruuhkaikkuna = yksi segmentti**
  - Alussa hidas siirtonopeus= MSS/RTT
- **Kukin kuittaus kasvattaa ruuhka-ikkunan kokoa yhdellä**
  - Eksponentiaalinen kasvu
  - Ikkuna koko kaksinkertaistuu yhden RTT:n aikana
- **Kun ruuhkaikkunan kynnysarvo saavutettu, kasvata ikkunaa vain yksi segmentti/RTT**
  - Lineaarinen kasvu (Additive increase)
  - Ruuhkan välttely (congestion avoidance)
- **Jos uudelleenlähetys, ruuhkaikkunan kooksi 1 segmentti**
  - Multiplicative decrease
- Siirtonopeus = CognWin / RTT tavua/sek

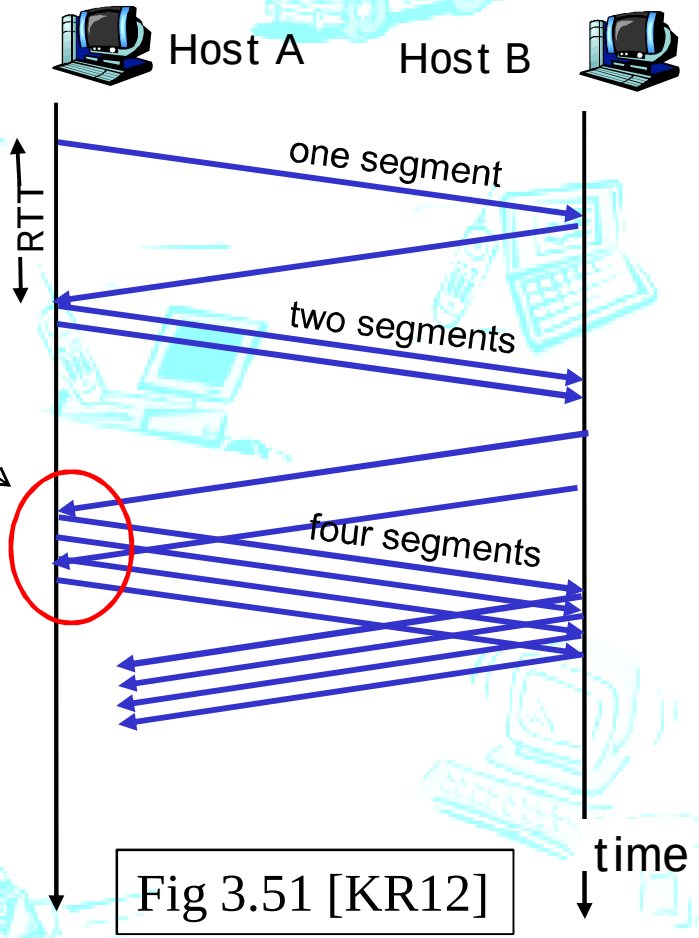


Fig 3.51 [KR12]

RTT- round-trip time, kiertoviive

TCP Reno: Toipuminen paketin katoamisesta (eri tapoja havaita ja toipua)

- **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?

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

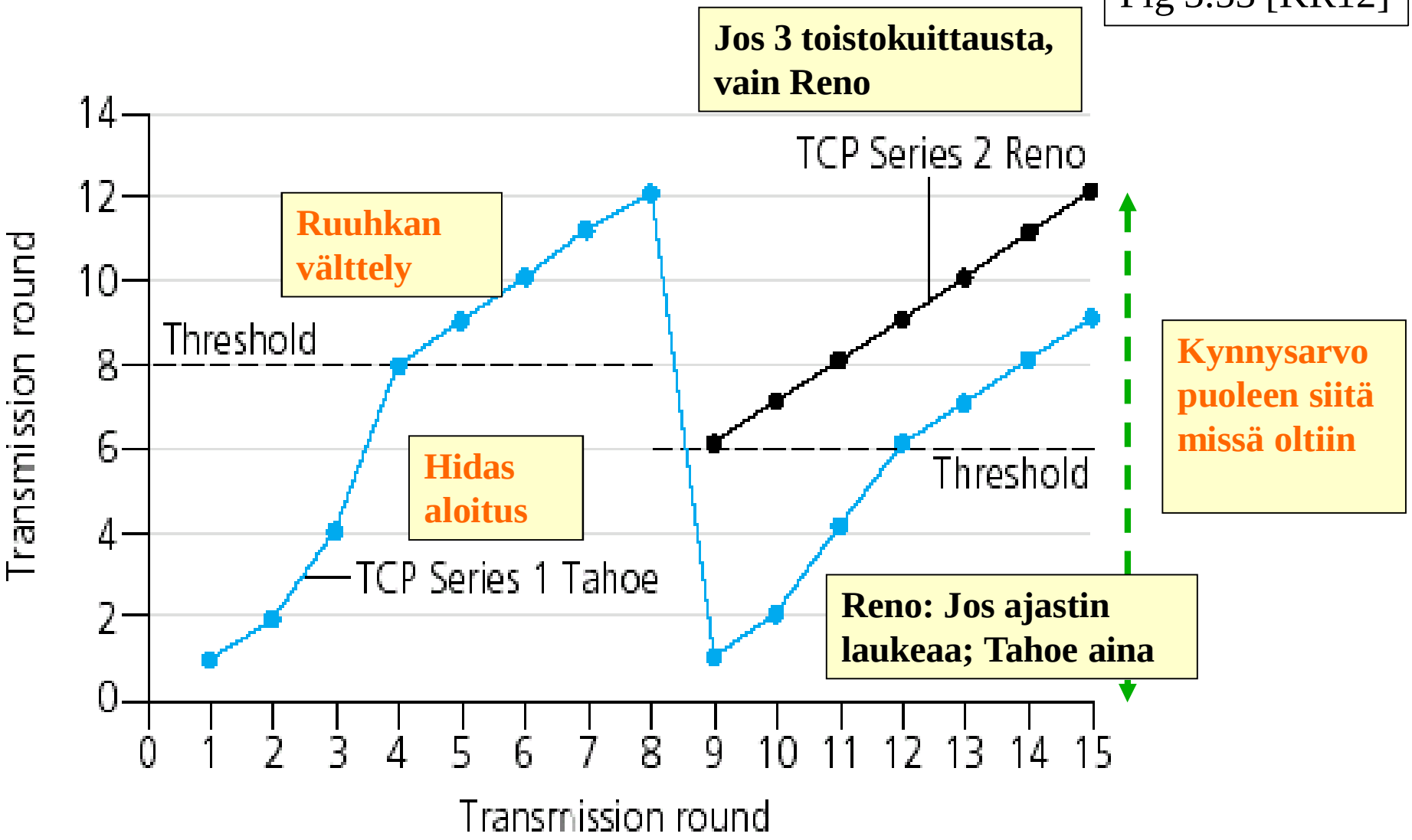
Vanha TCP Tahoe osasi vain aikakatkaisun.

Hidas aloitus

ruuhkanvälttely

TCP Reno: Ruuhkaikkunan koko

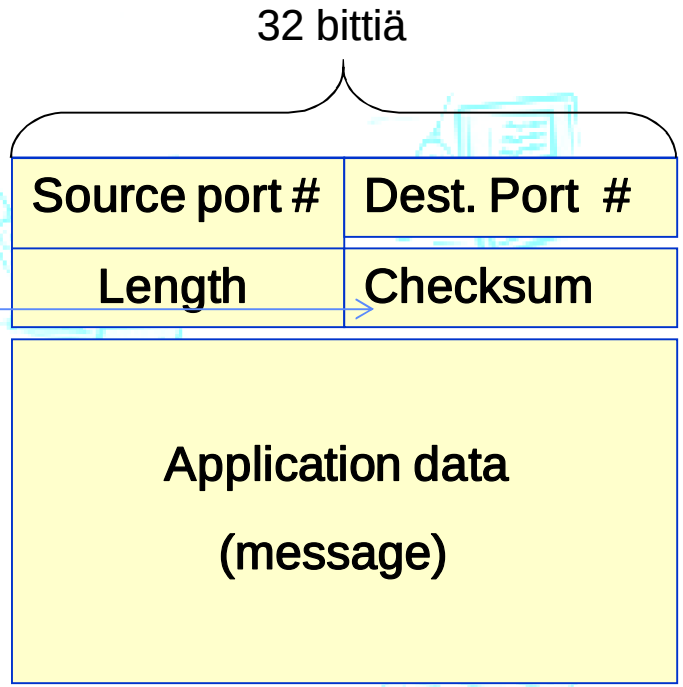
Fig 3.53 [KR12]



UDP-segmentin rakenne

- Porttinumerot
  - Prosessien välinen palvelu
- Pituus
  - Segmentin kokonaispituus
  - otsake (8 B) mukaanluettuna
- Tarkistussumma (optionaalinen)
  - Bittivirheen havaitsemiseen
  - UDP ei yritä toipua, hävittää virheellisen segmentin
- Data
  - Pitkä sanoma pilkottuna useaksi segmentiksi
- IP-osoitteet vasta verkkokerroksen otsakkeessa
  - Näitä tarvitaan reitityksessä

Fig 3.7 [KR12]

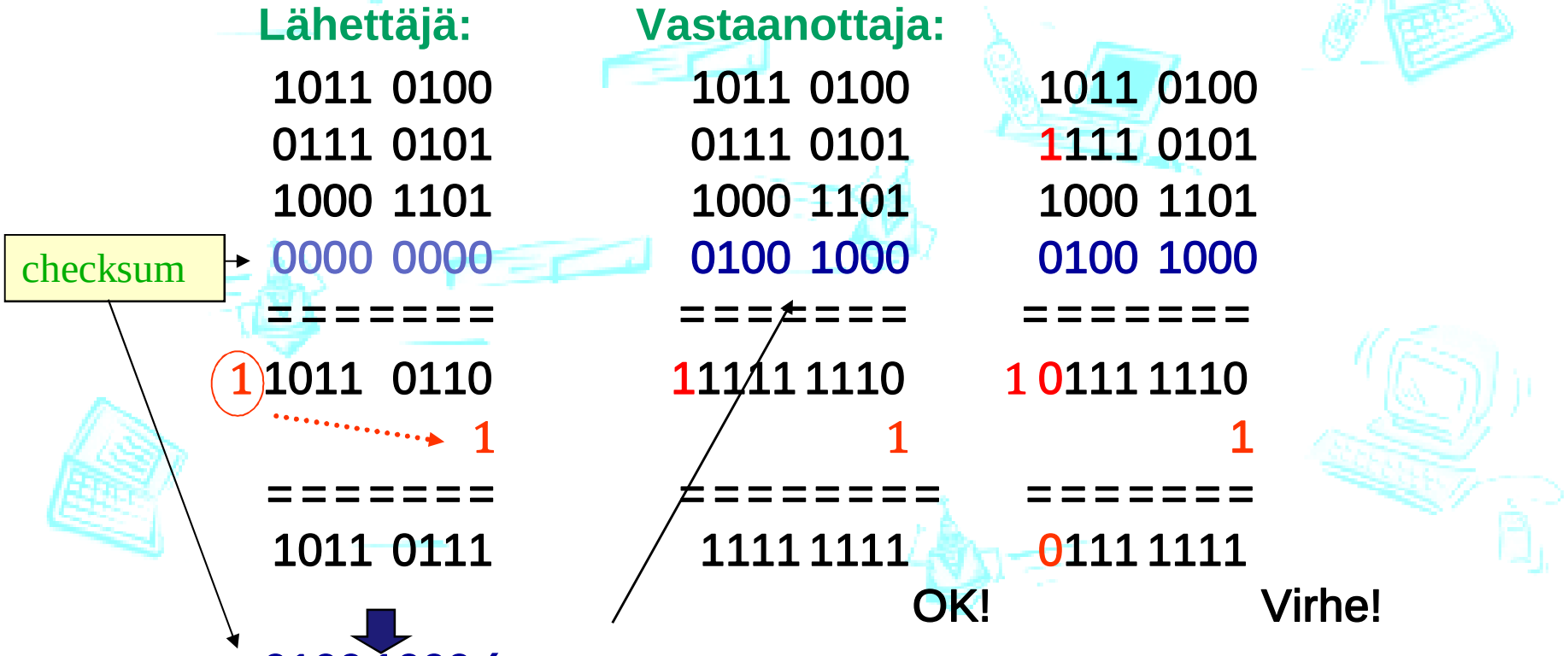


UDP-segmentti

Esimerkki

0+0 = 0 no carry
1+0 = 1 no carry
0+1 = 1 no carry
1+1 = 0 with carry

- Lasketaan tarkistussumma kolmen tavun mittaiselle sanomalle (tässä vain 8 bitin mittaisena):

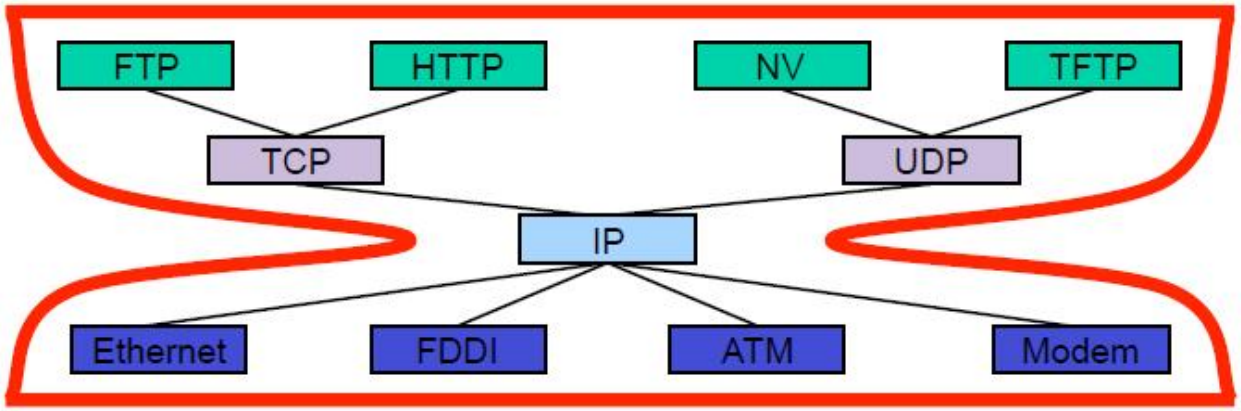


Verkkoprotokolla: IP

”Everything over IP, IP over everything”

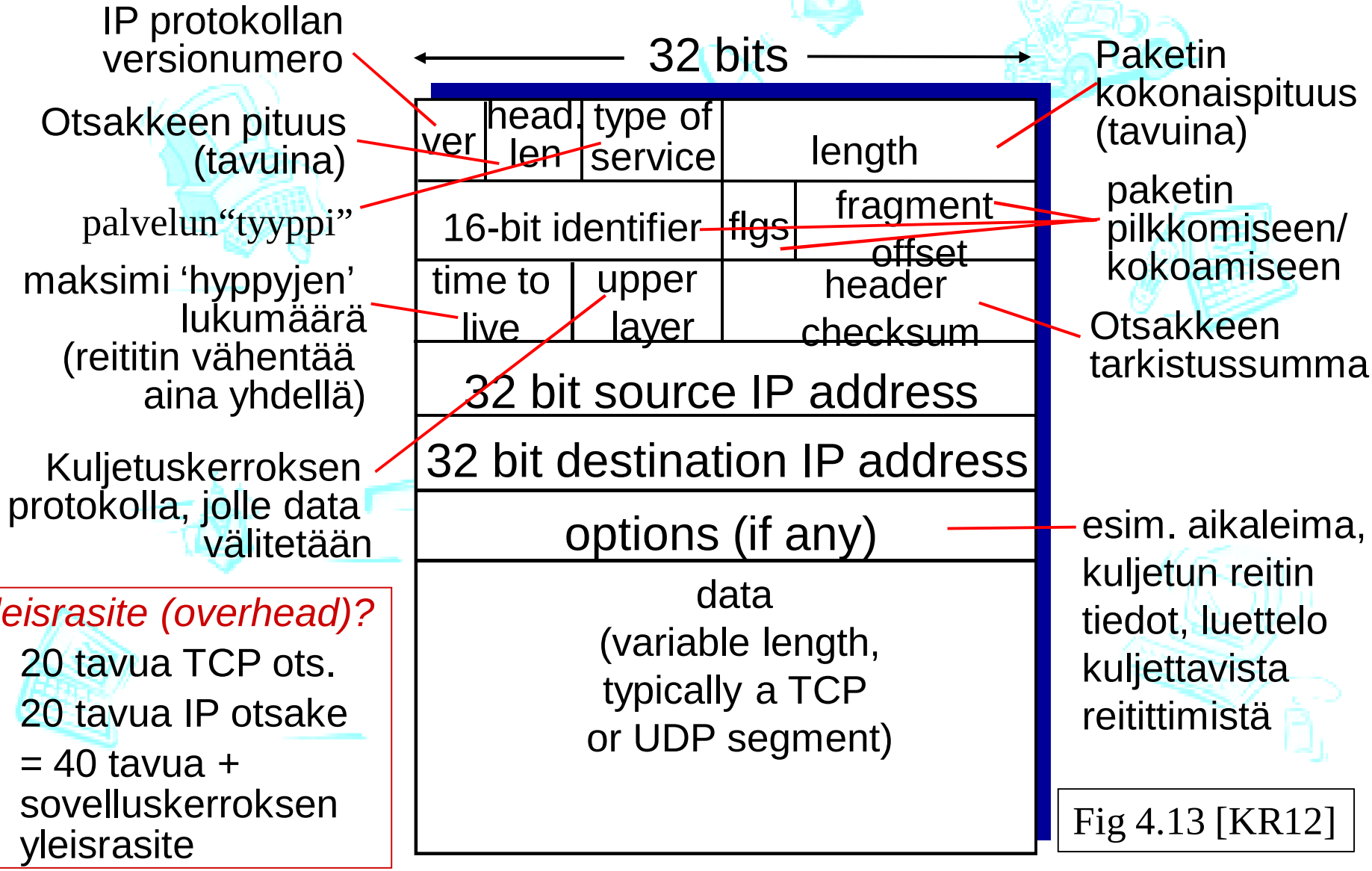
- Verkkoprotokolla IP (Internet Protocol) on verkkokerroksen yhteinen kieli
  - Internetin isäntäkoneiden ja reitittimien kommunikointitapa
    - “Kullakin omat murteensa ja tapansa, mutta kaikki osaavat IP:tä.”
- Yhteinen osoitustapa: IP-osoite
  - Yksikäsitteiset osoitteet

Prof. B. Godfrey, 4.3.2011, luentokalvo, Univ. Illinois



IPv4 paketin rakenne

otsake 20+ tavua



Yleisrasite (overhead)?

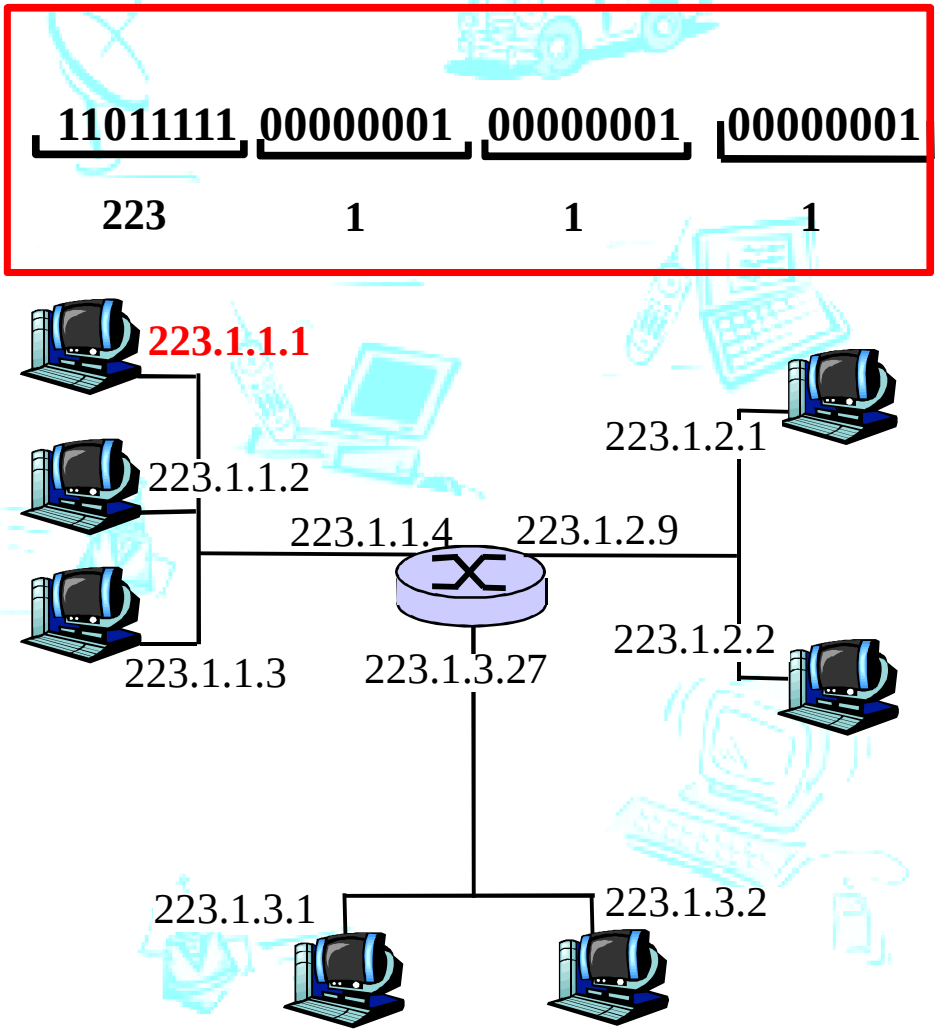
- 20 tavua TCP ots.
- 20 tavua IP otsake
- = 40 tavua + sovelluskerroksen yleisrasite

Fig 4.13 [KR12]

IP-osoitteet (IPv4)

Fig 4.15 [KR12]

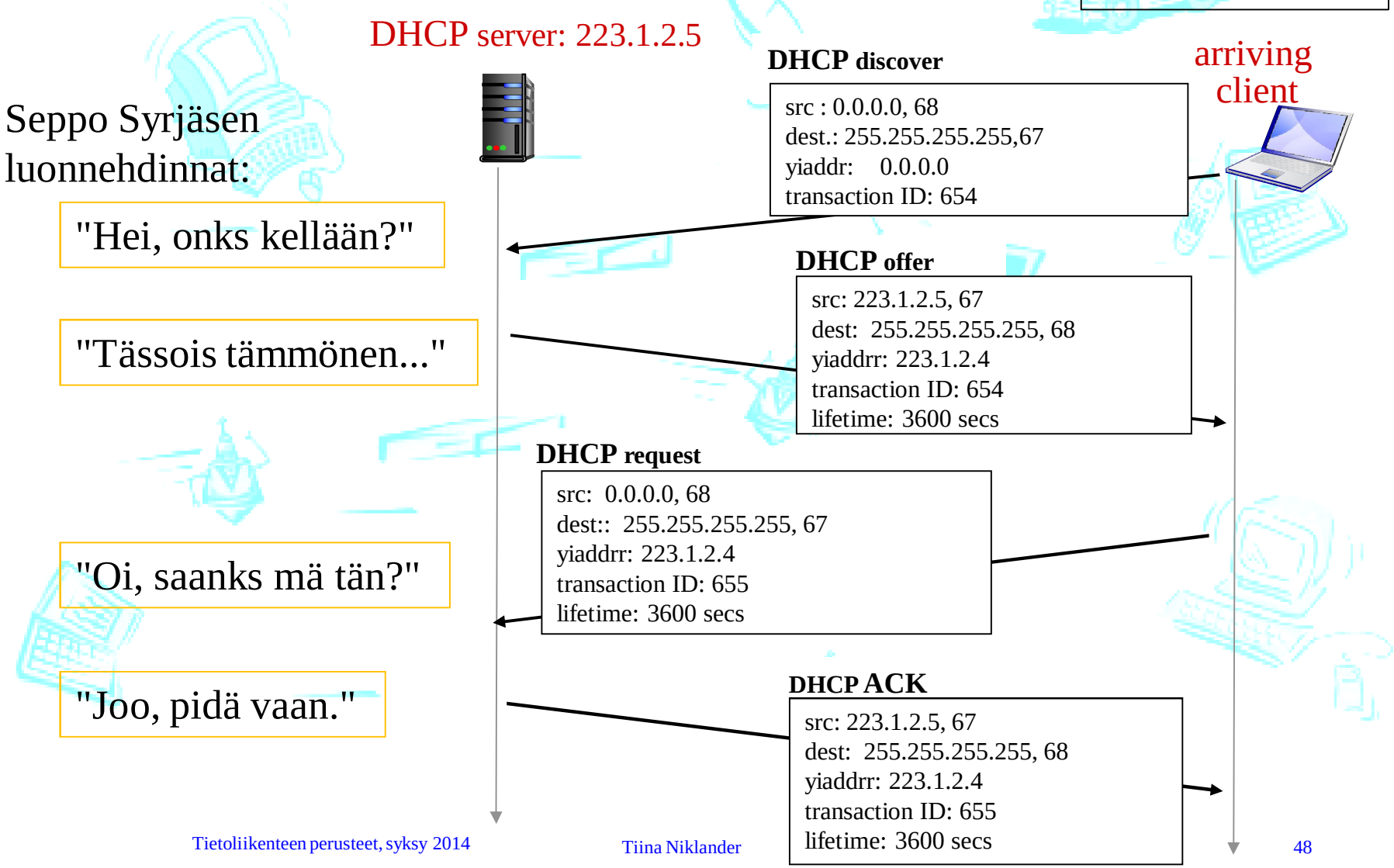
- 32-bittinen tunniste isäntäkoneille ja reitittimien linkeille
  - verkkoliittymän tunniste
- Reitittimellä useita liittymiä
  - kullakin oma IP-osoite
- Myös isäntäkone voi olla liitettynä useaan verkkoon



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

DHCP: Osoitekysely

Fig 4.21 [KR12]



Seppo Syrjäsen luonnehdinnat:

"Hei, onks kellään?"

"Tässois tämmönen..."

"Oi, saanks mä tän?"

"Joo, pidä vaan."

Aliverkon osoitteiden piilottaminen yhden julkisen IP-osoitteen taakse

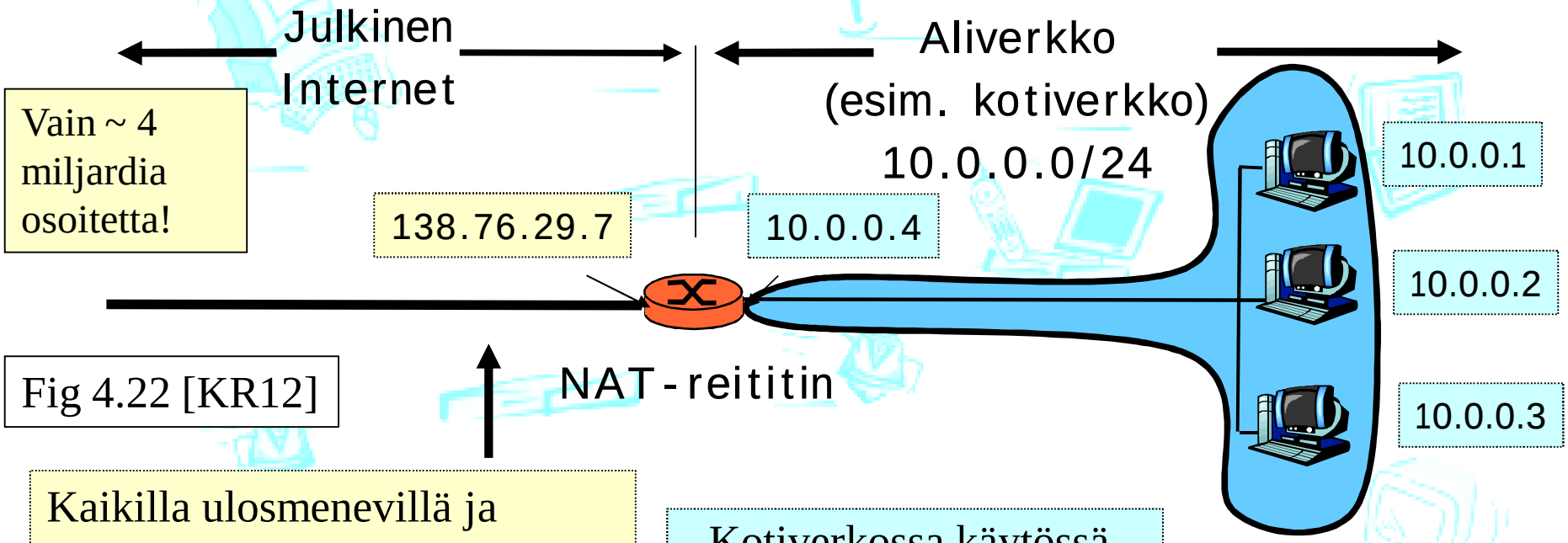


Fig 4.22 [KR12]

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

Kotiverkossa käytössä sisäiset IP-osoitteet
10.0.0.0/24
(esim. DHCP:llä)

Sisäisiä ovat:
10.0.0.0/8
172.16.0.0/12
192.168.0.0/16

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

Keskitin (hub)
Perustuu toistimiin
Fyysisellä kerrokselle

Paketin välitys ja reititys

- Reitittimen kaksi tehtävää:
- Paketin välitys (forwarding)
  - Paikallinen päätös
  - Mihin ulosmenolinkkiin paketti lähetetään
  - Katsoo linkin reititystaulusta
- Reititys (routing)
  - Globaali päätös
  - Mitä reittiä paketti kulkee lähettäjältä vastaanottajalle
  - Reititystaulun ylläpito

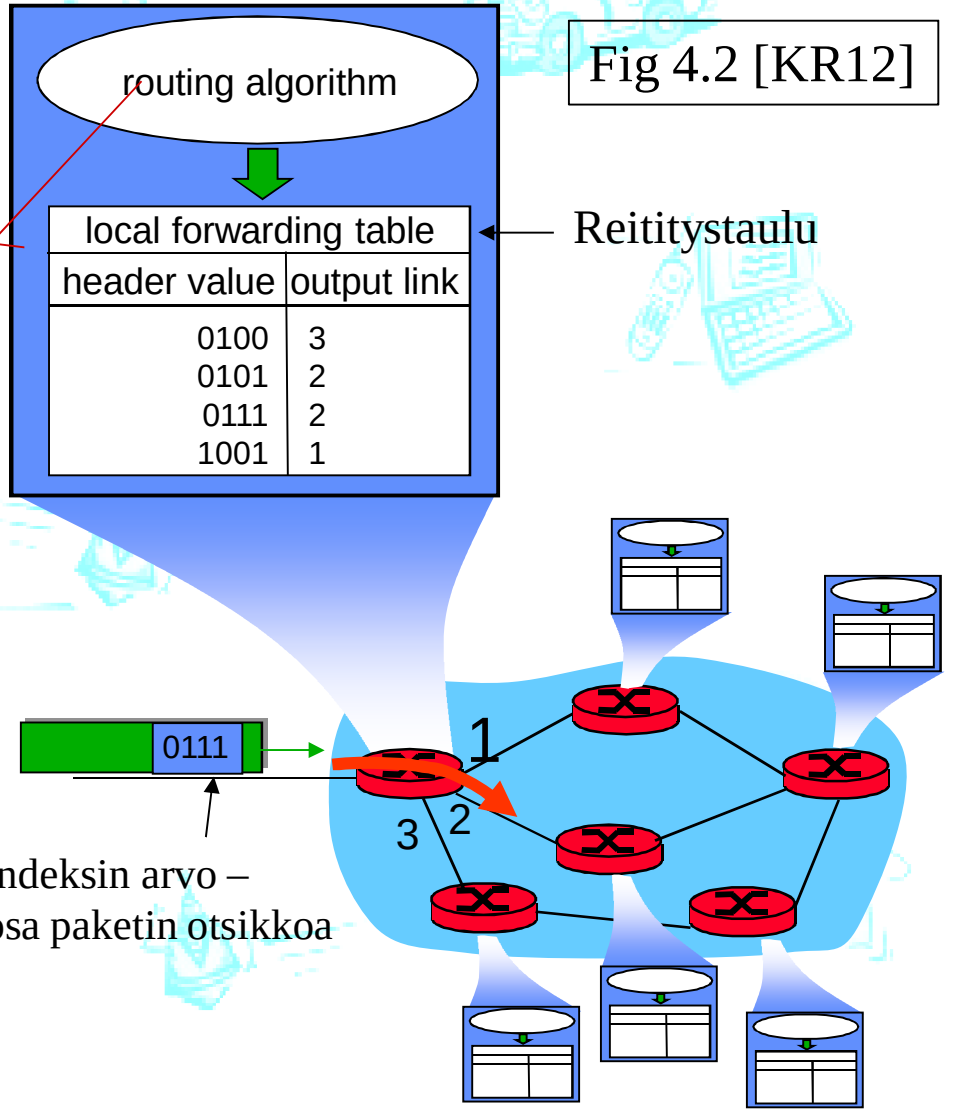


Fig 4.2 [KR12]

Reititystaulu

Reititystaulusta etsitään osumaa. Jos useita, niin käytetään sitä, jolla on pisin yhteinen osoitteen alkuosa.

Destination Address Range	Link interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

esimerkki:

DA: 11001000 00010111 00010110 10100001 Mihin linkkiin?

DA: 11001000 00010111 00011000 10101010 Mihin linkkiin?

Reititysalgoritmeja

Reititysalgoritmi, joka tarvitsee täydellisen tiedon verkosta

- Ennen laskentaa käytössä koko kuva verkosta:
  - Kaikki linkkiyhteydet solmujen välillä ja niiden kustannukset
    - Käytännössä vain tietystä autonomisesta alueesta
- Parhaat reitit lasketaan joko keskitetysti tai hajautetusti
- **Linkkitila-algoritmi** (link-state algorithm)

Reititysalgoritmi, jolle riittää epätäydellinen kuva verkosta

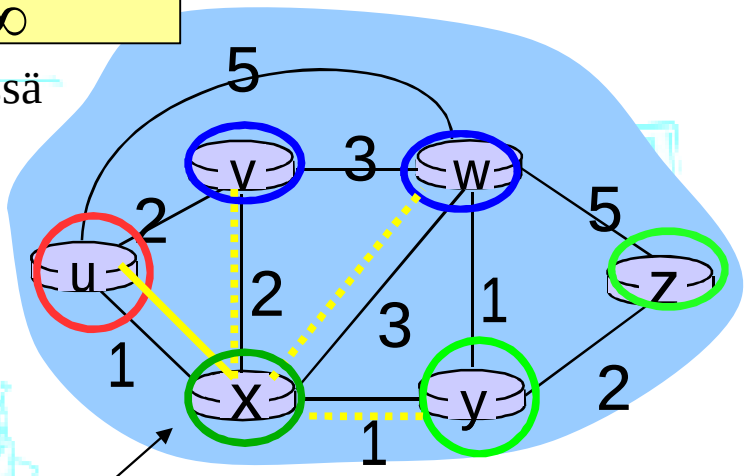
- Aluksi reititin tietää vain niistä koneista, joihin itse on yhdistetty
- Iteratiivinen algoritmi: reititin vaihtaa tietoja naapuriensa kanssa ja saa tietoa muusta verkosta
- **Etäisyysvektorialgoritmi** (distance vector algorithm)

Dijkstran algoritmi

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

1 ***Initialization:***
2 N' = **{u}**
3 for all nodes a
4 if a **adjacent** to u
5 then **D(a) = c(u,a)**
6 else **D(a) = ∞**
7
8 ***Loop***
9 find b not in N' such that D(b) is a minimum
10 add b to N'
11 update D(k) for all k adjacent to b and not in N' :
12 D(k) = min(D(k), D(b) + c(b,k))
13 /* new cost to k is either old cost to k or known
14 shortest path cost to b plus cost from b to k */
15 ***until all nodes in N'***

1. Eli jos u:n vieressä



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

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

Etäisyysvektorireititys

Alussa kukin solmu tuntee vain etäisyydet naapureihinsa itsensä kautta:

x:

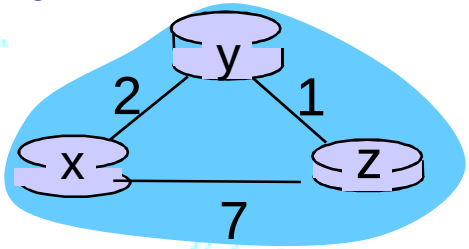
from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

y:

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

z:

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

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

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

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

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

x:

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

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

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

y:

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

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

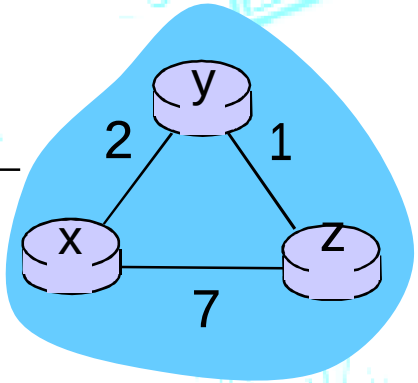
from	cost to x	cost to y	cost to z
x	0	2	3
y	2	0	1
z	3	1	0

z:

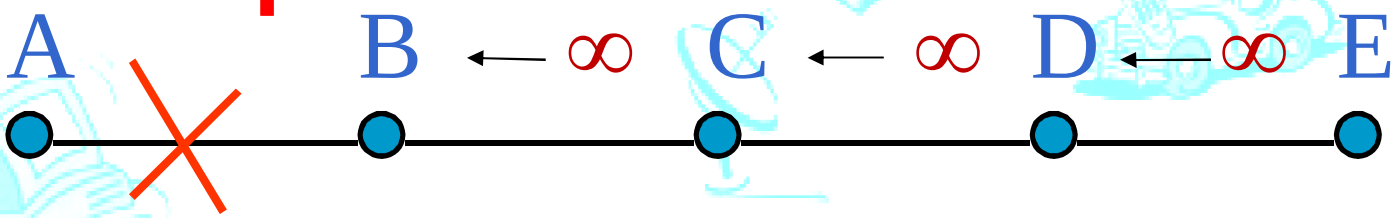
from	cost to x	cost to y	cost to z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

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

from	cost to x	cost to y	cost to 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

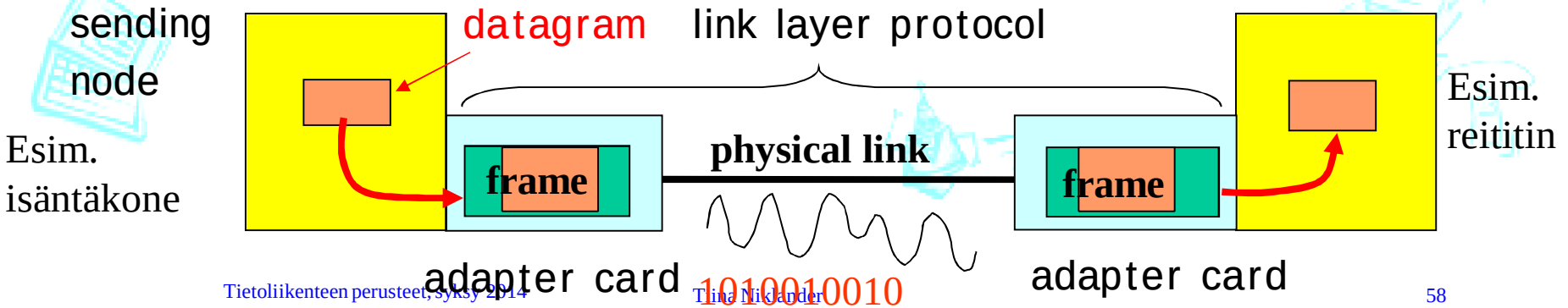
Etäisyys A:han

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

Linkkikerroksen tehtäviä

- **Vuonvalvonta, puskurointi**
  - Kytkimessä on useita erinopeuksisia linkkejä
- **Yksisuuntainen /kaksisuuntainen liikenne**
  - Yksisuuntainen: lähetysvuorojen hallinta
- **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

NIC (Network Interface Card) linkki- ja fyysinen kerros



Linkkikerroksen tehtäviä

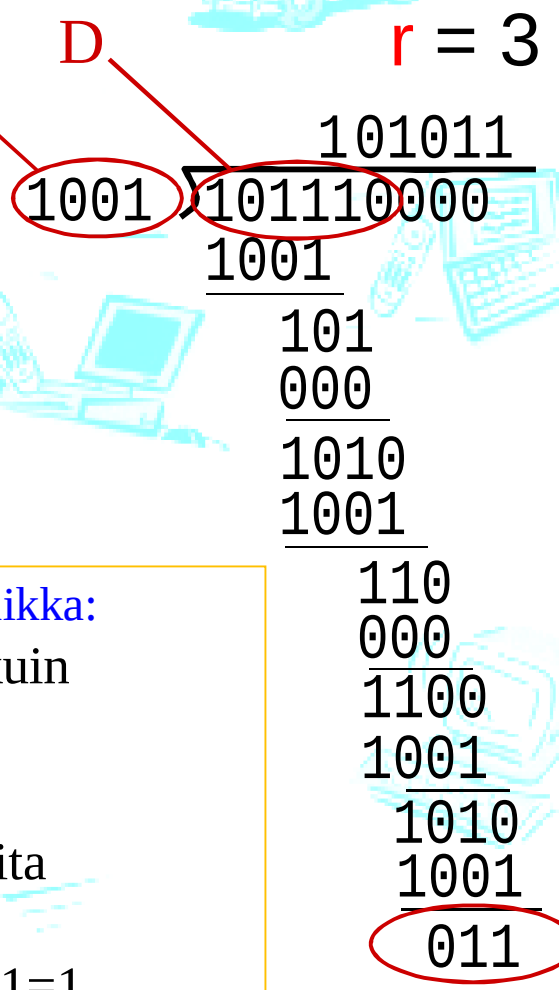
- **Kehystys** (framing)
  - Kehyksen rakenne ja koko riippuu siitä, millainen linkki on kyseessä
  - Otsake, data, lopuke
- **Kohteen ja lähteen osoittaminen**
  - Yhteiseen linkkiin voi olla liitettynä useita laitteita
  - Käytössä laitetason MAC-osoite (Medium access control)
- **Yhteisen linkin varaus 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 ..

CRC esimerkki

G pituus 4b

Fig 5.7 [KR12]



Tavoite (virheetön kun):

$D \cdot 2^r$ XOR R = nG

Ekvivalenttimuoto:

$D \cdot 2^r$ = nG XOR R

Edelleen sama:

Kun $D \cdot 2^r$ jaetaan G:llä, täytyy jäännöksen R olla:

$$R = remainder\left[\frac{D \cdot 2^r}{G}\right]$$

Modulo 2-aritmetiikka:
vähennyslasku kuin yhteenlasku
ei lainaamista,
ei muistinumeroita
= bittitason XOR
1+1=0, 1+0=0+1=1, 0+0=0

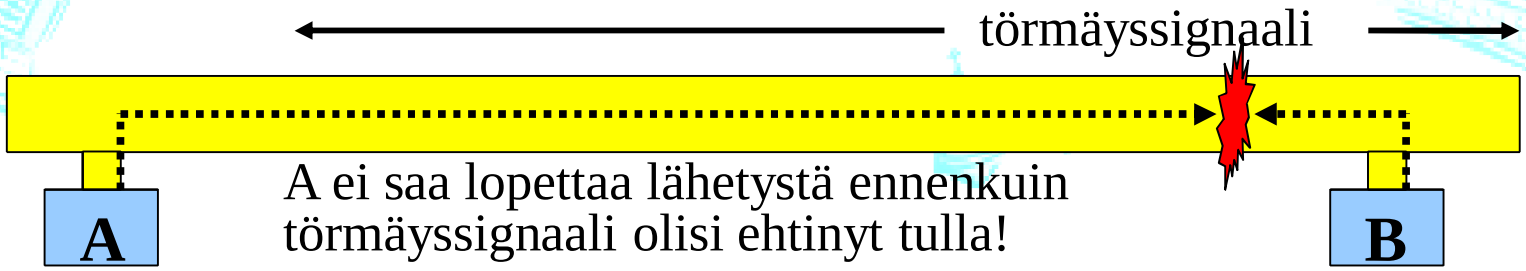
Lähetysvuorojen jakelu

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:
   - vuorokysely (polling), vuoromerkki, ...
   - “Minä ensin, sinä sitten.”

CSMA/CD

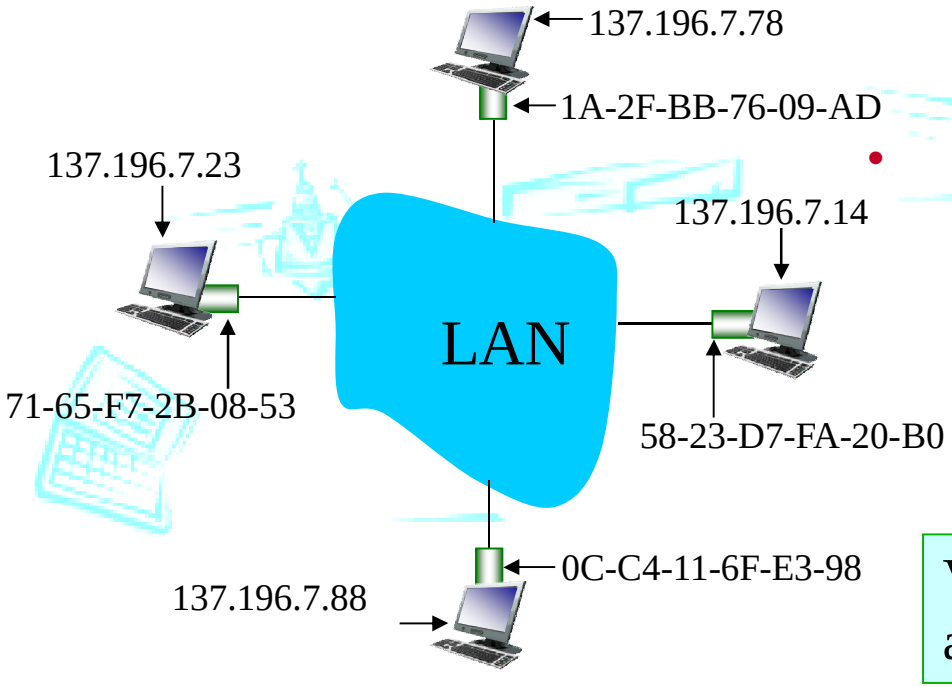
(Carrier Sense Multiple Access with Collision Detection)

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



Koneen MAC-osoitteen selvittäminen: ARP: address resolution protocol

Miten selvittää koneen MAC-osoite, kun tiedetään IP-osoite?



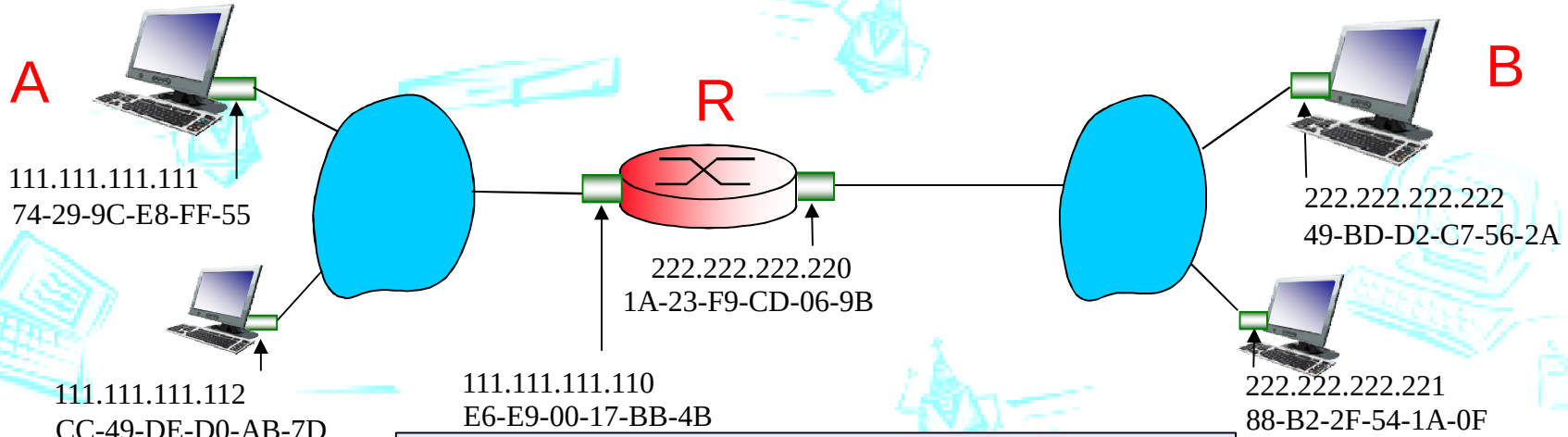
- *ARP-taulu:* jokaisella koneella (isäntä, reititin) oma taulu kullekin aliverkolle
  - IP/MAC osoitteen muunnos saman aliverkon koneille:
  - < IP-osoite; MAC-osoite; TTL>
  - TTL (Time To Live): Voimassaoloaika (tyypillisesti 20 min), ajan kuluttua muunnoksen voi unohtaa

Vrt: sovelluskerroksen **DNS**, jonka avulla voi selvittää IP-osoitteen nimelle

Lähettäminen toiseen verkkoon

Kts Fig 5.19 [KR12]

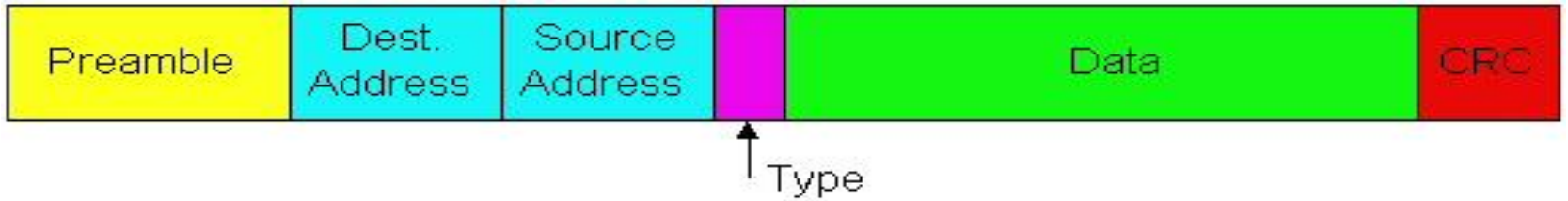
- Ensin omalle reitittimelle sen MAC-osoitteella ja reititin ohjaa eteenpäin
  - Reititystaulussa on verkko-osoite, jonne paketti seuraavaksi ohjattava
  - Katso kohdeverkon ARP-taulusta kohteen MAC-osoite
  - Jos ei ole taulussa, tee ARP-kysely kohdeverkon koneille



Reitittimellä on useita ARP-tauluja.

Ethernet-kehys

Fig 5.20 [KR12]



- Tahdistuskuvio (preample) (8 B)
  - 7 ensimmäistä tavua: 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

Kytkentätaulu (switching table):
-tietojen keruu saapuvista kehyksistä

- 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
- Kohde Y ja lähde X taulussa:
  - X ja Y samassa portissa => hylkää kehys **(on jo oikeassa verkon osassa)**
  - X ja Y eri porteissa => lähetä kehys Y:n porttiin

Luento 10

Skenaario: opiskelija yhdistää koneensa langattomaan verkkoon ja pyytää sivua www.google.com

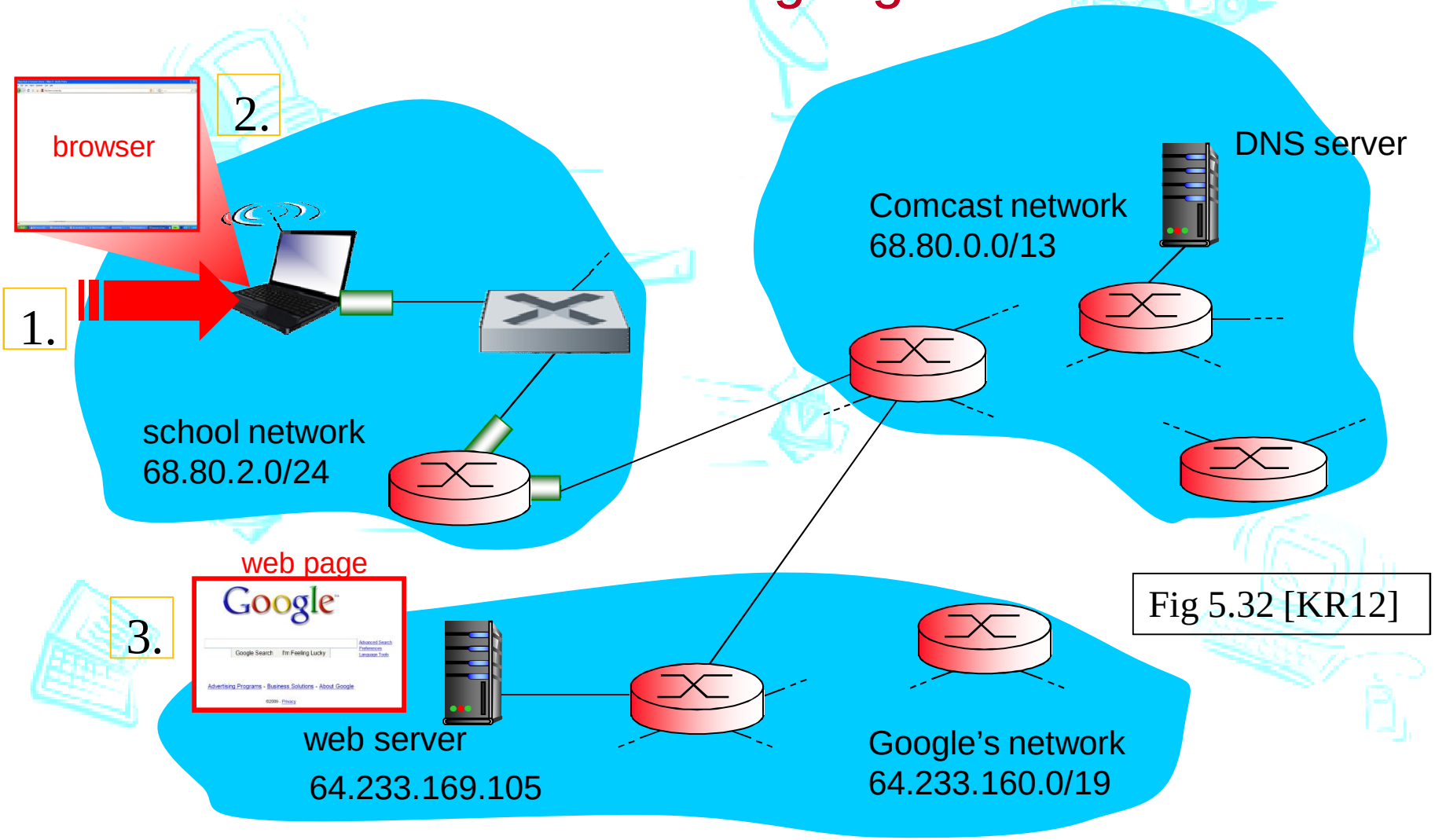
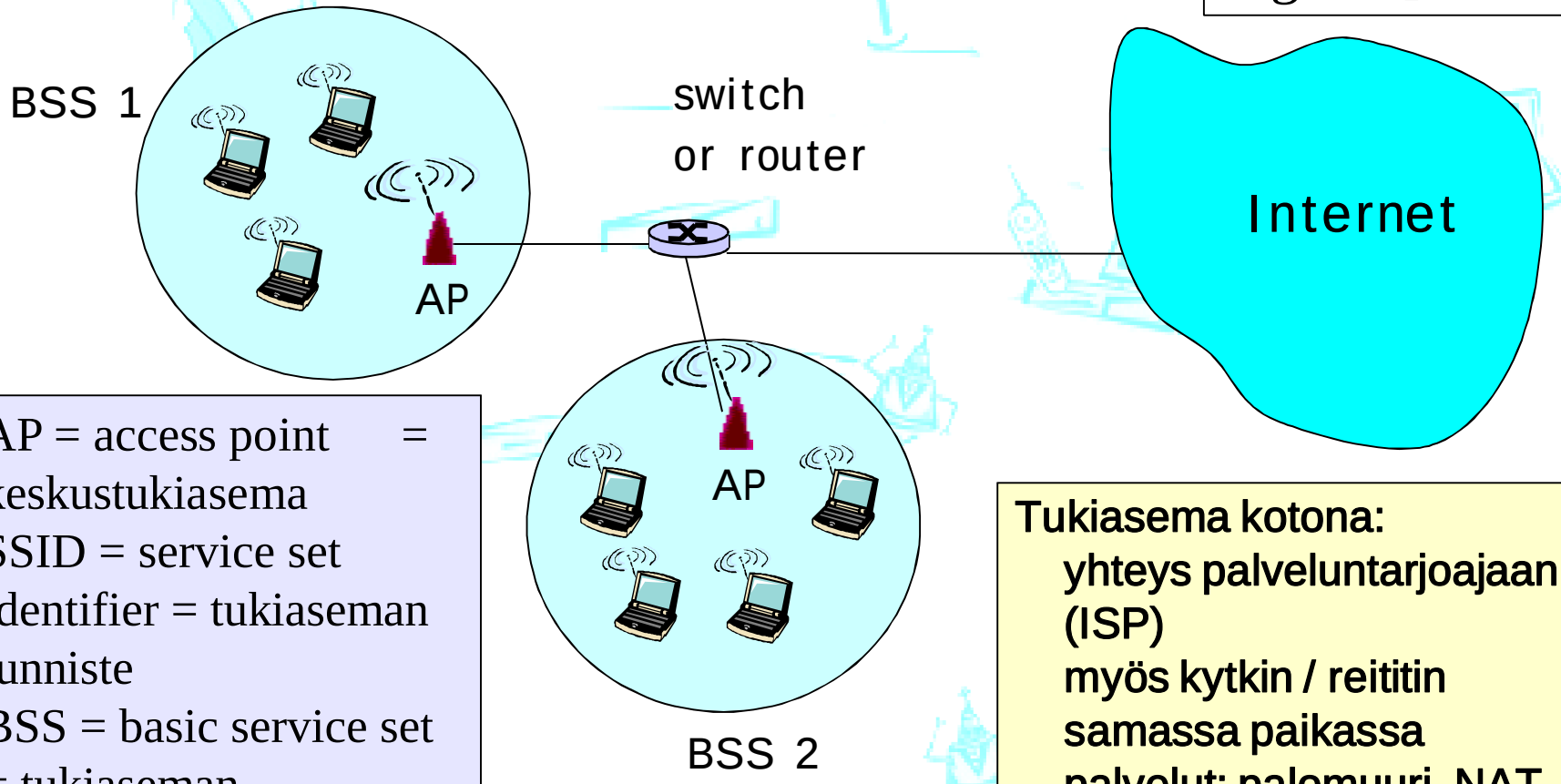


Fig 5.32 [KR12]

IEEE 802.11 -lähiverkko

(infrastructure wireless LAN, Wi-Fi)

Fig 6.7 [KR12]



AP = access point = keskustukiasema
SSID = service set identifier = tukiaseman tunniste
BSS = basic service set = tukiaseman palvelemat koneet

Tukiasema kotona:
- **yhteys palveluntarjoajaan (ISP)**
- **myös kytkin / reititin samassa paikassa**
- **palvelut: palomuuri, NAT, DHCP**

802.11: CSMA/CA

Fig 6.10 [KR12]

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)

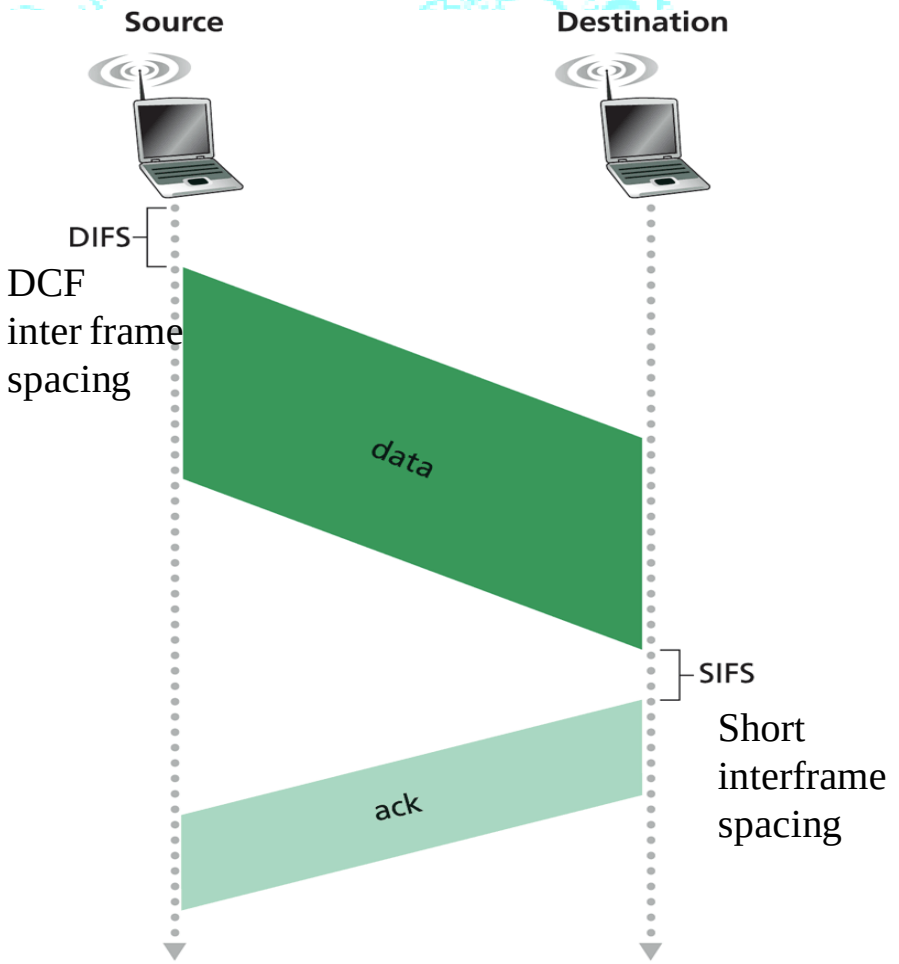


Figure 6.8 ♦ 802.11 uses link-layer acknowledgments

802.11: Kehyksen rakenne

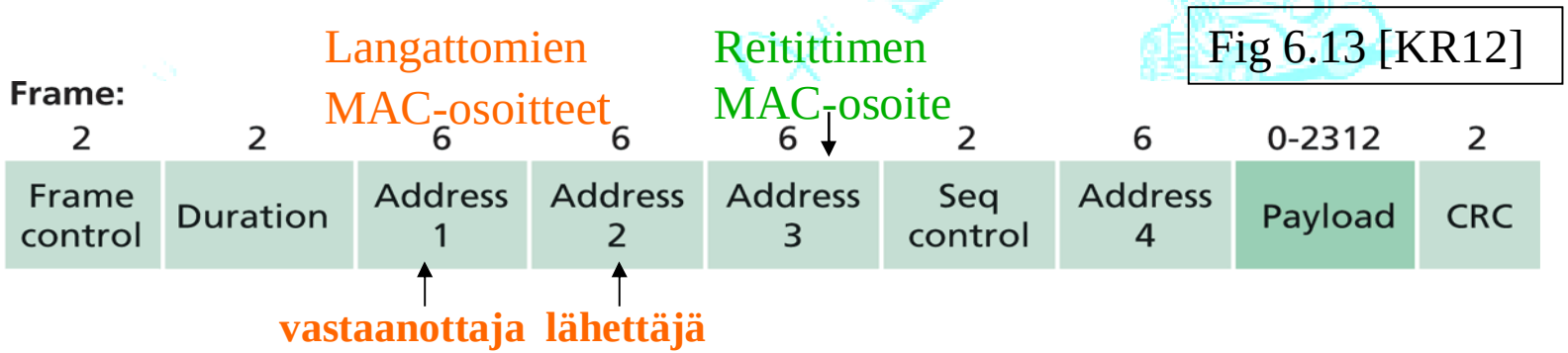


Fig 6.13 [KR12]

- HUOM: **4** osoitekenttää
  - isännän ja tukiaseman MAC-osoitteet (kenttä 1 ja 2)
  - Sen reitittimen osoite, jossa tukiasema on kiinni (kenttä 3)
    - Reitittimen ja tukiaseman välillä tavallinen kehys (esim. Ethernet)
    - Tukiasema on 'näkymätön' reitittimelle, reititin luulee saavansa
    - kehyksen suoraan isäntäkoneelta
  - Kenttä 4 käytössä vain ad hoc -verkossa
- Lähetyksen kesto (duration)
  - Jos RTS/CTS kehys, varauksen kesto (lähetys-kuittaus)
- Seq control - järjestysnumeroa tarvitaan kuittauksia varten

Request To Send
Clear To Send

802.11: Kehyksen rakenne

Frame control

Type, Suptype - miten kehystä tulkittava: RTS/CTS/ACK/ data?

ToAP ja FromAP – osoitekenttien tulkinta: lähettäjä/vastaanottaja/adhoc?

WEP (Wired Equivalent Privacy) ja WPA (WiFi Protected Acces) - Käyttääkö salausta (Huom. WEPin tietoturva surkea → ÄLÄ KÄYTÄ)

.....

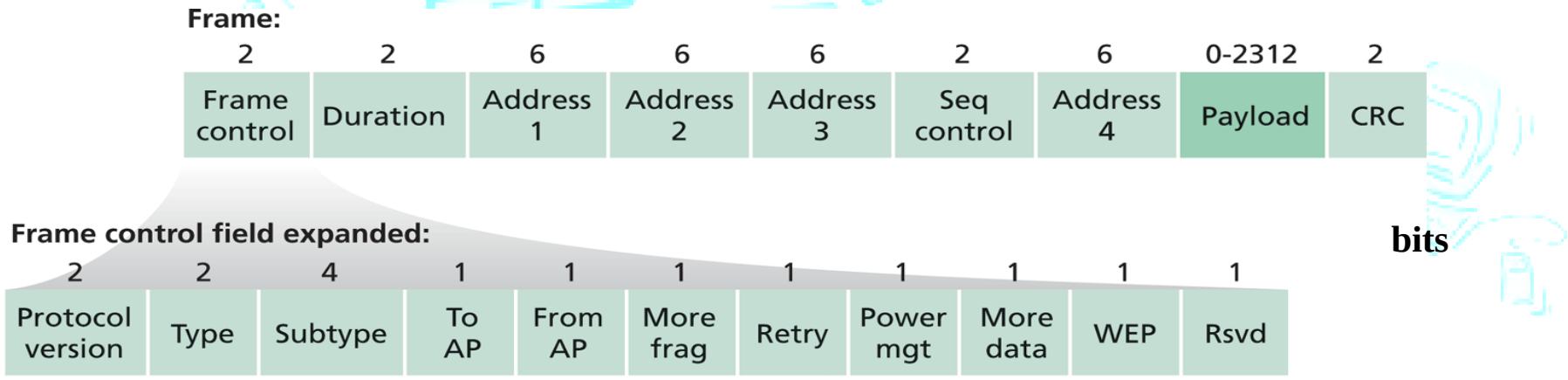


Figure 6.13 The 802.11 frame