



Luento 5: Kuljetuskerros

luotettavan tiedonsiirron periaatteet

Ma 11.11.2013

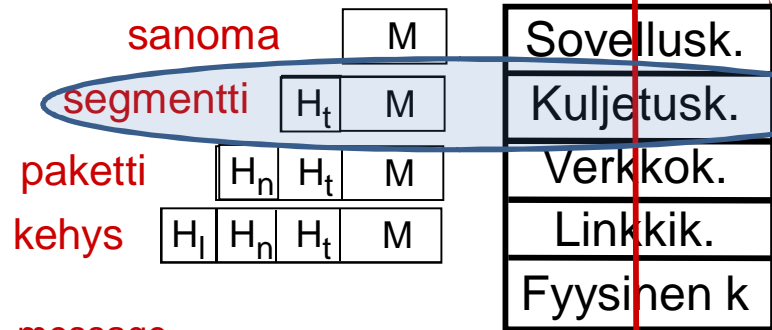
Tiina Niklander

Kurose&Ross
Ch3

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

Luennon sisältöä

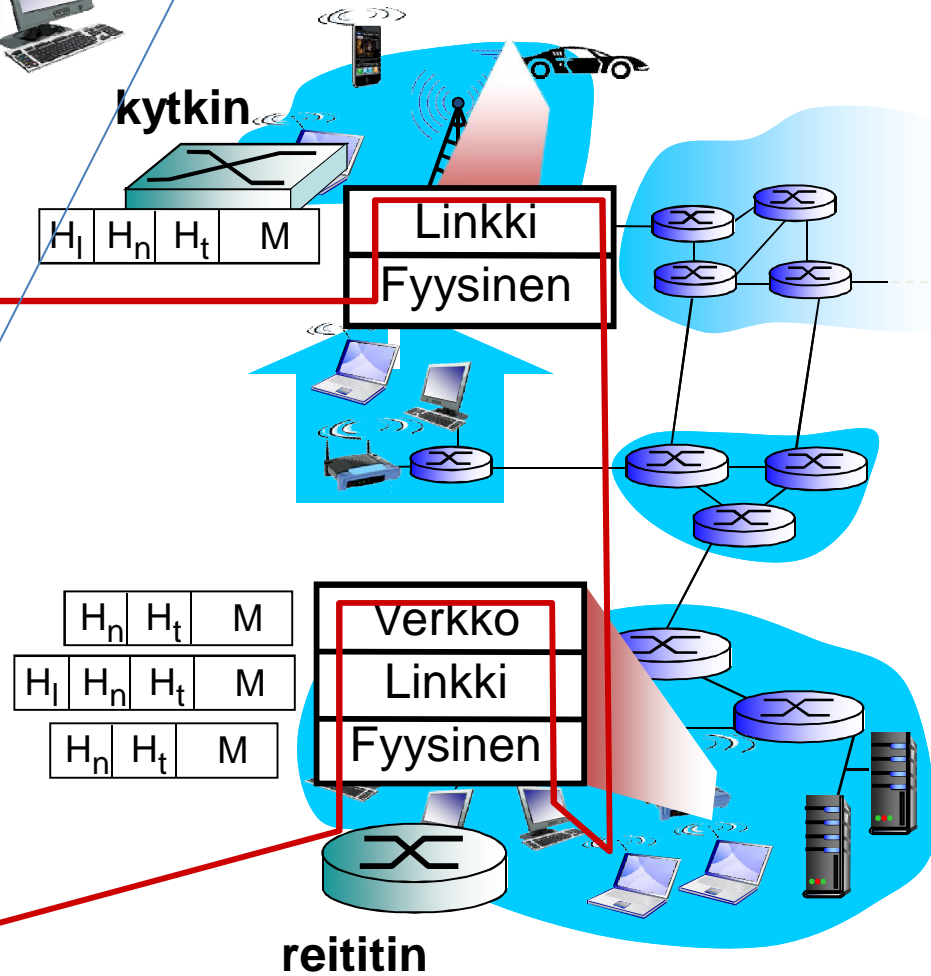
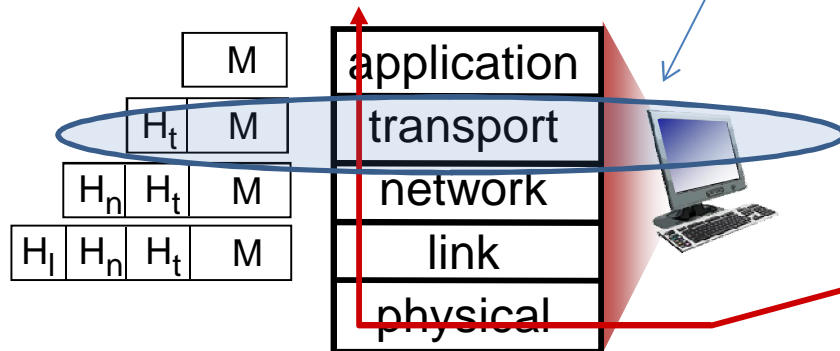
Lähettäjä (sender)



message,
segment
datagram
frame

Fig 1.24 [KR12]

Vastaanottaja (recipient)





Sisältöä

Kuljetuspalvelut

Luotettavan kuljetuspalvelun periaatteet

Yhteydetön kuljetuspalvelu, UDP

Yhteydellinen kuljetuspalvelu, TCP

Ruuhkanhallinta TCP:ssä

Oppimistavoitteet:

- Tuntea Internetin kuljetusprotokollien (UDP/TCP) toiminnallisuus ja periaatteet
- Osata luotettavan kuljetuspalvelun ja vuonvalvonnan periaatteet ja toteutukset
- Osata TCP-ruuhkanhallinnan





Kuljetuskerros

Kuljetuspalvelu



Kuljetuskerros

Tarjoaa kuljetuspalvelun
prosessien välille

Vain isäntäkoneissa

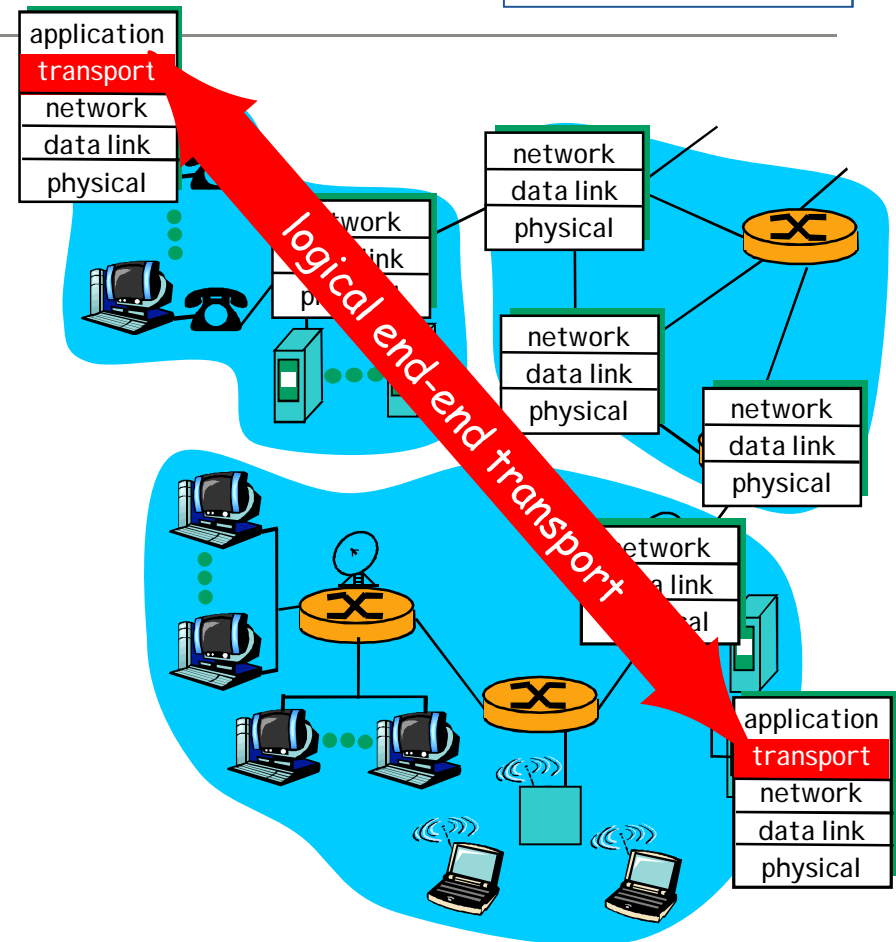
Lähetys: Pilko sovelluskerroksen
sanoma pienemmiksi **segmenteiksi**,
jotka verkkokerros toimittaa perille.

Vastaanotto: Kokoa segmentit
sanomaksi, jonka sovellus lukee.

Verkkokerros reitittää koneesta
koneelle

Segmentin koko s.e. verkkokerros
pystyisi välittämään sellaisenaan

Fig 3.1 [KR12]





Sovelluksen vaatimuksia

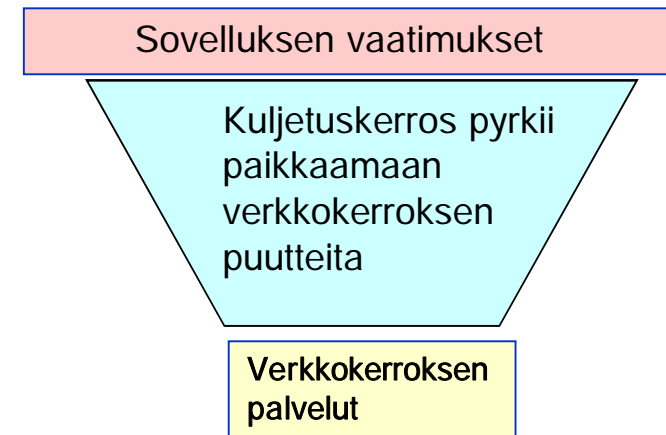
Verkkokerroksen palvelu voi

Epäluotettava!

- Muuttaa segmentin bittejä tai kadottaa segmenttejä
- Toimittaa segmentit epäjärjestyksessä kuljetuskerrokselle
- Viivyttaa segmenttejä satunnaisen pitkän ajan
- Luovuttaa kuljetuskerrokselle useita kopioita samasta segmentistä
- Rajoittaa segmentin kokoa

Sovellus edellyttää kuljetuspalvelulta

- Virheettömyyttä, luotettavuutta
- Järjestyksen säilymistä
- Kaksoiskappaleiden karsimista
- Mielivaltaisen pitkien sanomien sallimista
- Vuonvalvonnan mahdollistamista



Kuljetuskerros peittää verkkokerroksen puutteita ja parantaa sovelluksen näkemää palvelun laatua



Internetin kuljetusprotokollat

TCP: luotettava, järjestyksen säilyttävä tavujen kuljetuspalvelu

Virheenvalvonta (error control): Huomaa ja korjaa virheet, hylkää kaksoiskappaleet

Vuonvalvonta (flow control): Älä ylikuormita vastaanottajaa

Ruuhkanhallinta (congestion control): Älä ylikuormita verkkoa

Yhteyden muodostaminen ja purku

UDP: Ei-luotettava, ei-järjestyksen säilyttävä sanomien kuljetuspalvelu

Välittää vain sanomia, ei pyri mitenkään parantamaan verkkokerroksen tarjoamaa palvelun laatua

Luotettavuus jää sovelluskerroksen hoidettavaksi

Kumpikaan kuljetuspalvelu ei anna takuita viiveelle tai siirtonopeudelle (“best effort”)



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 tietyille palvelulle (well known ports)
 - Esim. www-palvelulle portti 80, SMTP-postipalvelulle portti 25



Segmenttien lomitusta (multiplexing) ja erottelu (demultiplexing)

Lomitus lähettäjällä:

Segmenttejä useasta pistokkeesta, liitä kuhunkin kuljetusotsake (käytetään erottelussa myöhemmin)

Erottelu vastaanottajalla:

Välitä saapuva segmentti oikeaan pistokkeeseen, tieto löytyy otsakkeesta

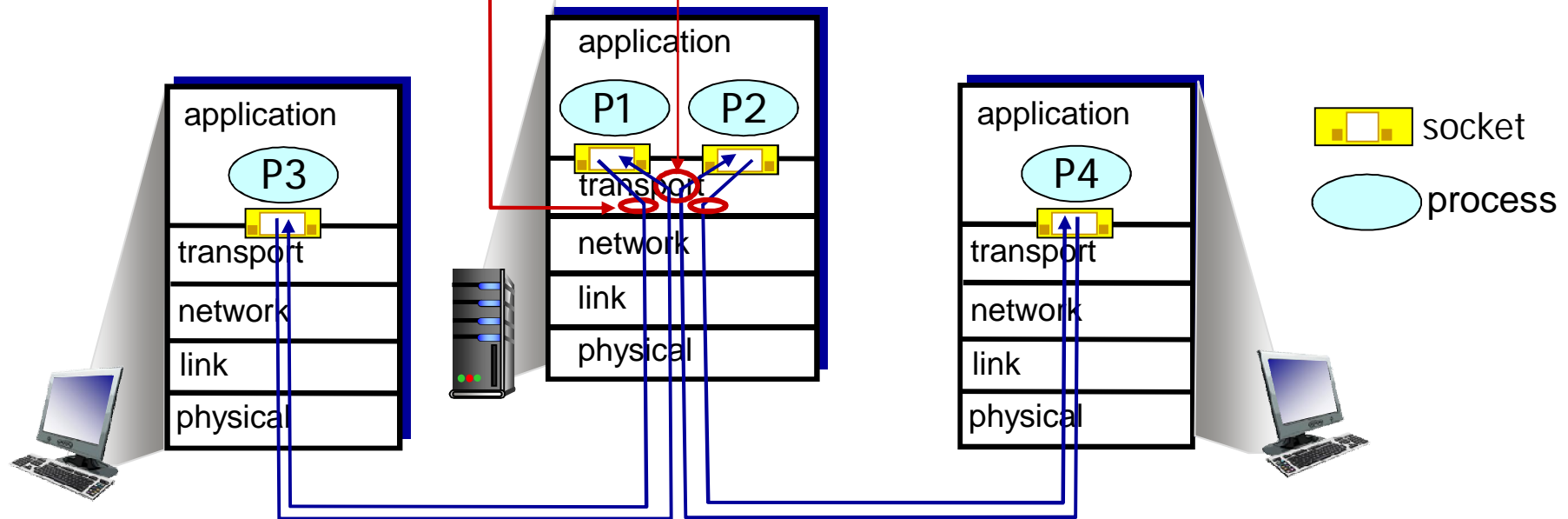


Fig 3.2 [KR12]



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

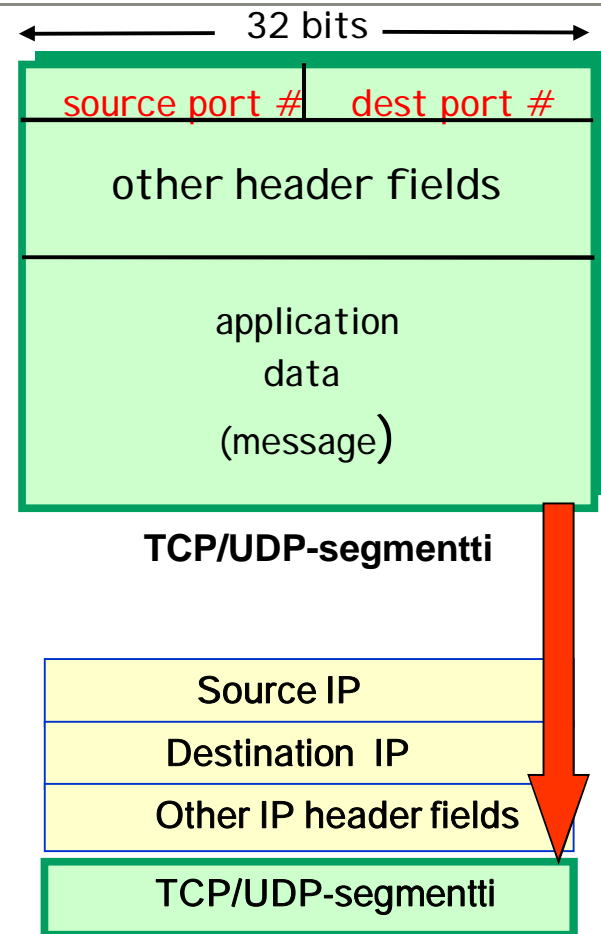
Verkkokerros

Paketin otsakkeessa lähde- ja kohdekoneen

IP-osoite,

reitittimet osaavat ohjata oikealle koneelle

Kts Fig 3.3 & 4.13 [KR12]





Mikä kone /Mikä prosessi?

Vastaanotto (palvelija)

Verkkokerros

Vastaanottaa IP-paketin
Poistaa verkkokerroksen otsaketiedot
Luovuttaa paketissa olleen segmentin
kuljetuskerrokselle

Kuljetuskerros

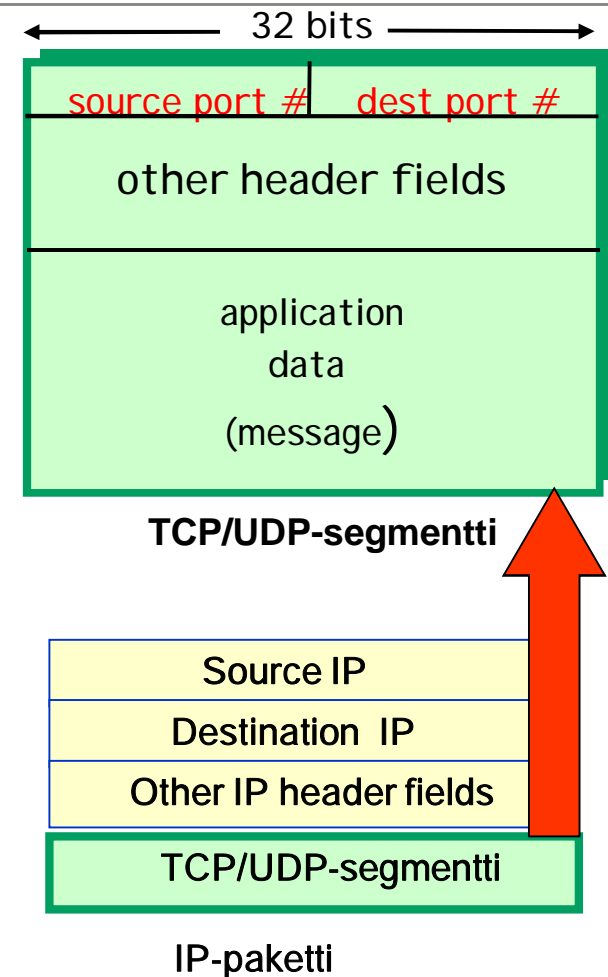
Poistaa kuljetuskerroksen otsaketiedot
Kokoaa yhteenkuuluvat segmentit
sanomiksi (tavuvirraksi)
Ohjaa sanoman (tavuvirran) oikealle
prosessille (eli oikeaan pistokkeseen)
porttinumeron avulla

– TCP: huolehtii myös luotettavuudesta

– UDP: tarjoaa pelkän välityspalvelun

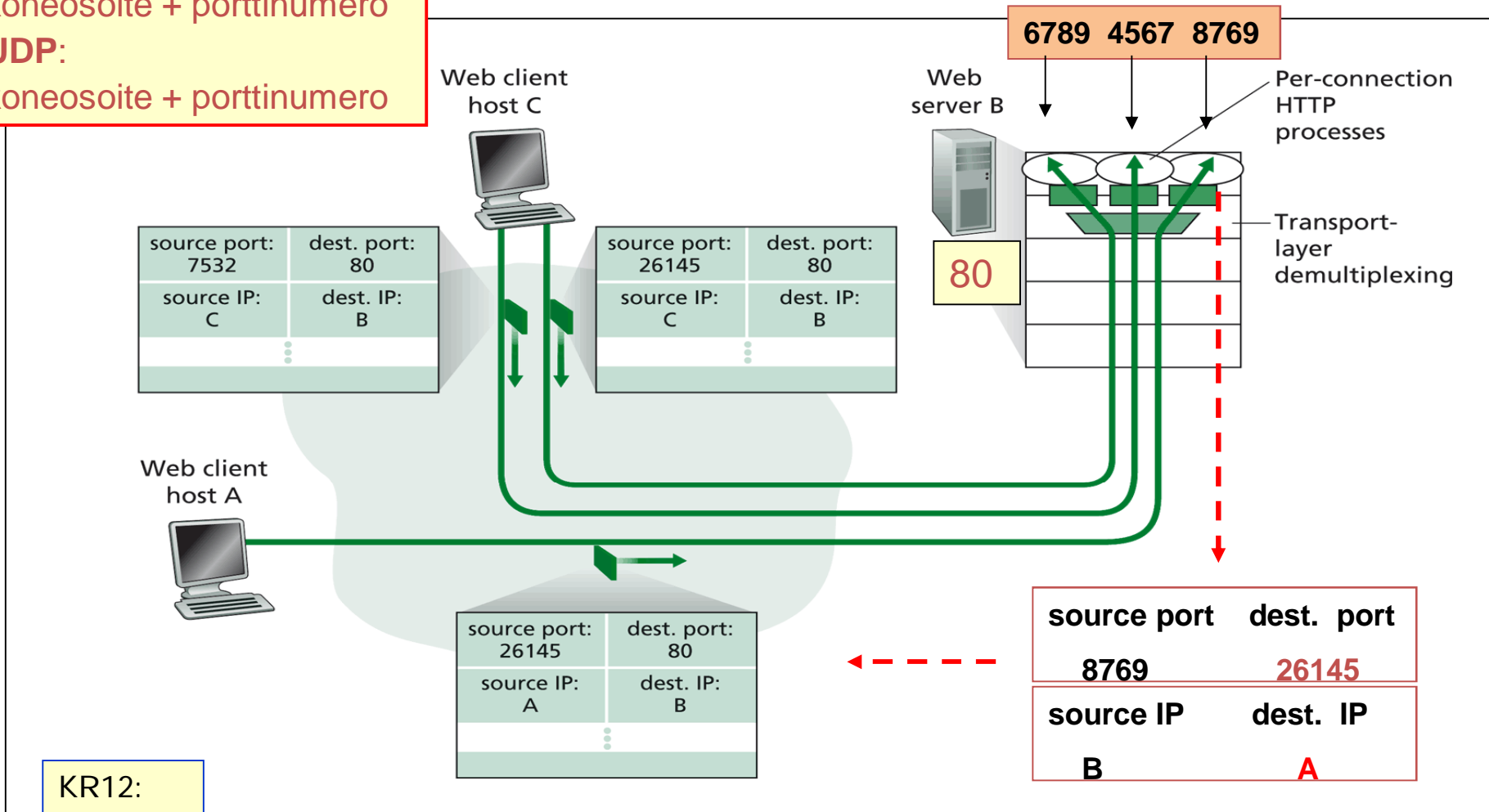
Tietoliikenteen perusteet, Tiina Niklander 2013

Kts Fig 3.3 & 4.13 [KR12]



Kaksi www-asiakasta ja palvelija

TCP-yhteys:
koneosoite + porttinumero
UDP:
koneosoite + porttinumero



KR12:

Figure 3.5 ♦ Two clients, using the same destination port number (80) to communicate with the same Web server application

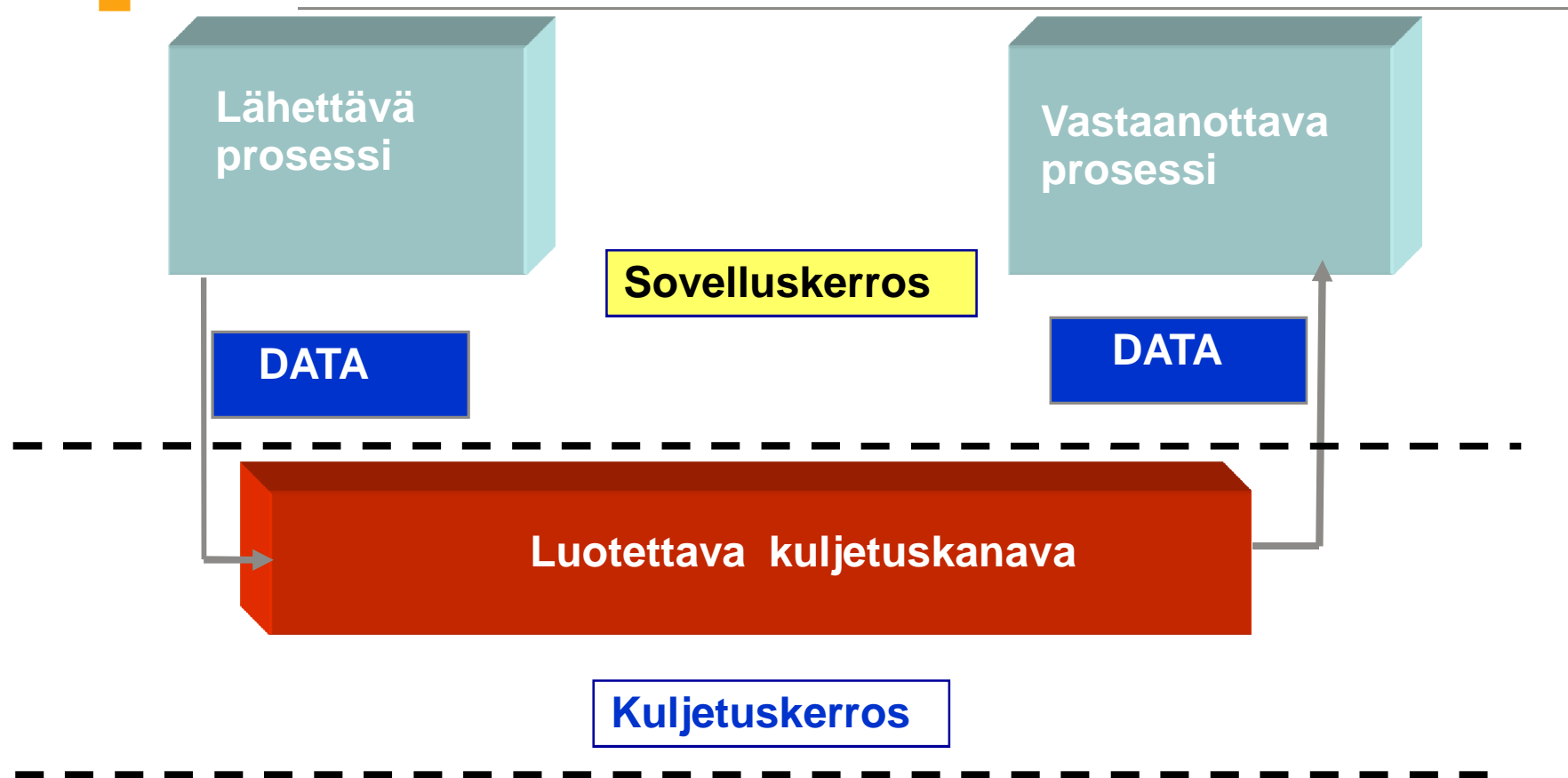


Kuljetuskerros

Luotettavan kuljetuspalvelun periaatteet

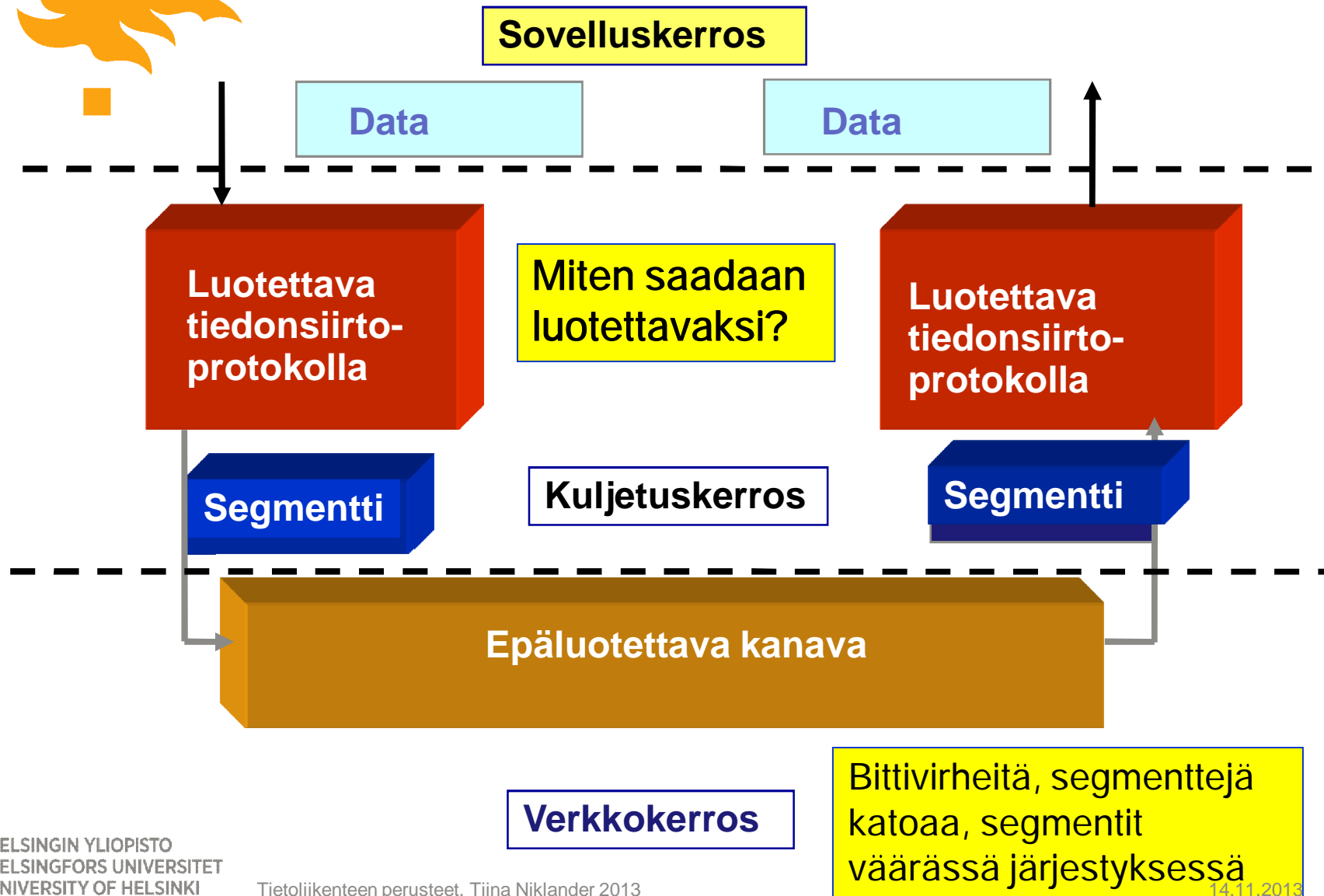


Luotettava tiedonsiirto?





Luotettava tiedonsiirto?





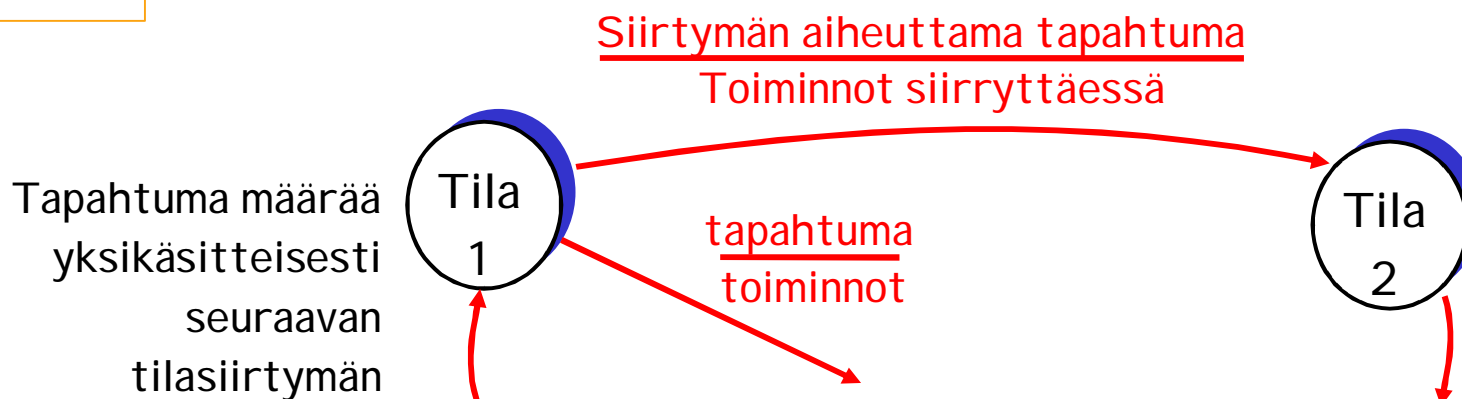
Kuinka saada luotettavaksi?

Tarkastellaan yleisesti luotettavan tiedonsiirron ongelmia ja erilaisia ratkaisuyrityksiä

Edeten ideaalitalanteesta yhä ongelmallisempaan

Käyttäen **äärellisiä tila-automaatteja** lähettäjän ja vastaanottajan toiminnan kuvaamiseksi

LaMa:n termi:
Tilasiirtymä-
järjestelmä





Ideaalitalanne (\rightarrow rtd1.0)

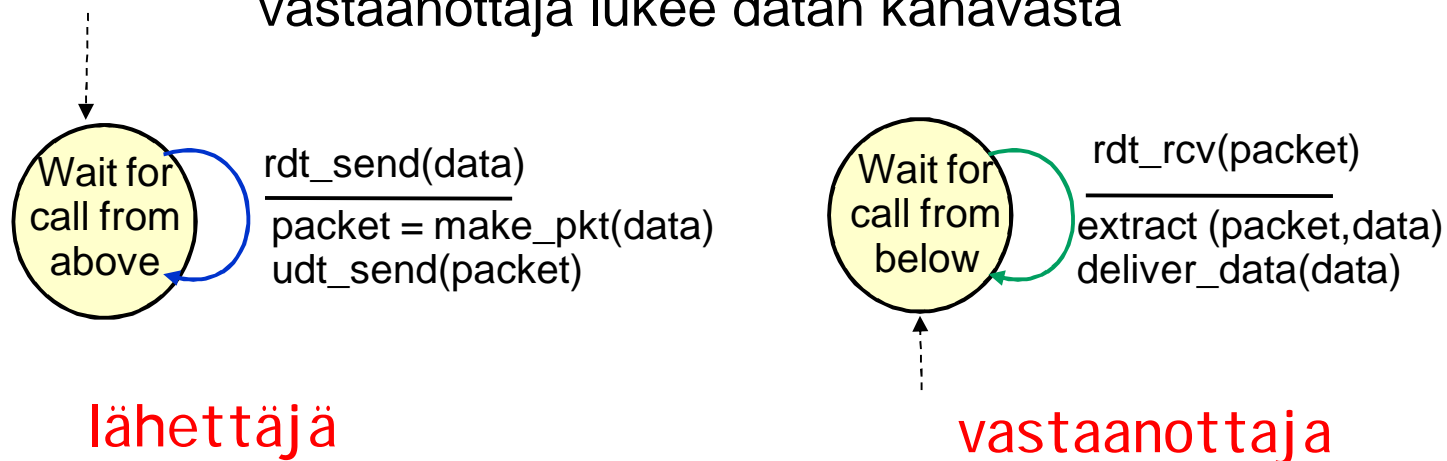
Oletus: **siirtokanava on täysin luotettava**

Ei bittivirheitä, ei katoavia paketteja, ei epäjärjestystä

Luotettava kuljetuspalvelu =

Lähettäjä kirjoittaa datan kanavaan,

Vastaanottaja lukee datan kanavasta

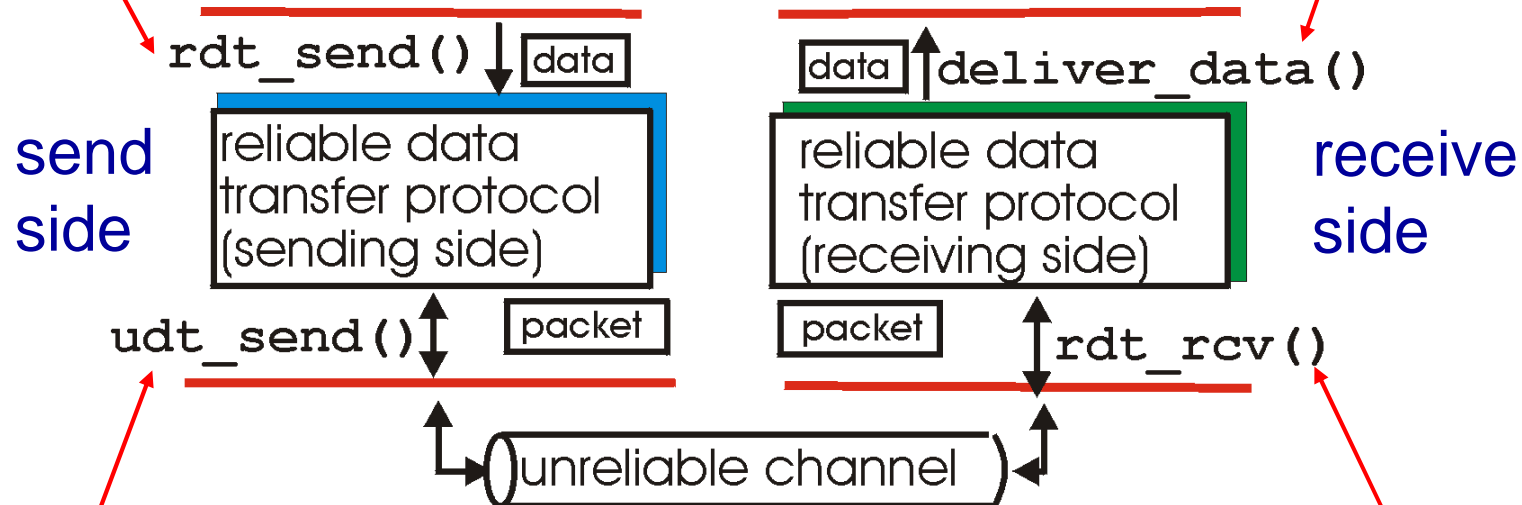


Tila-automaattien tapahtumia (kalvot ja kirja)

Kts. Fig 3.8 [KR12]

rdt_send() : kutsutaan yläpuolelta (esim. Sovellus). Kutsussa tulee välitettävä data.

deliver_data() : rdt kutsuu. Kutsussa välitetään data yläpuolelle.



udt_send() : rdt kutsuu. Kutsussa paketti epäluotettavan kanavan kautta vastaanottajalle

rdt_rcv() : kutsutaan, kun paketti saapuu kanavasta vastaanottajalle



Vain bittivirheitä (→ rdt2.0)

- Oletus: **Voi olla vain bittivirheitä**

- Bitti voi kääntyä siirron aikana
- Siirto ei kadota paketteja

ACK – Positive acknowledgment
NAK – Negative acknowledgment

- Kuinka toipua?

- Kuittaukset: vastaanottaja kuittaa ACK:lla virheettömän paketin,
- NAK-kuittaus, jos paketti on virheellinen + hylkää
- Jos NAK, niin lähettäjä lähettää paketin uudelleen

- Luotettava kuljetuspalvelu

- Virheen huomaaminen: tarkistussumma
- Palaute vastaanottajalta: **kuittausanoma (ACK / NAK)**
- Uudelleenlähetys: dataa puskuroitava

- **Stop-and-wait**-protokolla

- Lähettäjä odottaa kuittausta ennenkuin lähettää seuraavan

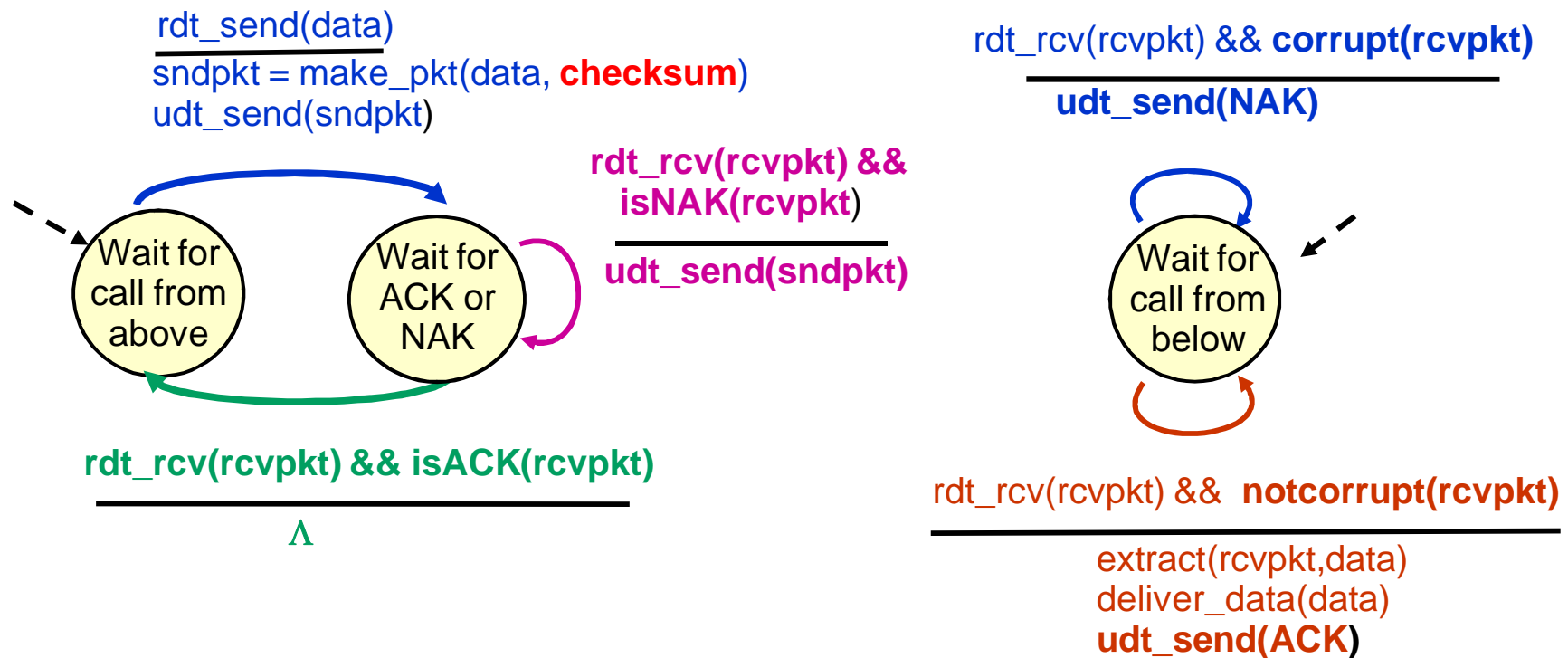


Rtd 2.0: Vain bittivirheitä

Fig 3.10 [KR12]

sender

receiver



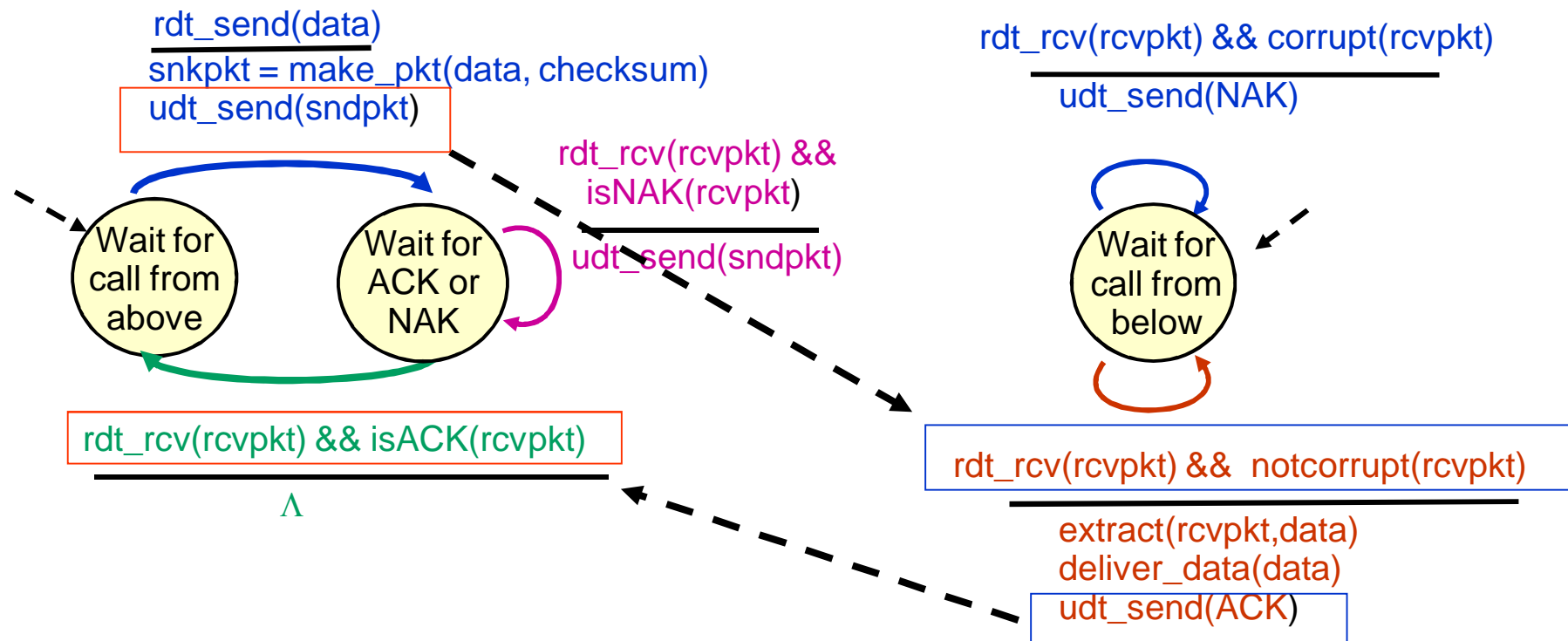


rdt2.0: Toiminta, kun ei ole virhettä

Fig 3.10 [KR12]

sender

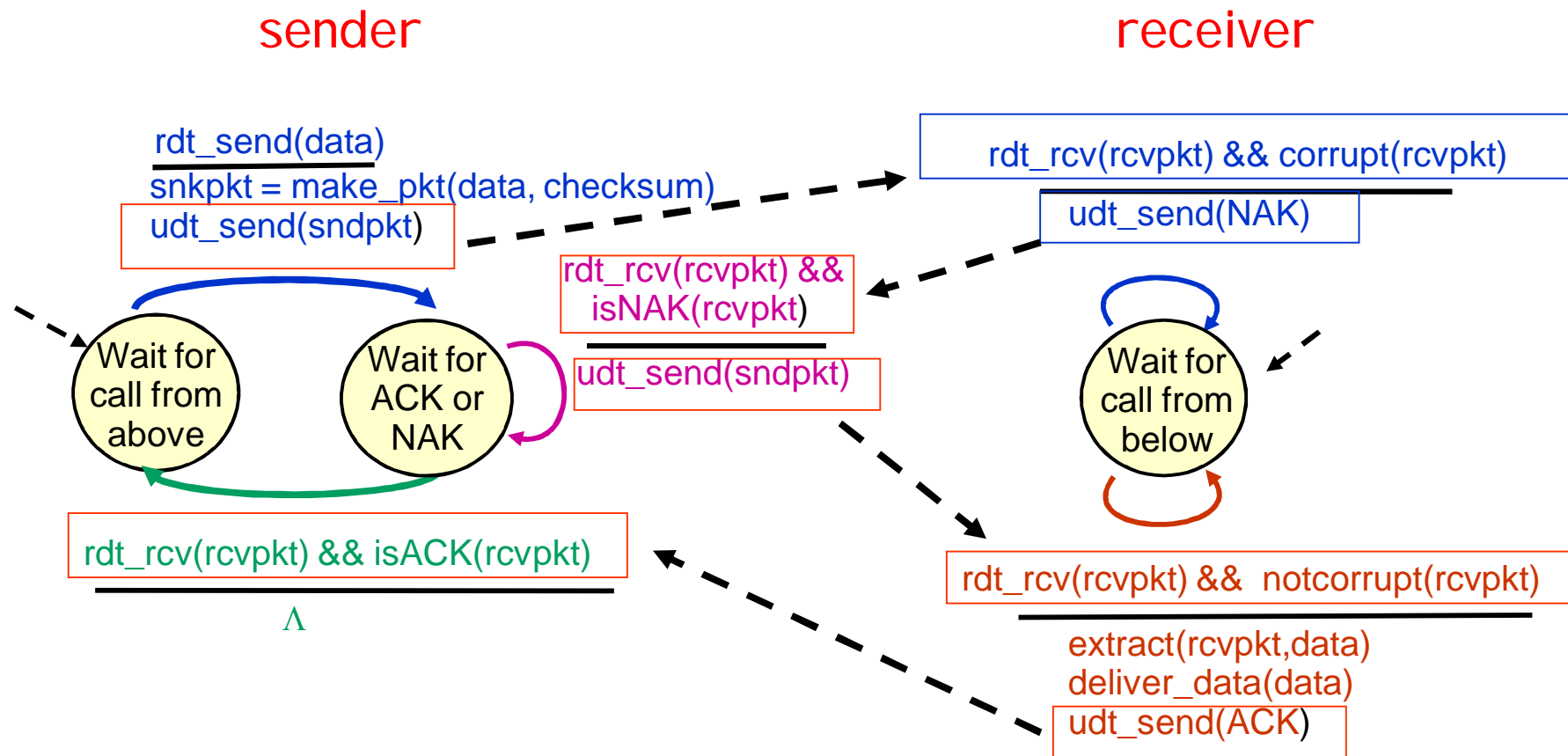
receiver





rdt2.0: Toiminta virhetilanteessa

Fig 3.10 [KR12]





rdt2.0: Missä voi mennä väärin?

Ei toimi, jos ACK/NAK -paketeissa bittivirheitä!

Onko OK vai ei?

Korjaus: ACK/NAK-paketteihin tarkistusbitit, jotta virhe huomataan

Entä toipuminen?

Jos kuittauksessa virhe, lähetetään paketti uudelleen

Uudelleenlähetys voi tuottaa kaksoiskappaleen (duplicate), joka on huomattava ja hylättävä

=> **Paketteihin järjestysnumero**

Kaksoiskappale: jos sama numero

Vastaanottaja kuittaa normaalisti, mutta ei anna sovellukselle



Versio rdt2.1: enemmän bittivirheitä

Lähettäjä:

Lisää pakettiin järjestysnumeron
(numerot 0 ja 1 riittävät tässä
protokollassa. Miksi?)

Ns. **vuorotteleva bitti**

Tarkista, että ACK/NAK ei ole
korruptoitunut

Tilakaaviossa nyt kaksinkertaisesti tiloja

Kaavion tilan 'muistettava' paketin
numero

Säilytä kopio lähetetystä paketista

Vastaanottaja:

Tarkista, ettei ole
kaksoiskappale

Kaavion tilan 'muistettava'
seuraavan paketin
numero: 0 vai 1

Myös kaksoiskappale
kuitattava, koska ei tiedä
menikö edellinen kuittaus
perille



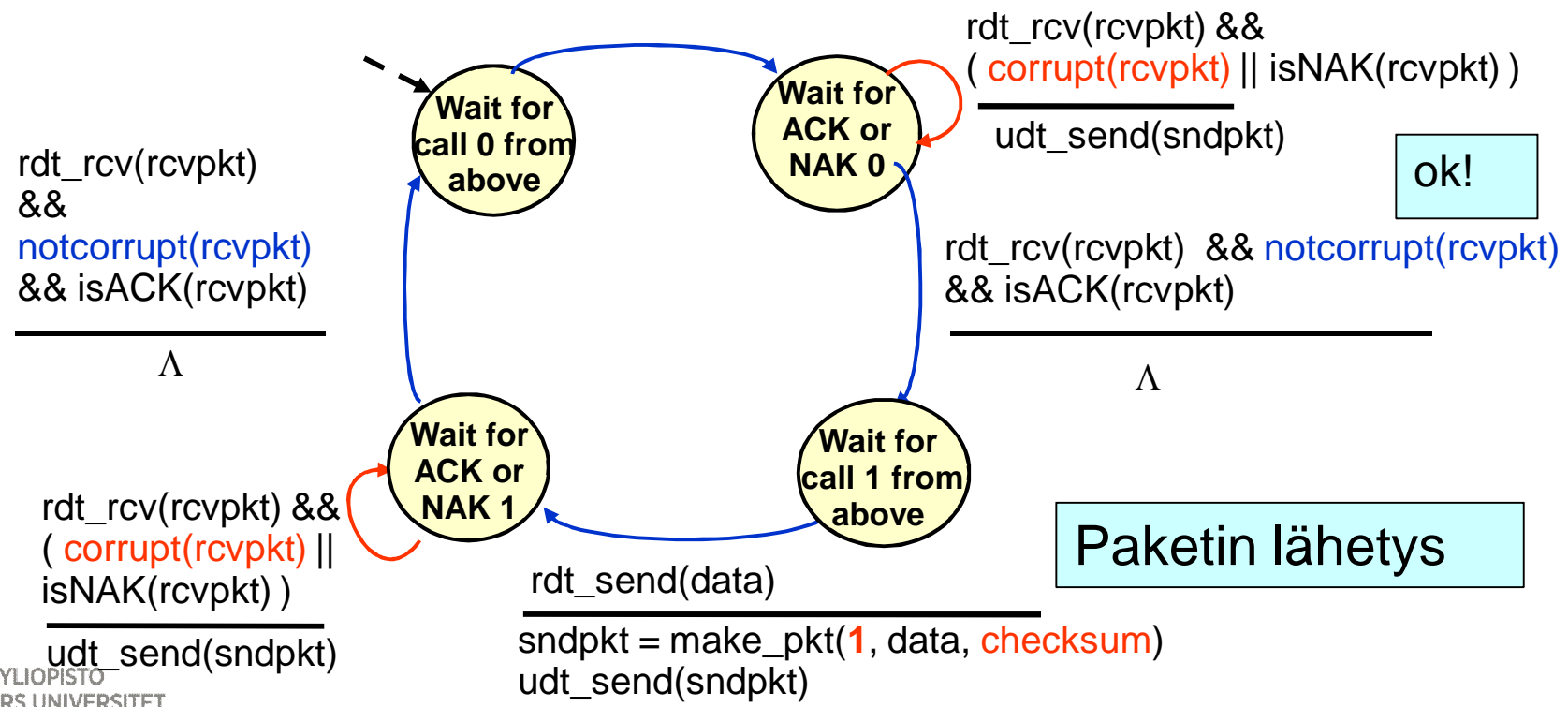
rdt2.1: Lähettäjän tila-automaatti

Fig 3.11 [KR12]

Paketin lähetys

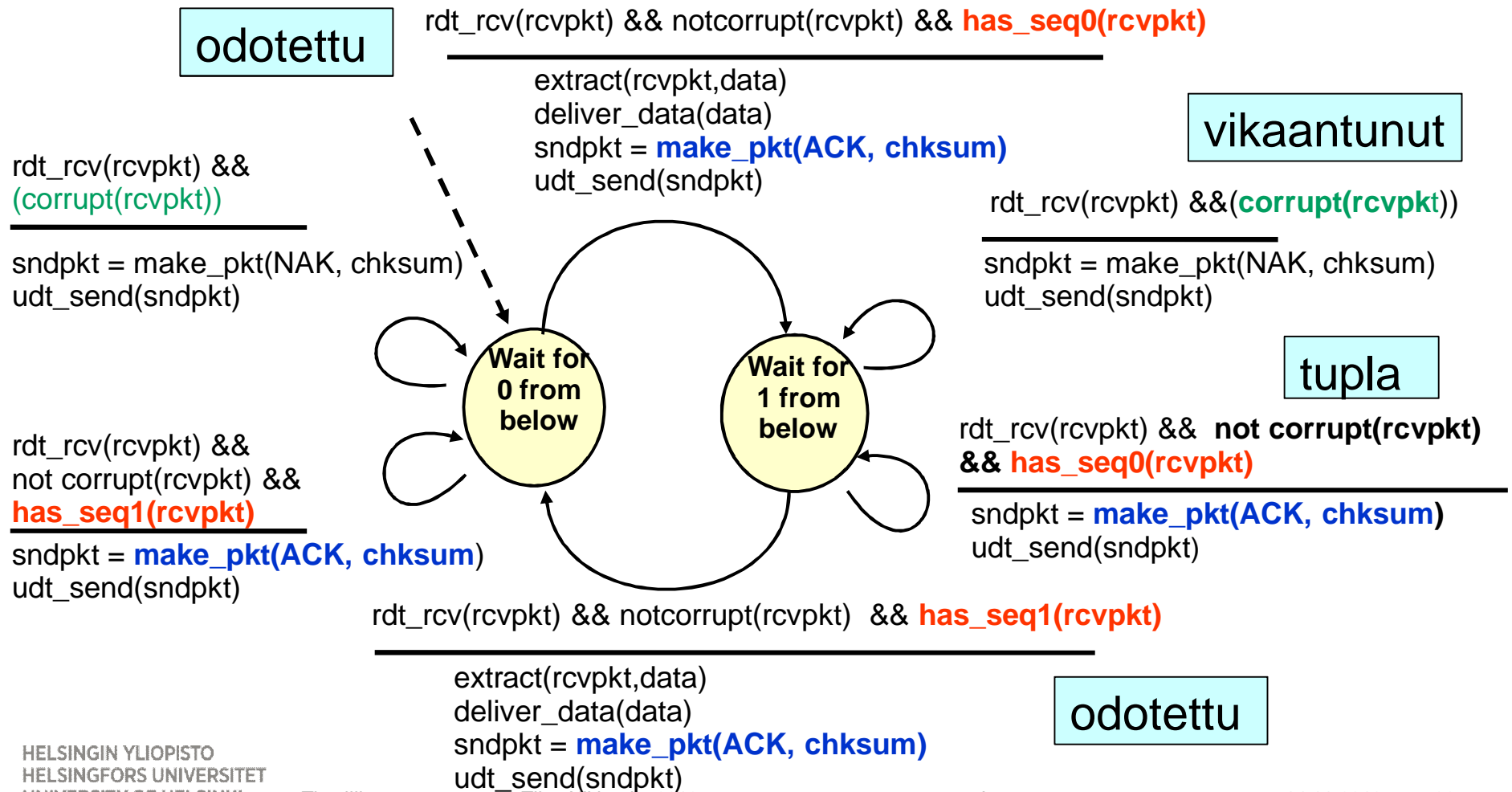
```
rdt_send(data)
sndpkt = make_pkt(0, data, checksum)
udt_send(sndpkt)
```

vikaa!





rdt2.1: Vastaanottajan automaatti





Versio rdt2.2: vain ACK-kuittaus ja vain bittivirheitä

- Sama toiminnallisuus kuin edellä
- Käyttää vain ACK-kuittauksia
 - Vastaanottaja kuittaa viimeksi saamansa kelpollisen paketin
 - Kuittaukseen on liitettävä kuitattavan paketin numero
- Jos samalle paketille (nro X) tulee useita ACK-kuittauksia (duplicate ACK), niin sitä seuraava paketti (nro X+1) joko puuttuu tai on virheellinen
 - ~NAK-kuittaus
 - Lähetetään uudelleen sanoma X+1

rdt2.2: Bittivirheitä ja vain ACK-kuittaus

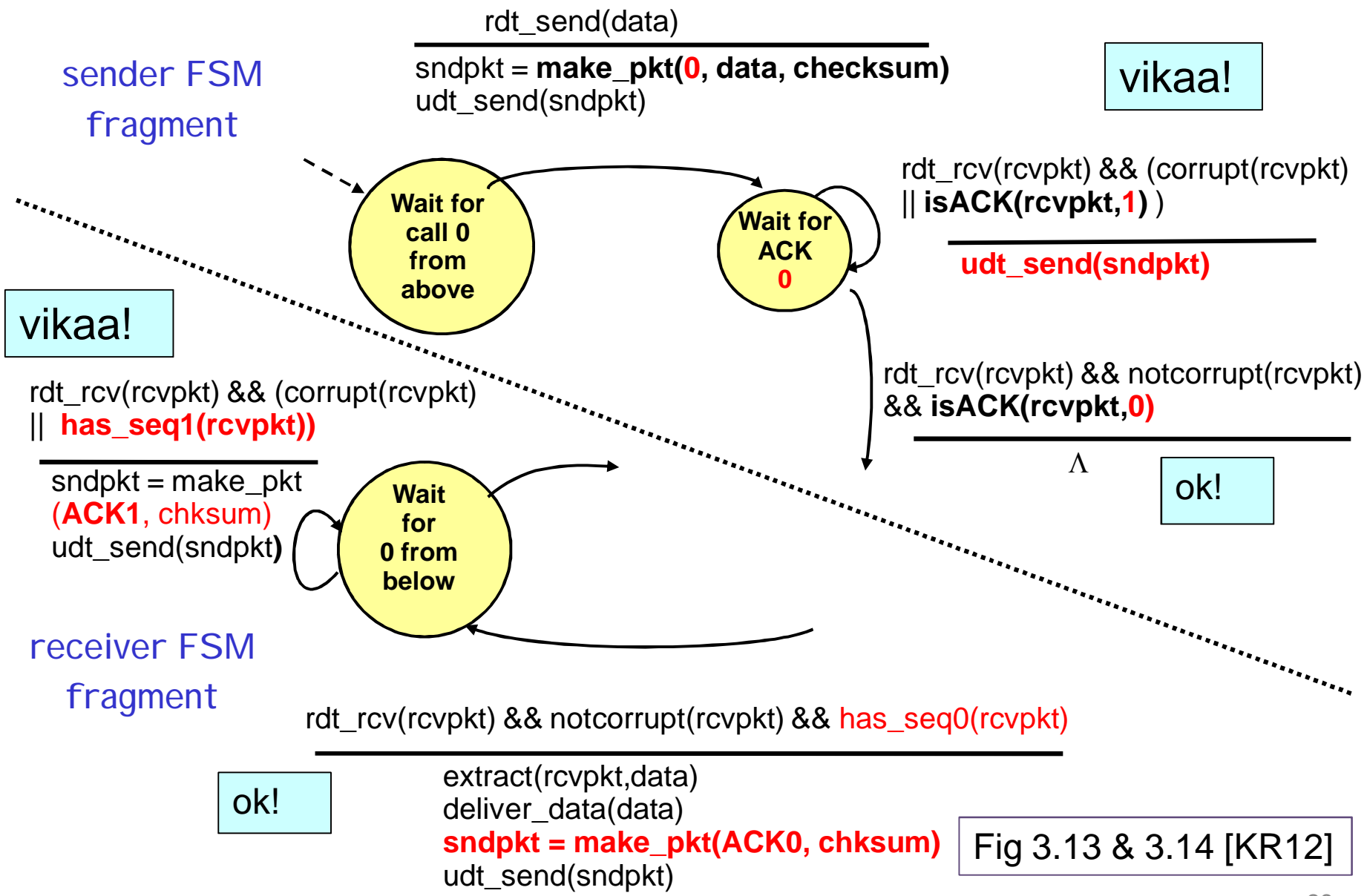


Fig 3.13 & 3.14 [KR12]



Paketteja voi kadota (→rtd 3.0)

Oletus: **Siirtokanava voi kadottaa paketteja**

Sekä datapaketteja että kuittauspaketteja voi kadota.

Tarkistussumma, pakettinumero, ACK eivät vielä riitä! Miksi?

Lähettäjä odottaa jonkin aikaa kuittausta

Jos ei saavu, lähettää paketin uudelleen

Ajastin laukaisee uudelleenlähetyksen

Jos paketti (data / kuittaus) kuitenkin vain viivästyy eikä olekaan kadonnut

Syntyy duplikaatti, joka havaitaan sanomanumeroinnin avulla

Kuittauksessa mukana kuitatun paketin numero



rdt3.0: vuorottelevan bitin protokolla

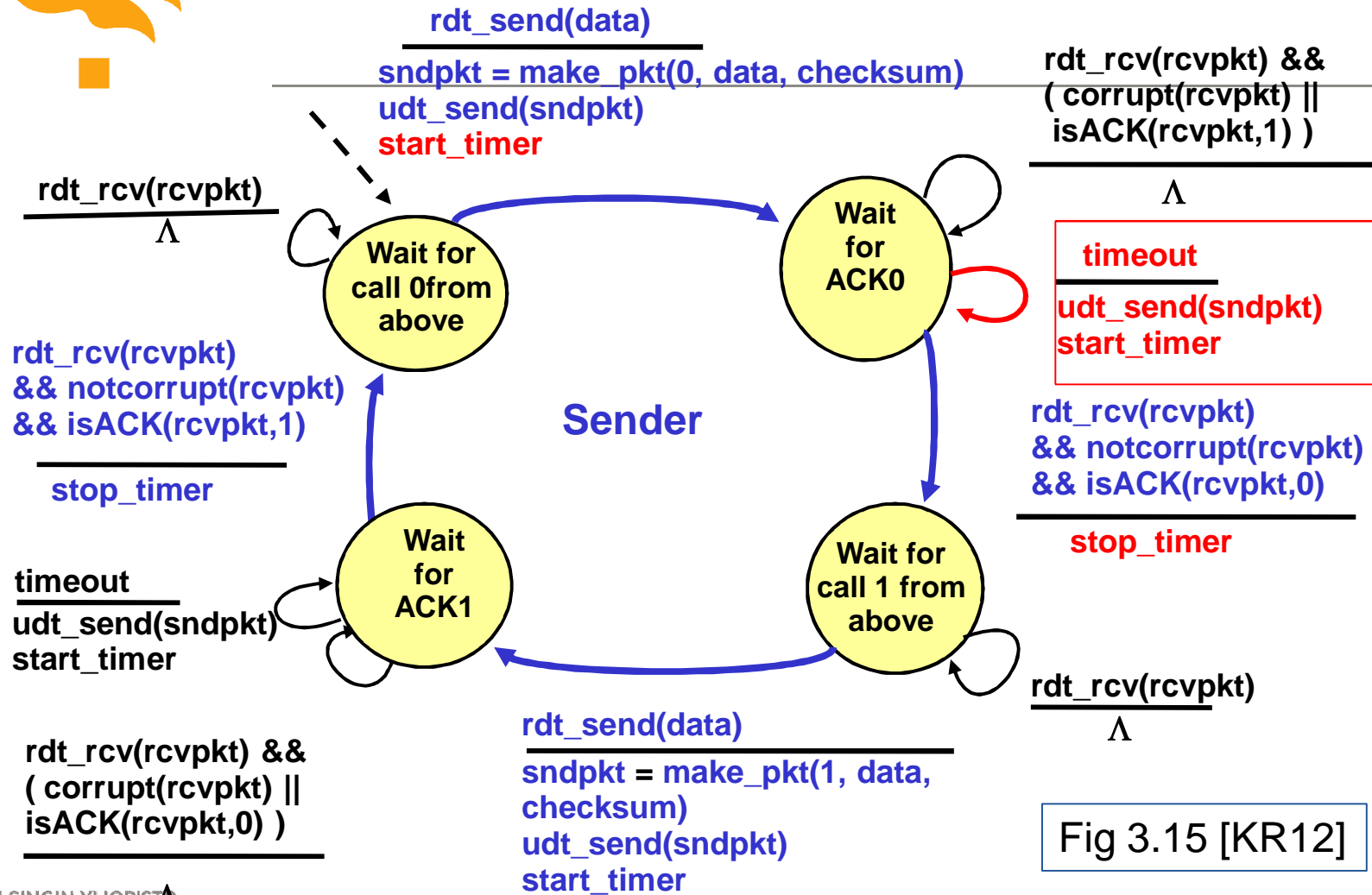
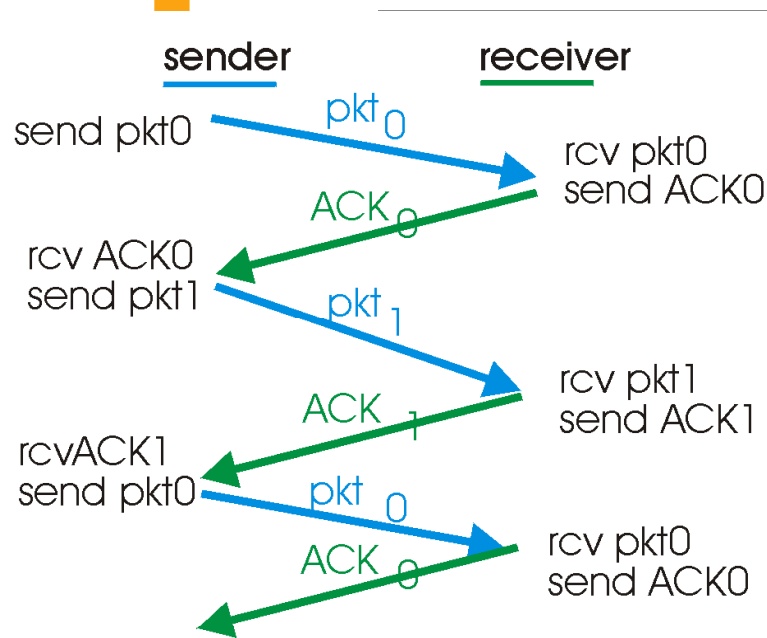


Fig 3.15 [KR12]

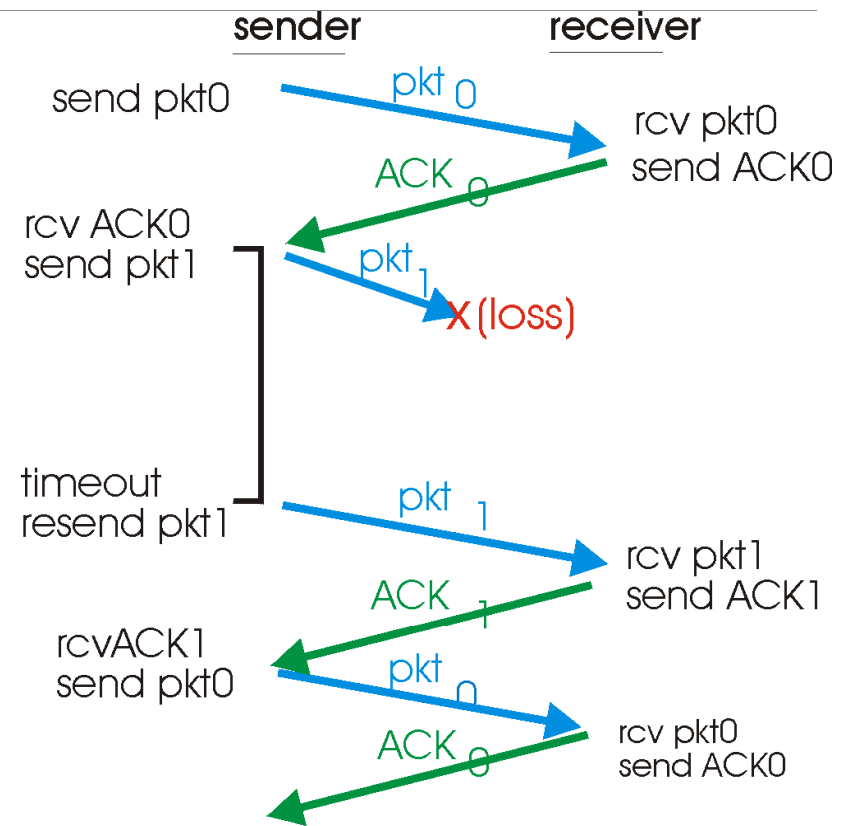


rdt3.0 toiminnassa

Fig 3.16 [KR12]



(a) operation with no loss

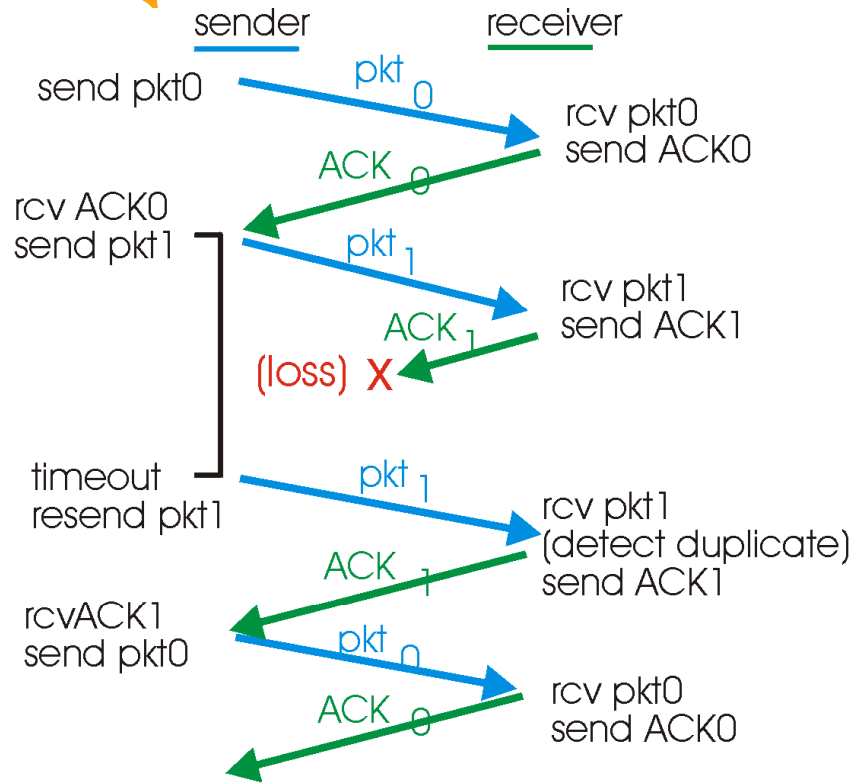


(b) lost packet

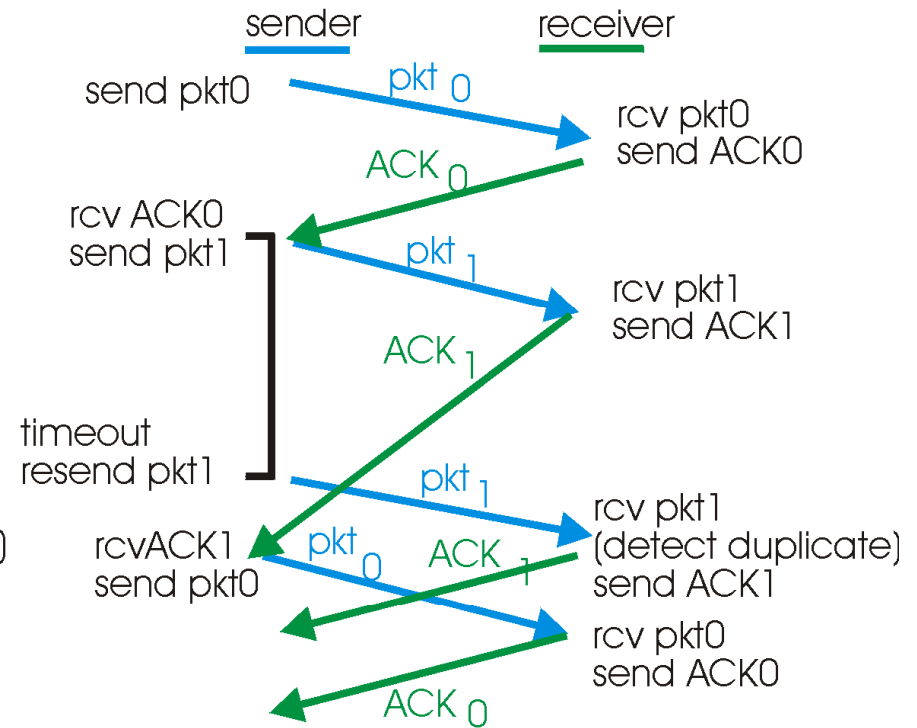


rdt3.0 toiminnassa

Fig 3.16 [KR12]



(c) lost ACK



(d) premature timeout



Yhteenveto

Ideaalitilanne (RDT 1.0)

Yksinkertainen lähetys ja vastaanotto

Vain bittivirheitä datapaketeissa (RDT 2.0)

Mekanismit: virheen tunnistus/korjaus, **ACK ja NAK** signaalit
Uudelleenlähetys NAK signaalilla

Enemmän bittivirheitä (data+signaalit) (RDT 2.1)

Mekanismit: virheen tunnistus/korjaus kaikille paketeille,
järjestysnumerot paketeille

Miksi? vääristynyt signaali voi aiheuttaa tuplalähetysten, jota ei voi tunnistaa ilman numeroita

Vain ACK käytössä (RDT 2.2)

Lisätään **järjestysnumero ACKiin**

Paketteja voi kadota (RDT 3.0)

Mekanismit: virheen tunnistus/korjaus kaikille paketeille,
järjestysnumerot paketeille (data ja ACK), ajastin



rdt3.0: Tehokkuus?

Esim: 1 Gbps linkki, 15 ms päästä-päähän etenemisviive eli
RTT = 30 ms, 1 KB:n paketti

$$T_{\text{transmit}} = \frac{L \text{ (packet length in bits)}}{R \text{ (transmission rate, bps)}} = \frac{8\text{kb/pkt}}{10^{**9} \text{ b/sec}} = 8 \text{ microsec}$$
$$U_{\text{sender}} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

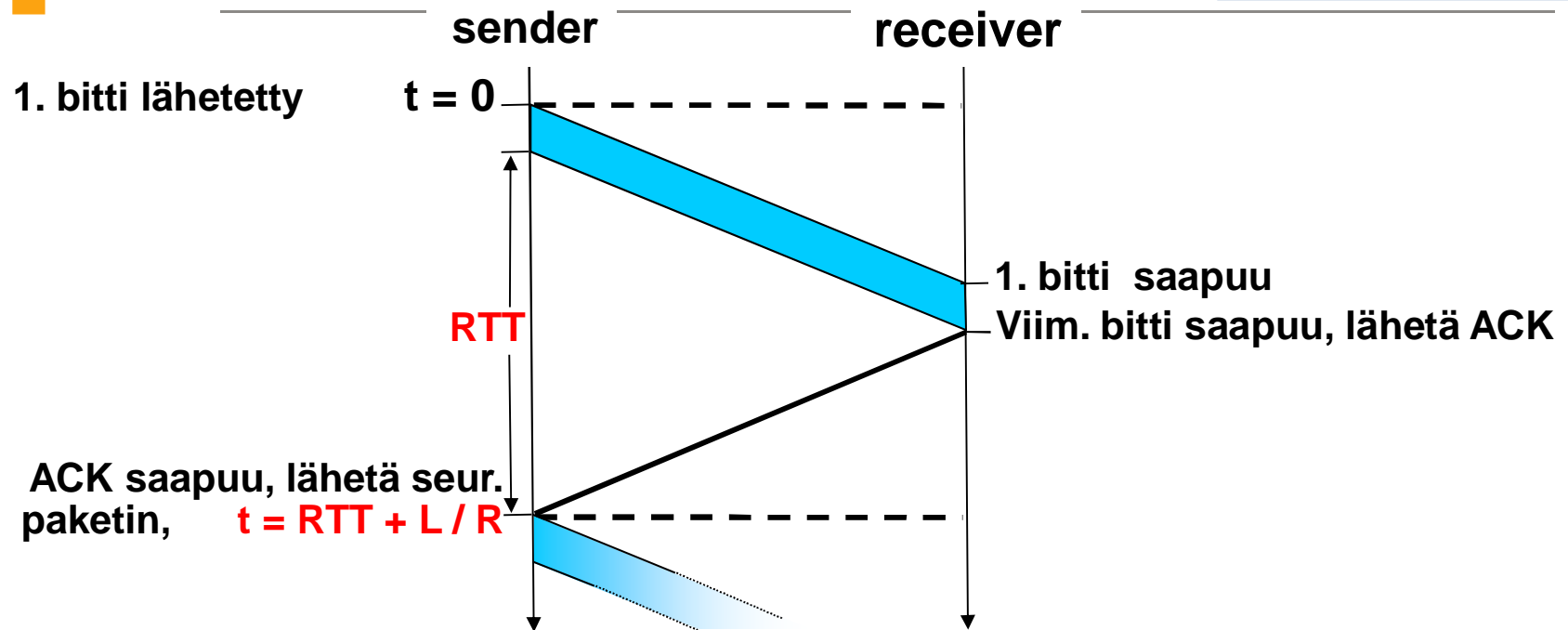
Käyttöaste (utilization): se osa kokonaisajasta, jolloin lähettäjä lähettää (RTT+L/R on kokonaisaika lähetyksessä kun odotetaan ACK)

- 1KB paketti 30 ms:n välein -> **33kB/s nopeus 1 Gbps linkillä.**
- Stop-and-wait-protokolla rajoittaa, ei linkin kyky siirtää dataa (linkin siirtonopeus)



rdt3.0: stop-and-wait tehokkuuus

Fig 3.18a [KR12]



$$U_{\text{sender}} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$



rdtX.X: Liukuhihnaprotokollat

Lähettäjä saa lähettää useita paketteja, vaikka ei ole saanut kuittauksia edeltäviin

Numerointi (0,1) ei enää riitä, lisää numeroita tarvitaan

Tarvitaan puskurointia molemmissa päissä

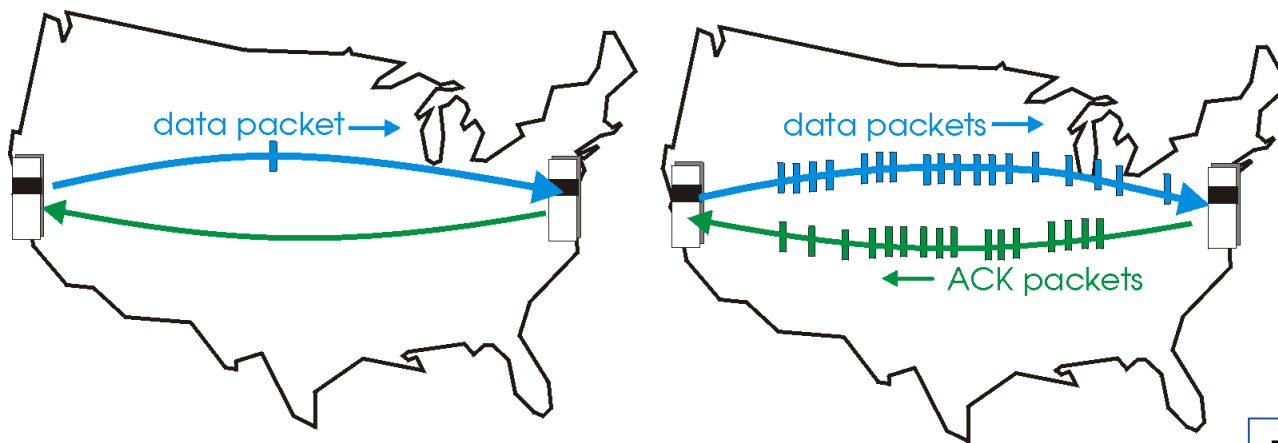
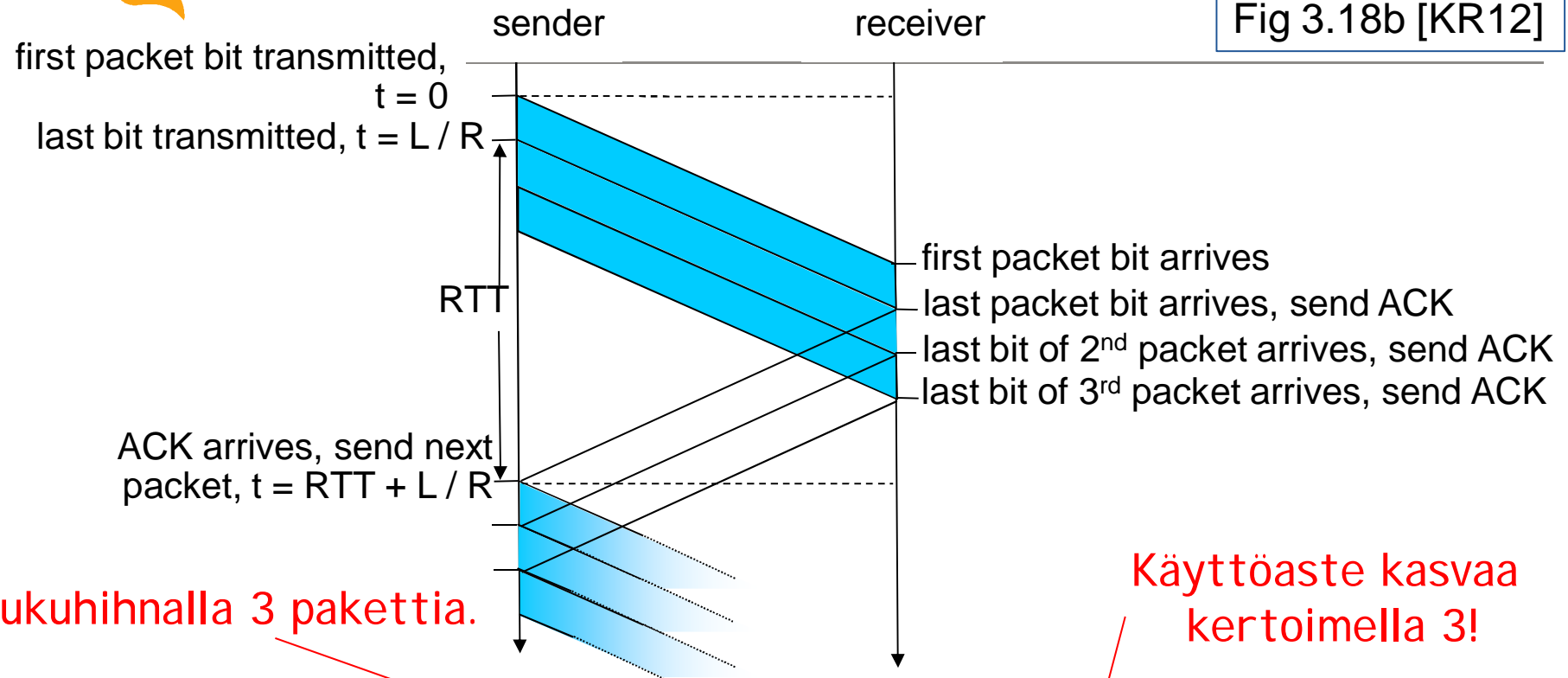


Fig 3.17 [KR12]



Liukuhihnoitus: käyttöasteen kasvu

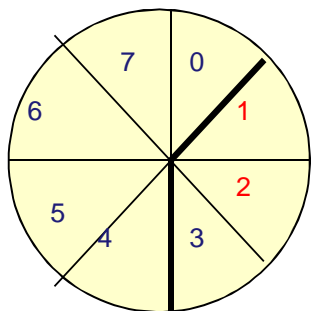
Fig 3.18b [KR12]



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



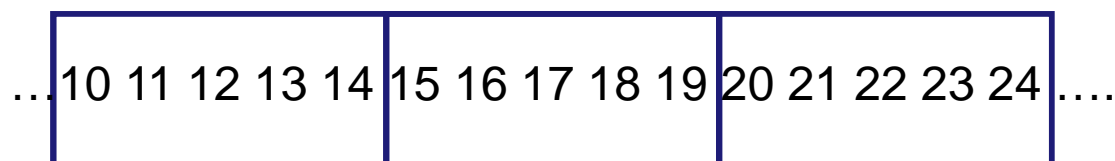
Liukuva ikkuna (sliding window)



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





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 vastapäätä
 - Ikkuna estää uusien pakettien vastaanoton
- Paketin kuittaus liu'uttaa myös vastaanottajan ikkunaa
 - Hyväksytään uusia pakettinumeroita



Kun ikkunan koko on 1

Vain yksi paketti kuittaamattomana

One Bit Sliding Window –protokolla

= stop-and-wait –protokolla

Pakettinumerot 0 ja 1 riittävät

ACK ilmoittaa

Joko **seuraavaksi odotetun** paketin numeron (esim.TCP)
tai **viimeksi vastaanotetun** virheettömän paketin numeron

ACK sisältää paketin numeron

Kuittausduplikaatti ei voi kuitata väärää paketteja



Virhetilanteen käsittely

Entä, kun huomataan virhe?

Monta muuta pakettia jo matkalla!

Pakettiin ei tule kuittausta

Paketti katosi tai virheellinen

Kuittaus katosi tai virheellinen

=> Ajastin laukeaa aikanaan

”Go-Back-N” (paluu N:ään)

Paketit uudelleenlähetetään virheellisestä lähtien

Selective Repeat (Valikoiva toisto)

Lähetetään vain virheelliset paketit



Go-Back-N

Vastaanottaja hyväksyy paketit vain järjestyksessä

- Kuittaa järjestyksessä tulleen virheettömän paketin
- Hylkää kaikki puuttuvan tai virheellisen paketin jälkeiset paketit eikä lähetä niistä kuittauksia

Kun lähettäjä ei saa pakettiin kuittausta

- Lähetysikkuna täyttyy ja estää uusien pakettien lähettämisen
- Lähettäjän ajastimet laukeavat
- Lähettäjä lähettää uudestaan kaikki viimeisen kuittauksen jälkeiset paketit
- Näiden kuittaukset siirtävät taas lähetysikkunaa

Tehoton, jos paljon virheitä ja iso ikkuna



Go-Back-N

Fig 3.19 [KR12]

Kumulatiivinen ACK

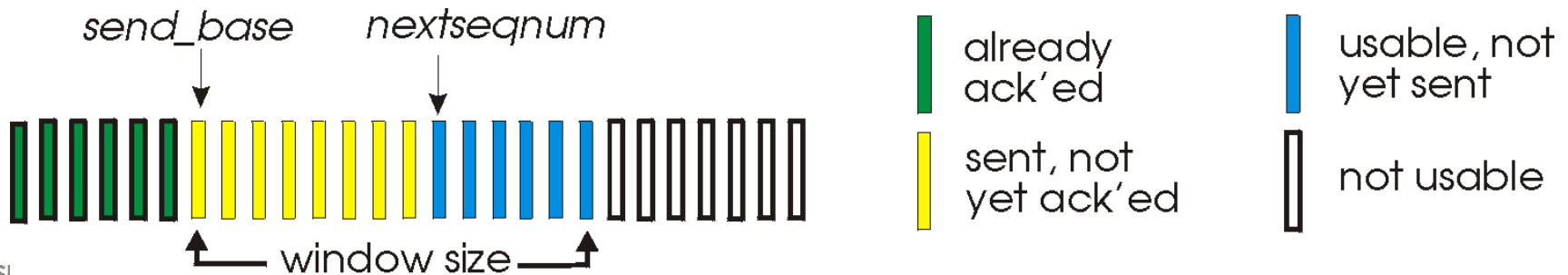
Lähetä ACK, jossa korkein järjestyksessä saadun kelvollisen paketin numero
Tämä kuittaa kaikki pienemmällä numerolla lähetetyt paketit

Jos välistä puuttuu paketti

Lähetä uudestaan ACK, jossa korkein järjestyksessä saadun paketin numero (~ NAK)

Tuplakuittaus (duplicate ACK)

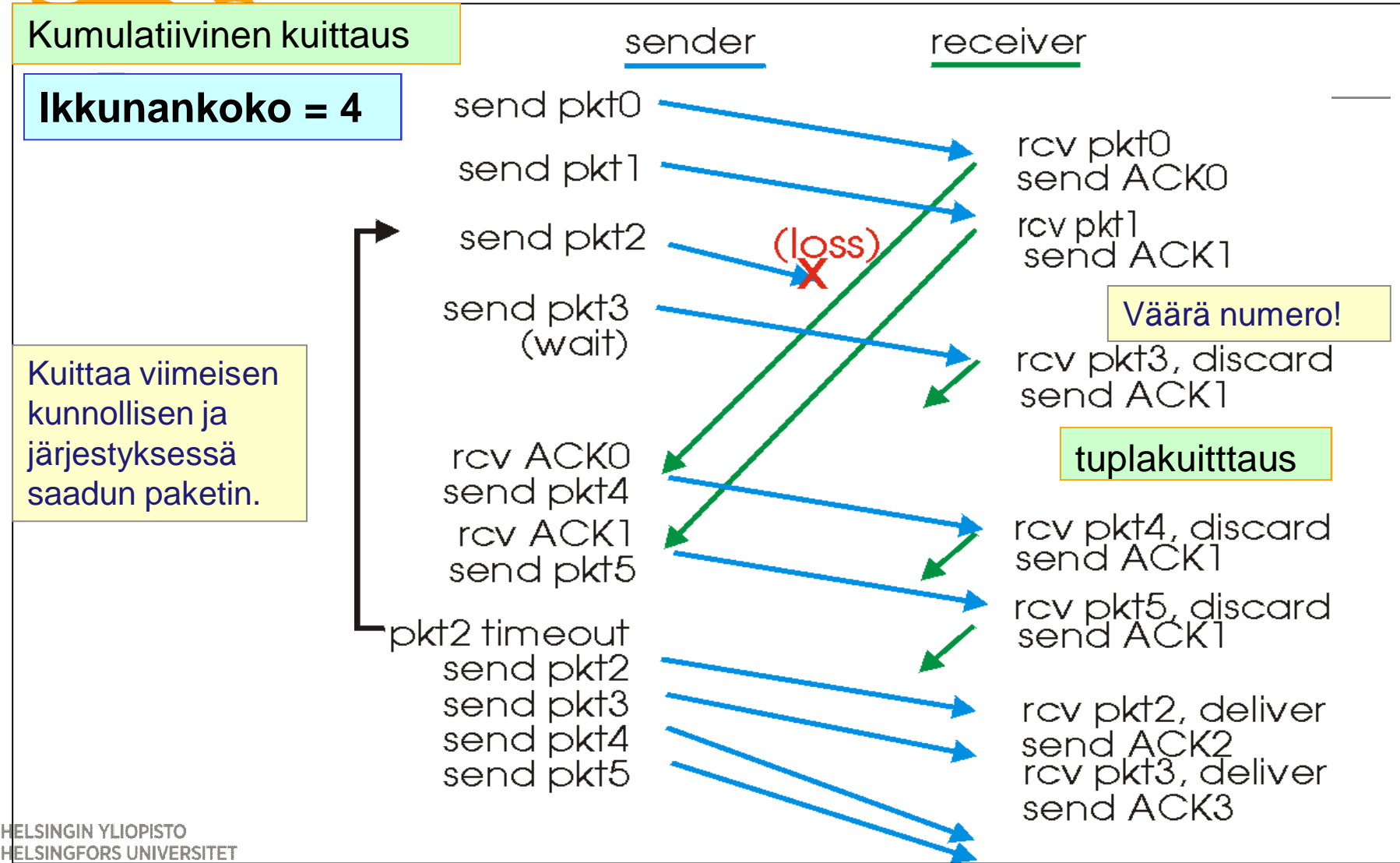
Parannus: => nopeampi reagointi puuttuvaan



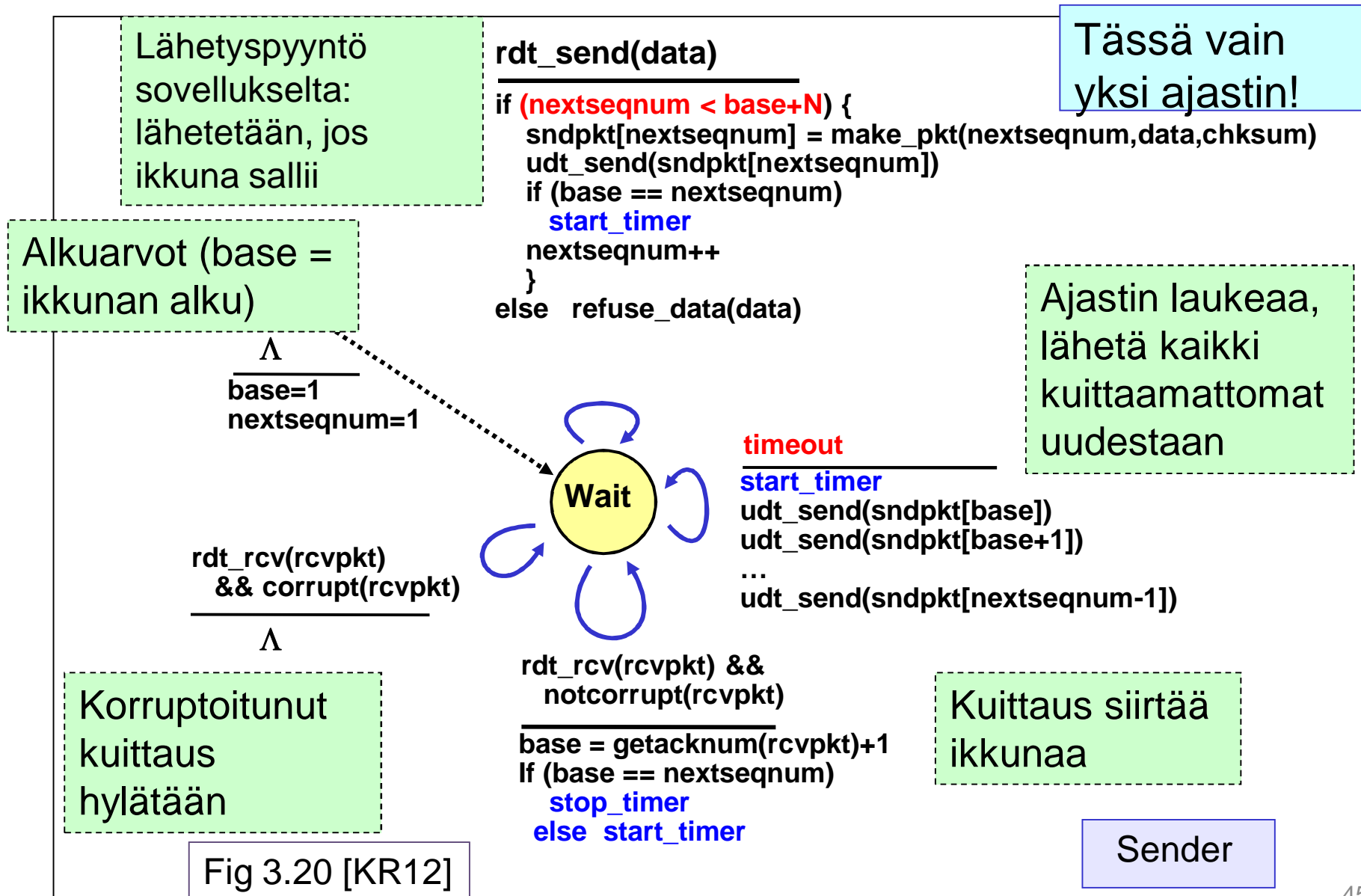


Go-Back-N: Esimerkki

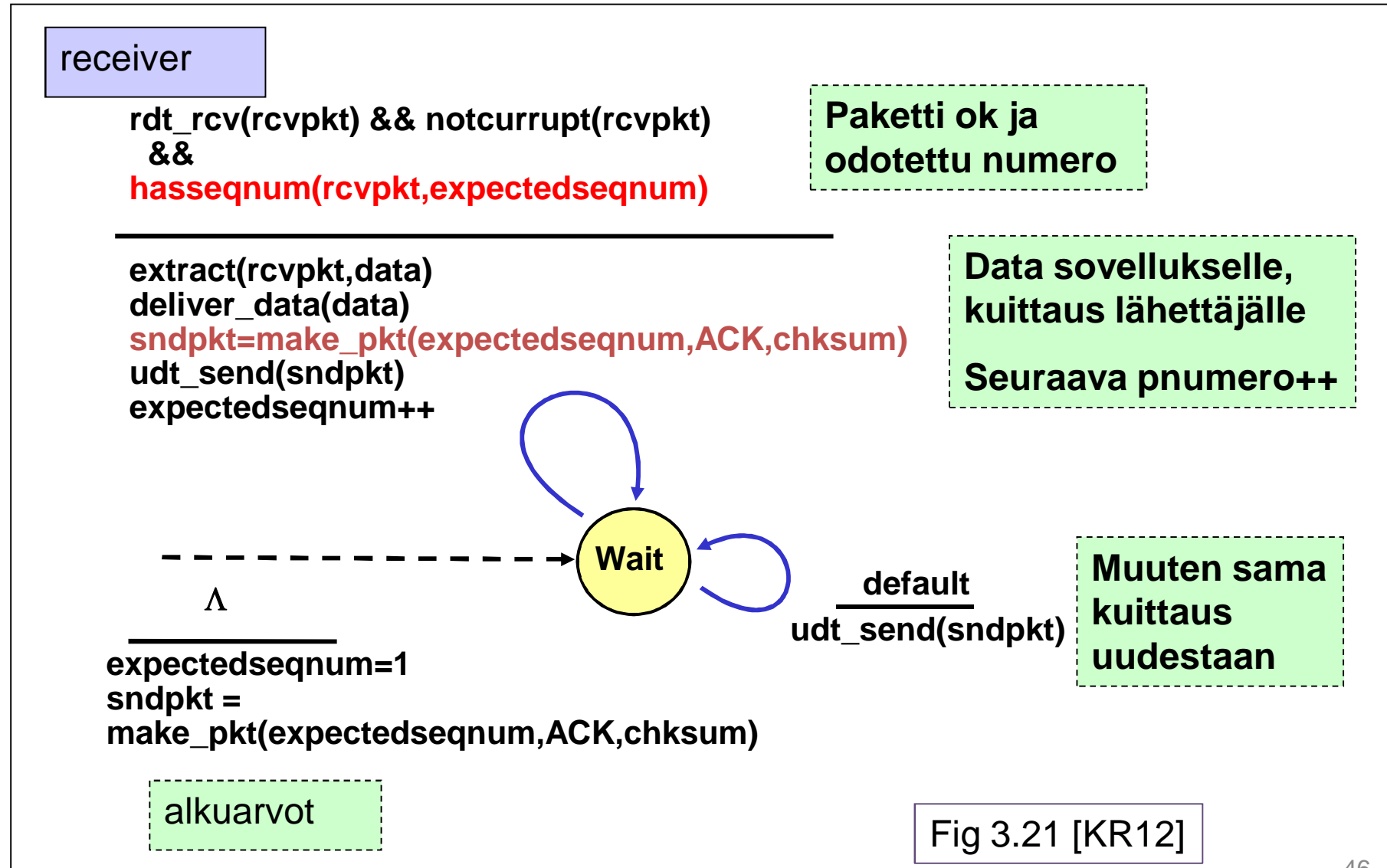
Fig 3.22 [KR12]



Go-Back-N: lähettäjän tilakaavio



Go-Back-N: Vastaanottajan tilakaavio





Valikoiva toisto (Selective Repeat)

Valikoiva uudelleenlähetys

Lähetä uudelleen vain virheellinen /puuttuva paketti

Kuittaus jokaiselle kelvolliselle paketille

Paketit sovellukselle oikeassa järjestyksessä

Vastaanottajalla oltava puskuritilaa pakettien järjestämiseen

Jos lähettäjä ei saa kuittausta paketista

Lähetysikkunan täyttyminen pysäyttää lähettämisen

Aikanaan ajastin laukeaa ja aiheuttaa uudelleenlähetysten

Jokaisella paketilla on oma ajastin

Ikkuna liikuu nytkin tasaisesti

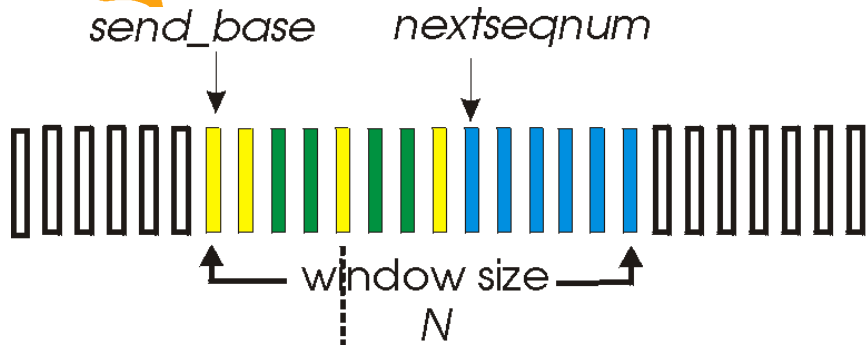
Yksi puuttuva kuittaus voi pysäyttää lähetyksen

Kun puuttuva paketti saatu, ikkuna liikuu kaikkien kuitattujen yli



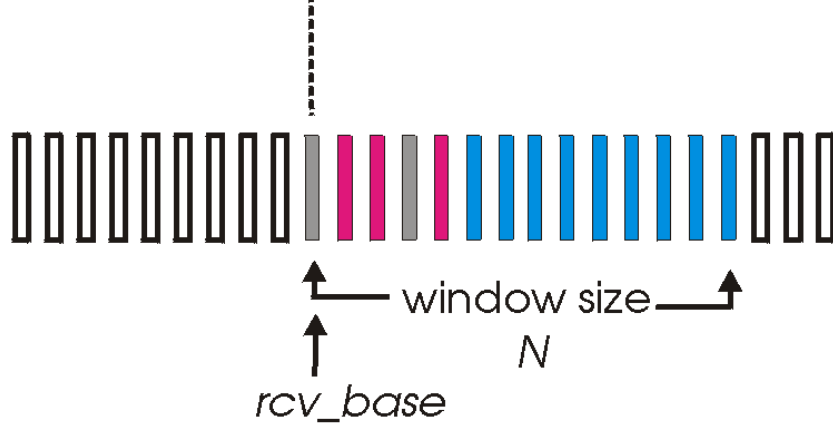
Valikoiva toisto

Fig 3.23 [KR12]



- already ack'ed
- sent, not yet ack'ed
- usable, not yet sent
- not usable

(a) sender view of sequence numbers



- out of order (buffered) but already ack'ed
- Expected, not yet received
- acceptable (within window)
- not usable

(b) receiver view of sequence numbers

Esimerkki: Valikoiva toisto

Jokainen sanoma
kuitattava erikseen!

Paketin 2 katoaminen estää
ikkunaa liikkumasta

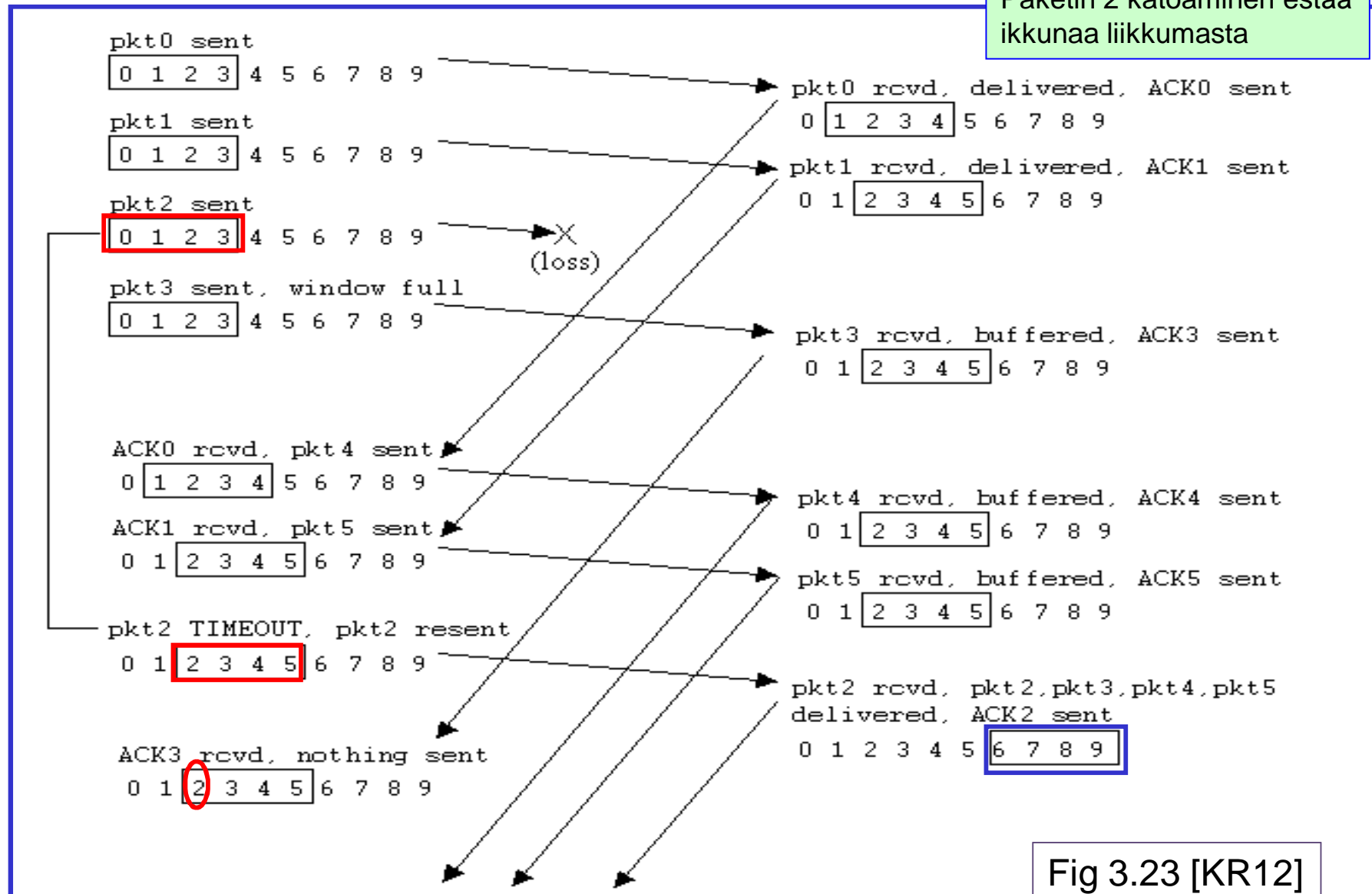


Fig 3.23 [KR12]



Ikkunankoko

Pakettinumeroille varatun kentän koko vaikuttaa myös ikkunankokoon

Yleensä jokin kakkosen potenssi

Kentän koko k bittiä \Rightarrow käytössä 2^k pakettinumeroa

$n+1$ kpl

Kun paketit numeroidaan $0, 1, \dots, n$, niin ikkunan koko saa olla korkeintaan:

MIKSI NÄIN?

Harjoitustehtävinä!

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



Yhteenvedo menetelmistä

Kts Table 3.1 [KR12]

Tarkistussumma

Ajastin

Järjestysnumero

Uudelleenlähetys vai uusi paketti

Kuittaukset

Positiiviset ACK, tuplakuittaukset

Negatiiviset NAK

Ikkunat, liukuhihnoitus



ACK vai NAK

ACK ja NAK signaaleja voidaan yhdistellä

TCP perustuu vain ACKiin

TCP: Kumulatiivinen ACK

Tupla ACK $X = \text{NAK } X+1$

Selektiivinen ACK

Parempi toteuttaa oma NAK signaali, tehokkaampi

Ei välttämättä tarvita NAKia

Pelkkä NAK