

Tietoliikenteen perusteet

Kuljetuskerros

Kurose, Ross: Ch 3

Sisältöä

- n Kuljetuspalvelut
- n Yhteydetön kuljetuspalvelu, UDP
- n Luotettavan kuljetuspalvelun periaatteet
- n Yhteydellinen kuljetuspalvelu, TCP
- n Ruuhkanhallinta TCP:ssä

Oppimistavoitteet:

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





Kuljetuskerros

Kuljetuspalvelu



Kuljetuskerros

n Tarjoaa kuljetuspalvelun **prosessin** välille

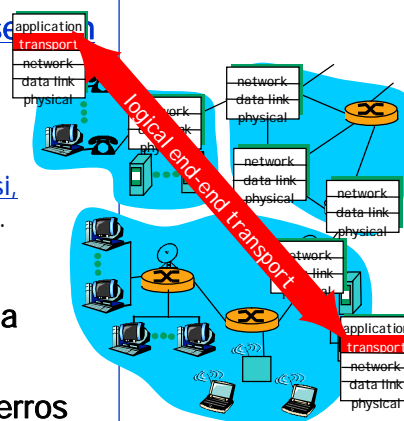
n Vain isäntäkoneissa

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

Vastaanotto: Kokoa segmentit sanomaksi, jonka sovellus lukee.

n Verkkokerros reitittää koneesta koneelle

n Segmentin koko s.e. verkkokerros pystyisi välittämään sellaisena



KuRo05: Fig 3.1



Sovelluksen vaatimuksia

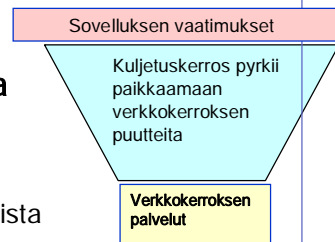
n Verkkokerroksen palvelu voi

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

n Sovellus edellyttää kuljetuspalvelulta

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

n Kuljetuskerros peittää verkkokerroksen puutteita ja parantaa sovelluksen näkemää palvelun laatua



Internetin kuljetusprotokollat

n TCP: luotettava, järjestyksen säilyttävä tavujen kuljetuspalvelu

- n **Virheenvalvonta** (error control): Huomaa ja korjaa virheet, hylkää kaksoiskappaleet
- n **Vuonvalvonta** (flow control): Älä ylikuormita vastaanottajaa
- n **Ruuhkanhallinta** (congestion control): Älä ylikuormita verkkoa
- n **Yhteyden muodostaminen ja purku**

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

- n Välittää vain sanomia, ei pyri mitenkään parantamaan verkkokerroksen tarjoamaa palvelun laatua
- n Luotettavuus jää sovelluskerroksen hoidettavaksi

n Kumpikaan kuljetuspalvelu ei anna takuita viiveelle tai siirtonopeudelle ("best effort")



Mikä kone /Mikä prosessi?

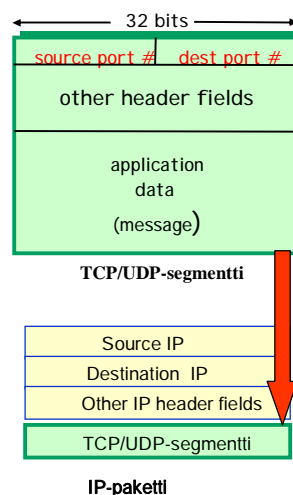
- n Kuljetuskerros tarjoaa päästä-päähän yhteyden
 - n Prosessilta prosessille (~ pistokkeesta pistokkeeseen)
 - n Prosessi lukee ja kirjoittaa sanomia halutessaan
- n Datana lisäksi on välitettävä osoitetietoja
 - n Vastaanottajan ja lähettäjän tiedot
 - n Eri koneiden prosessit voivat käyttää samaa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



Mikä kone /Mikä prosessi?

Lähetys (asiakas)

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



Mikä kone /Mikä prosessi?

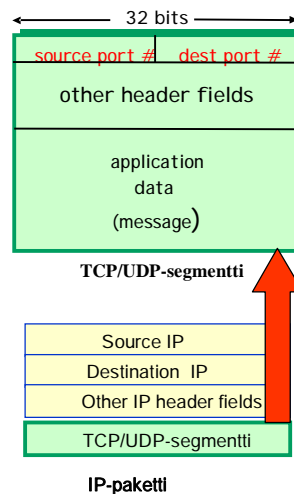
Vastaanotto (palvelija)

Verkkokerros

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

Kuljetuskerros

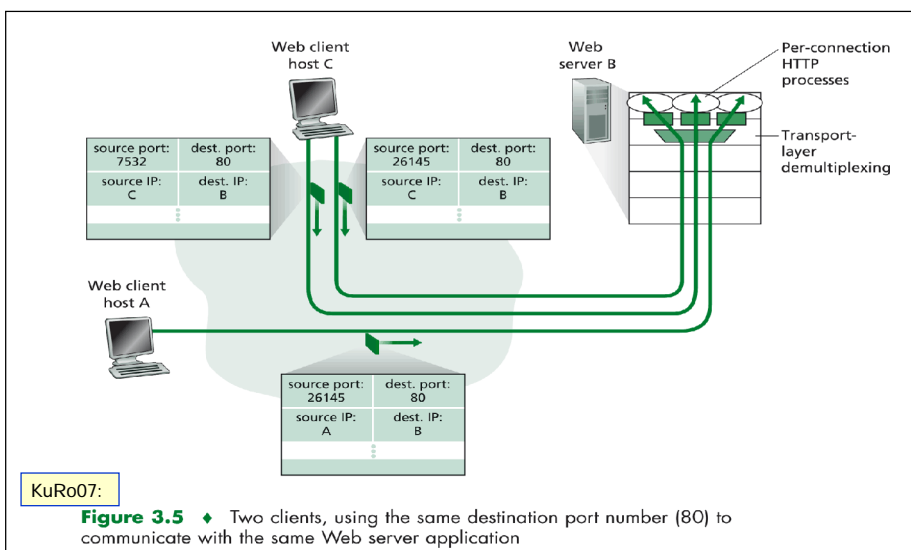
- Poistaa kuljetuskerroksen otsaketiedot
- Kokoaa yhteenkuuluvat segmentit sanomiksi (tavuvirraksi)
- Ohjaa sanoman (tavuvirran) oikealle prosessille (eli oikeaan pistokkeseen) porttinumeron avulla
 - TCP: huolehtii myös luotettavuudesta
 - UDP: tarjoaa pelkän välityspalvelun



Tietoliikenteen perusteet /2007/ Liisa Marttinen

9

Kaksi www-asiakasta ja palvelija



Tietoliikenteen perusteet /2007/ Liisa Marttinen

10



Kuljetuskerros

Yhteydetön kuljetuspalvelu UDP



UDP (User Datagram Protocol)

n Yhteydetön

KJ ei pidä tallessa mitään sovellusten väliseen keskusteluun liittyvää.

Sovellus antaa aina sanoman lisäksi kohdeosoitteen ja kohdeportin

n Ei takaa luotettavuutta (~'epäluotettava'?)

n vain minimi palvelu: mille koneelle, mihin porttiin

n voi kadottaa sanoman

n Ei myöskään säilytä sanomien järjestystä

n Sovellus saa sanomat siinä järjestyksessä kun ne tulevat perille

n Vähän yleisrasitetta

n Aikaa ei kulu yhteyden muodostukseen eikä purkuun

n Ei kulu resursseja yhteyden tilatietojen ylläpitoon

n Pieni otsake

n Ruuhkanhallinta ei säännöstele liikennettä



Käyttö

- n Vaikka UDP ei takaa luotettavuutta, se sopii silti monen sovelluksen tarpeisiin
 - Alkujaan oli vain TCP, UDP luotu myöhemmin
- n Miksi UDP?
 - Pieni yleisrasite
 - Sovellus voi sietää virheitä
 - Reaaliaikavaatimuksia, saavuttaa suuremman siirtonopeuden
- n Tarvittava luotettavuus on räätälöitävissä sovellukseen
 - Sovellusprotokolla: oma sanomanumerointi, uudelleenlähetykset



Kuljetusprotokollat

- n 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



Verkkosovelluksia

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	Typically UDP
Streaming multimedia	typically proprietary	Typically UDP
Internet telephony	typically proprietary	Typically UDP
Network management	SNMP	Typically UDP
Routing protocol	RIP	Typically UDP
Name translation	DNS	Typically UDP

Figure 3.6 ♦ Popular Internet applications and their underlying transport protocols

Kuro05:

Miksi nämä sovellukset suosivat UDP:tä?



UDP-segmenti rakenne

n Porttinumerot

- n Koska prosessien välinen palvelu

n Length

- n Segmentin kokonaispituus
otsake (8 B) mukaanluettuna

n Checksum (optionaalinen)

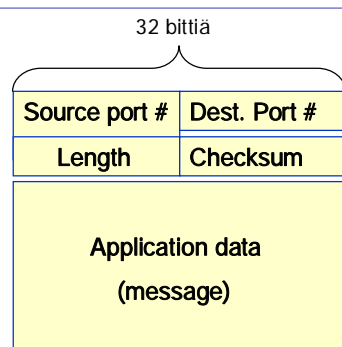
- n Bittivirheen havaitsemiseen
- n UDP ei yritä toipua, hävittää virheellisen segmentin

n Data

- n Pitkä sanoma pilkottuna useaksi segmentiksi

n IP-osoitteet vasta verkkokerroksen otsakkeessa

- n Näitä tarvitaan reitityksessä

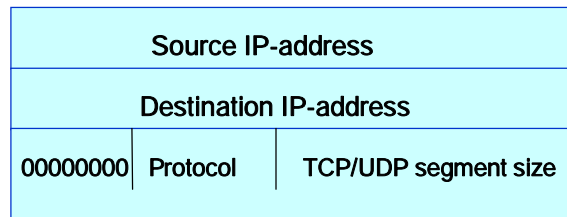


UDP-otsake



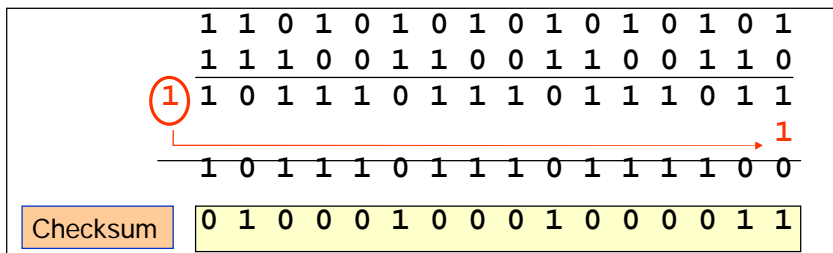
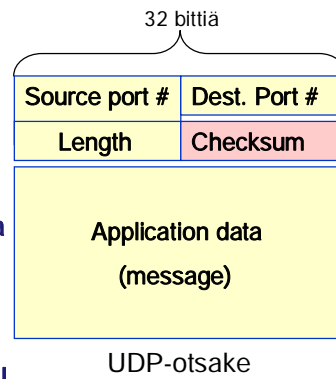
Kuljetuskerroksen pseudo-otsake

- n Käyttö vain isäntäkoneen sisäisesti. Ei lähetetä verkkoon.
 - n UDP laskee tarkistussumman otsakkeelle, datalle ja ns. **pseudo-otsakkeelle**, joka sisältää IP-otsakkeen tietoja
 - n Varmistus, että segmentti on tullut oikeaan koneeseen ja oikeaan porttiin



UDP-tarkistussumma

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





Esimerkki

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

Lähetäjä:	vastaanottaja:	
1011 0100	1011 0100	1011 0100
0111 0101	0111 0101	1111 0101
1000 1101	1000 1101	1000 1101
0000 0000	0100 1000	0100 1000
=====	=====	=====
1 1011 0110	11111 1110	1 0111 1110
.....→	1	1
=====	=====	=====
1011 0111	1111 1111	0111 1111
↓	OK!	Virhe!
0100 1000		

Yhden komplementti

Tietoliikenteen perusteet /2007/ Liisa Marttinen

19



Miksi UDP-tarkistussumma

n Kaikki linkkikerrokset eivät suorita tarkistuksia!

n Ethernet huolehtii kyllä

n UDP-tarkistussumma ei ole kovin tehokas havaitsemaan virheitä!

n UDP ei yritä toipua virheistä!

n Jotkut toteutukset voivat tuhota virheellisen segmentin

n Jotkut antavat sen sovellukselle varoituksen kera

n Lisärasite?

n Ei tarvitse käyttää, jos ei halua. Tällöin lähettäjä laittaa tarkistusummaksi pelkkiä nollia

Tietoliikenteen perusteet /2007/ Liisa Marttinen

20



Tehtäviä:

n Lähetetään 10 tavun viesti UDP:llä.

n Miten kauan kestää lähettäminen, jos lähetysnopeus on 56 kbps?

$$10 \text{ tavua} + 8 \text{ tavua} = 18 * 8 \text{ b} = 144 \text{ bittiä}$$
$$144 \text{ b} / 56\,000 \text{ b/s} = 2.57 \text{ ms}$$

n Miten suuri on etenemisviive, jos etäisyys lähettäjältä vastaanottajalle on 1000 km?

$$1000 \text{ km} / 200\,000 \text{ km/s} = 5 \text{ ms}$$

n Miten suuri on UDP-otsakkeen aiheuttama yleisrasite (overhead)?

$$8/18 = 0.44 \text{ eli } 44 \%$$

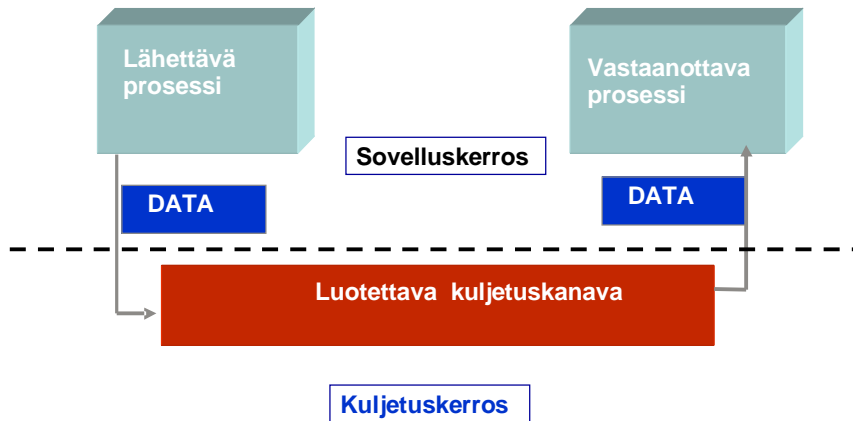


Kuljetuskerros

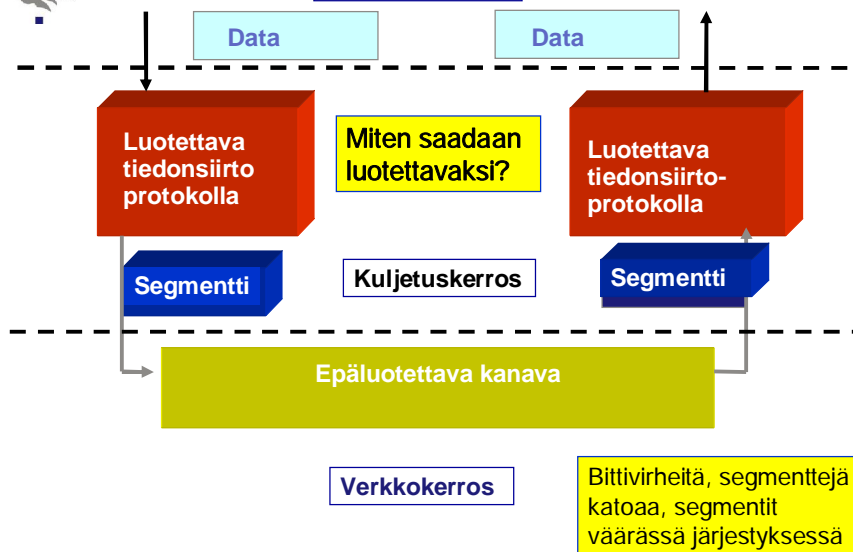
Luotettavan
kuljetuspalvelun
periaatteet



Luotettava tiedonsiirto

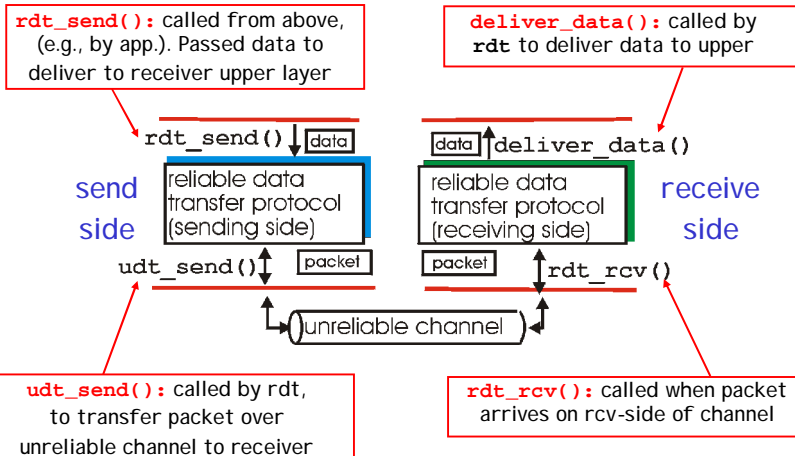


Sovelluskerros



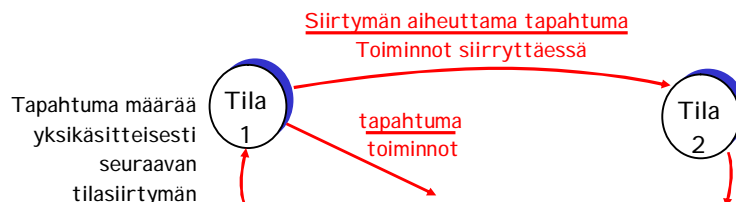


Kuinka tehdään luotettavaksi?



Kuinka saada luotettavaksi?

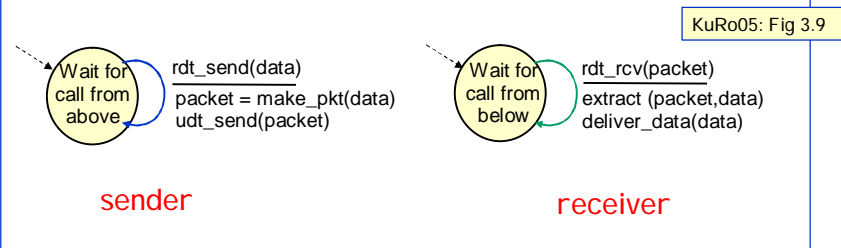
- r Tarkastellaan yleisesti luotettavan tiedonsiirron ongelmia ja erilaisia ratkaisuyrityksiä
 - r Edeten ideaalitalanteesta yhä ongelmaisempaan
 - r Käyttäen äärellisiä tila-automaatteja lähettäjän ja vastaanottajan toiminnan kuvaamiseksi





Versio rdt1.0: Ideaalitilanne

- Oletus: **siirtokanava on täysin luotettava**
 - Ei bittivirheitä, ei katoavia paketteja, ei epäjärjestyä
- Luotettava kuljetuspalvelu =
 - Lähettäjä kirjoittaa datan kanavaan,
 - Vastaanottaja lukee datan kanavasta

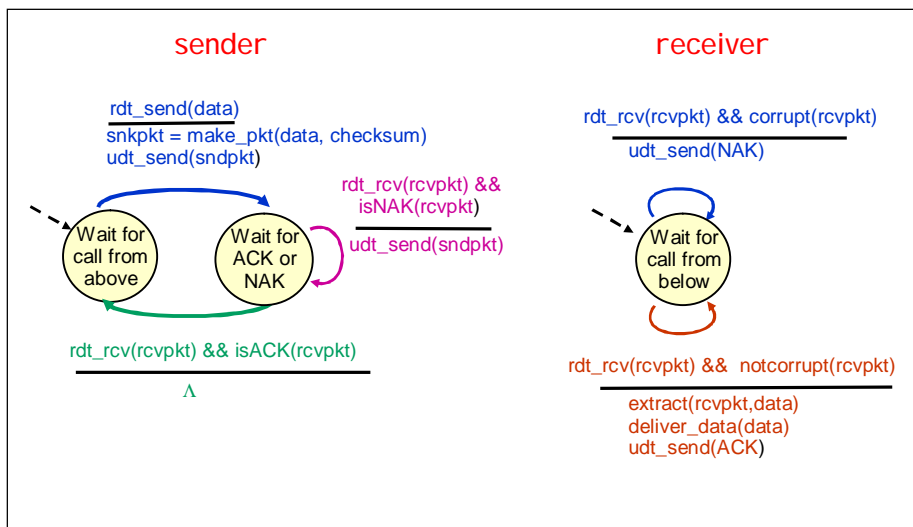


Versio rdt2.0: Bittivirheitä

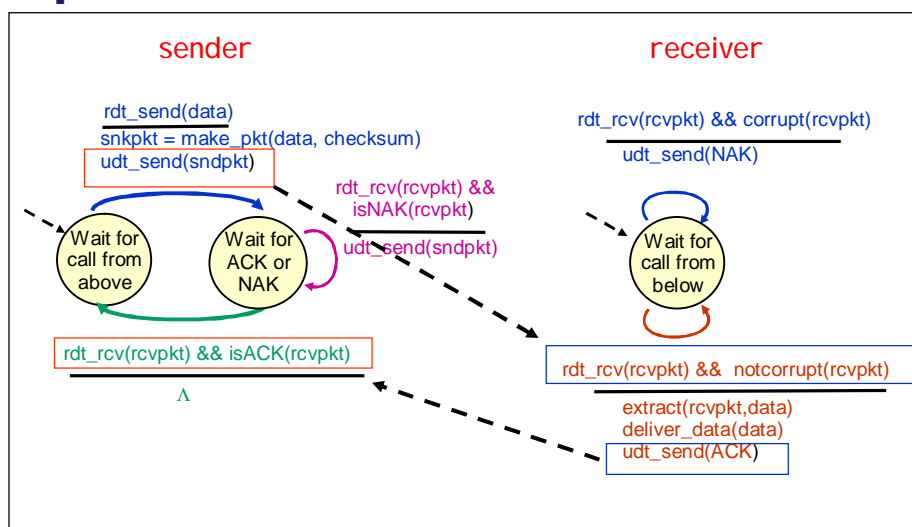
- Oletus: Voi olla **vain bittivirheitä**
 - Bitti voi käännyä siirron aikana
 - **Siirto ei kadota paketteja**
- **Kuinka toipua?**
 - Kuittaukset: vastaanottaja kuittaa **ACK**:lla virheettömän paketin,
 - **NAK**-kuittaus, jos paketti on virheellinen + hylkää
 - Jos NAK, niin lähettäjä lähettää paketin uudelleen
- **Stop-and-wait-protokolla**
 - Lähettäjä odottaa kuittausta ennenkuin lähettää seuraavan
- Luotettava kuljetuspalvelu
 - Virheen huomaaminen: **tarkistussumma**
 - Palaute vastaanottajalta: **kuittaussanoma** (ACK / NAK)
 - **Uudelleenlähety**s: dataa puskuroitava



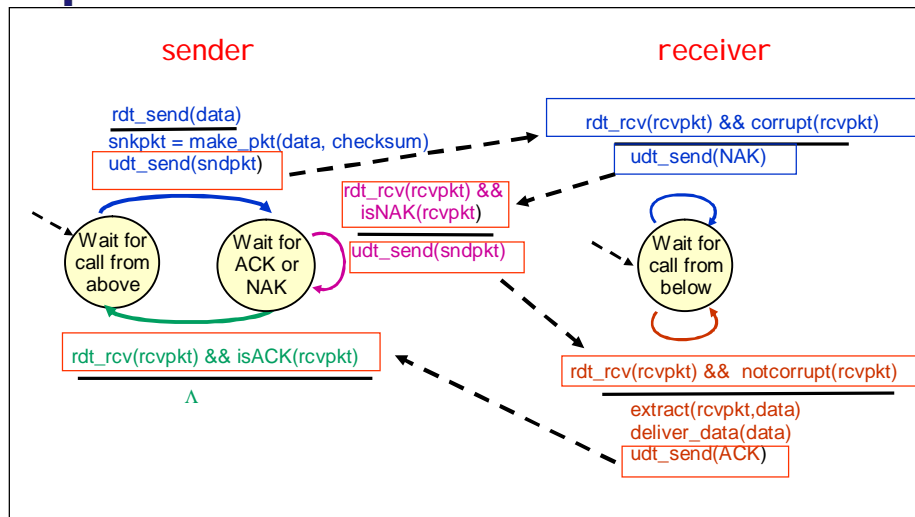
rdt2.0:



rdt2.0: Toiminta, kun ei ole virhettä



rdt2.0: Toiminta virhetilaneessa



rdt2.0: Missä voi mennä väärin?

- r Ei toimi, jos ACK /NAK -bitit korruptoituvat!
 - r Onko OK vai ei?
- r Korjaus:ACK/NAK-paketteihin tarkistusbitit, jotta virhe huomataan
- r Entä toipuminen?
 - r Jos jos kuittauksessa virhe, uudelleenlähetetään paketti
 - r Uudelleenlähetys voi tuottaa kaksoiskappaleen, jotka on huomattava ja hylättävä
- r => Paketteihin järjestysnumero
 - r Kaksoiskappale: jos sama numero
 - r Vastaanottaja kuittaa normaalisti, mutta ei anna sovellukselle



Versio rdt2.1: enemmän bittivirheitä

Lähettäjä:

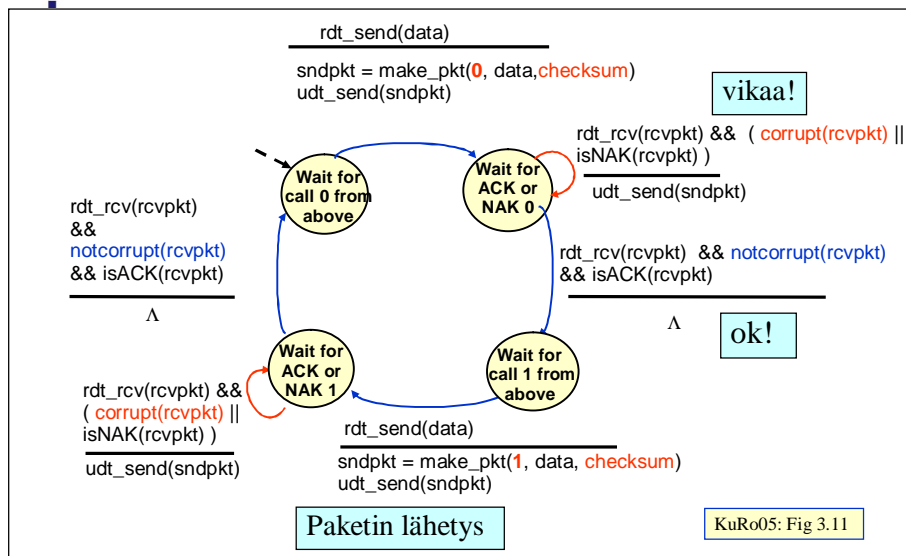
- n Lisää pakettiin järjestysnumeron
(numerot 0 ja 1 riittävät tässä protokollassa. Miksi?)
Ns. vuorottelevan bitin protokolla
- n Tarkista, että ACK/NAK ei ole korruptoitunut
Tilakaaviossa nyt kaksinkertaisesti tiloja
Kaavion tilan 'muistettava' paketin numero
- n Säilytä kopio lähetetystä paketista

Vastaanottaja:

- n Tarkista, ettei ole kaksoiskappale
 - n Kaavion tilan 'muistettava' seuraavan paketin numero: 0 vai 1
- n Myös duplikaatti kuitattava, koska ei tiedä menikö edellinen kuittaus perille

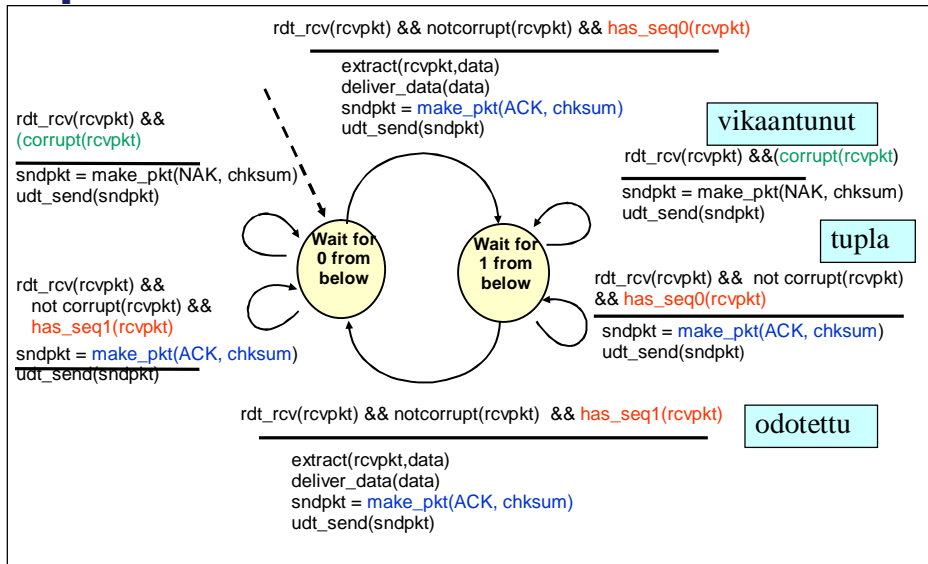


rdt2.1: Lähettäjän tilakaavio





rdt2.1: vastaanottajan tilakaavio



Tietoliikenteen perusteet /2007/ Liisa Marttinen

35

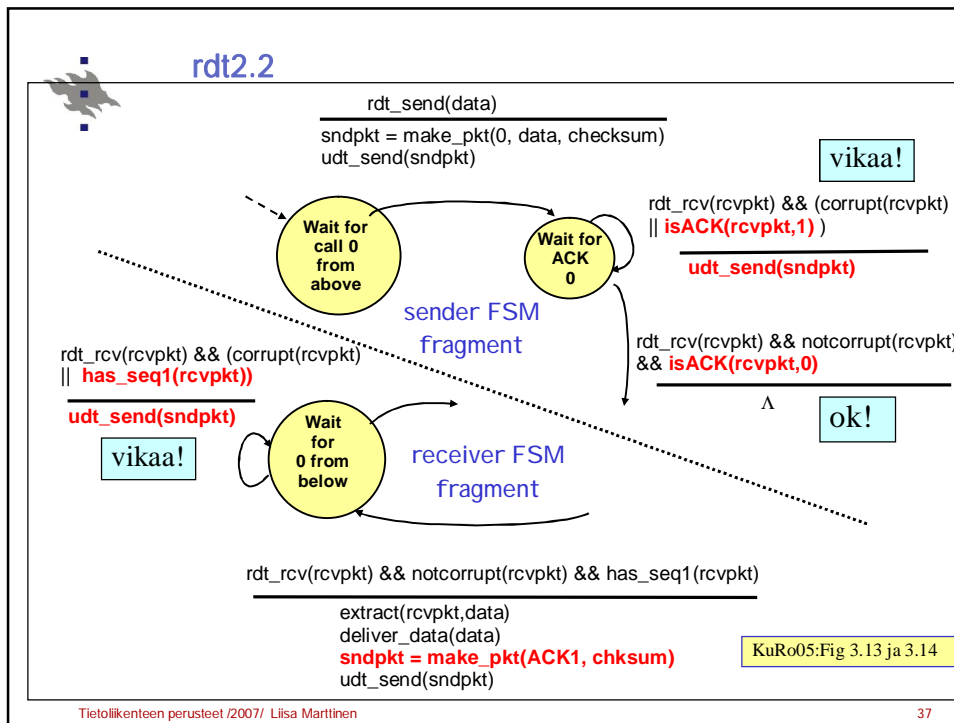


Versio rdt2.2: vain ACK-kuittaus käytössä

- ┆ Sama toiminnallisuus kuin edellä
- ┆ Käyttää vain ACK-kuittauksia
 - ┆ Vastaanottaja kuittaa viimeksi kunnossa saamansa 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

Tietoliikenteen perusteet /2007/ Liisa Marttinen

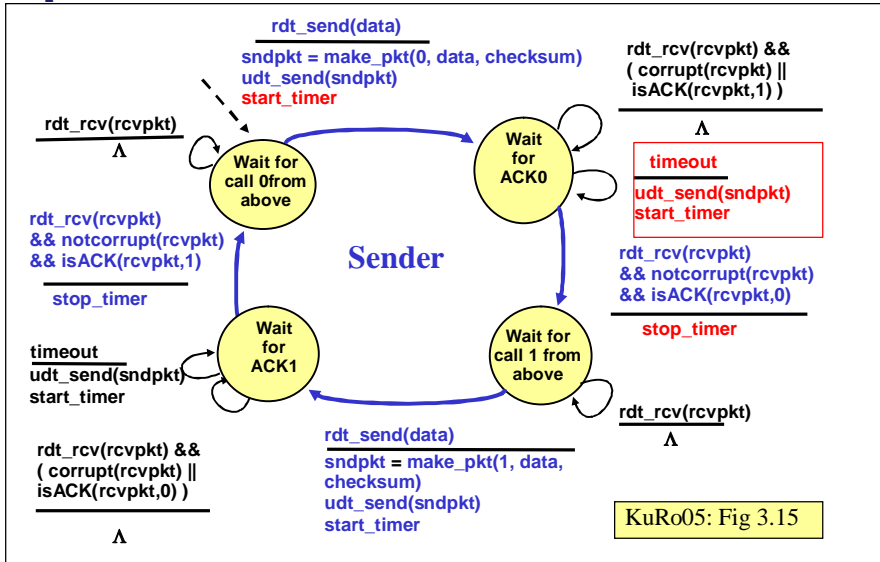
36



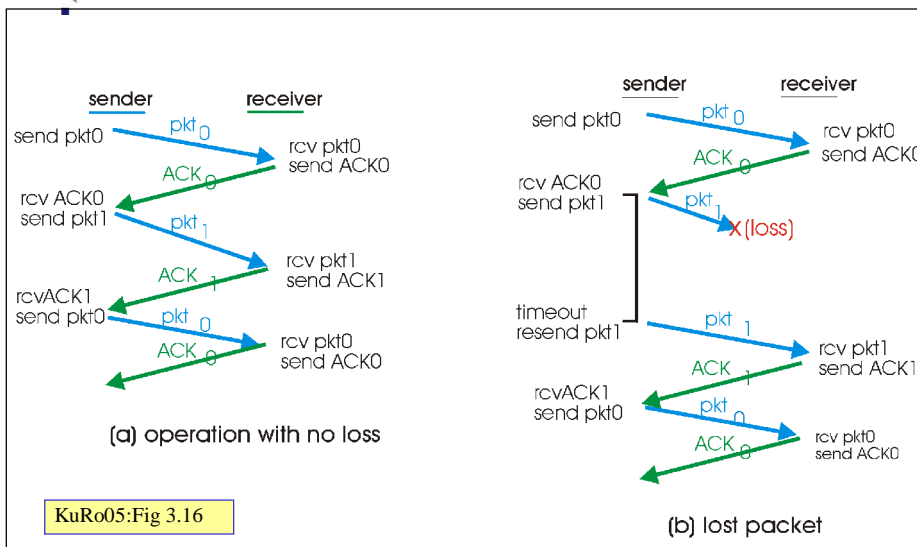
- ### Versio rdt3.0: paketteja voi kadota
- Q Oletus:** Siirtokanava voi kadottaa paketteja
 Sekä datapaketteja että kuittauspaketteja voi kadota.
 Tarkistussumma, pakettinnumero, ACK eivät vielä riitä! **Miksi?**
 - Q** Lähettäjä odottaa jonkin aikaa kuittausta
 Jos ei saavu, lähettää paketin uudelleen
Ajastin laukaisee uudelleenlähetyksen
 - Q** Jos paketti (data / kuittaus) kuitenkin vain viivästyy eikä olekaan kadonnut
 Syntyy duplikaatti, joka havaitaan sanomameroinnin avulla
 Kuittauksessa mukana kuitatun paketin numero
- Tietoliikenteen perusteet /2007/ Liisa Marttinen 38



rdt3.0

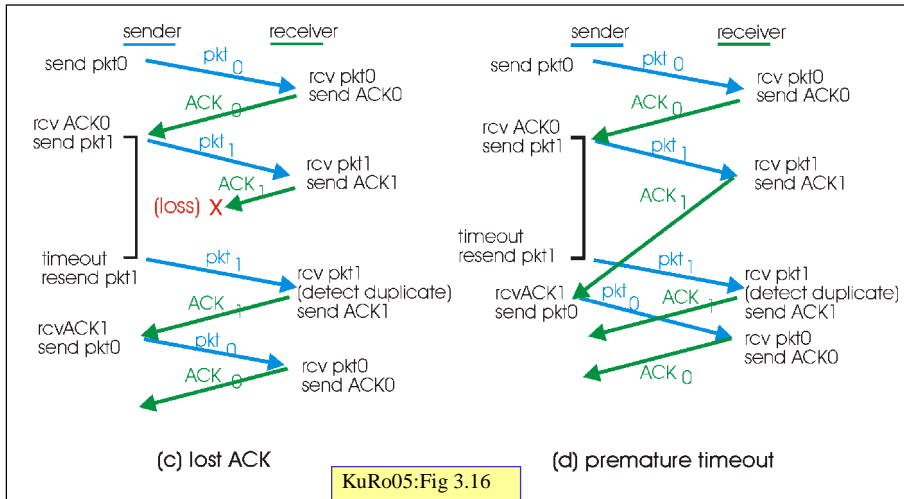


rdt3.0 toiminnassa





rdt3.0 toiminnassa



rdt3.0: Tehokkuus?

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

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

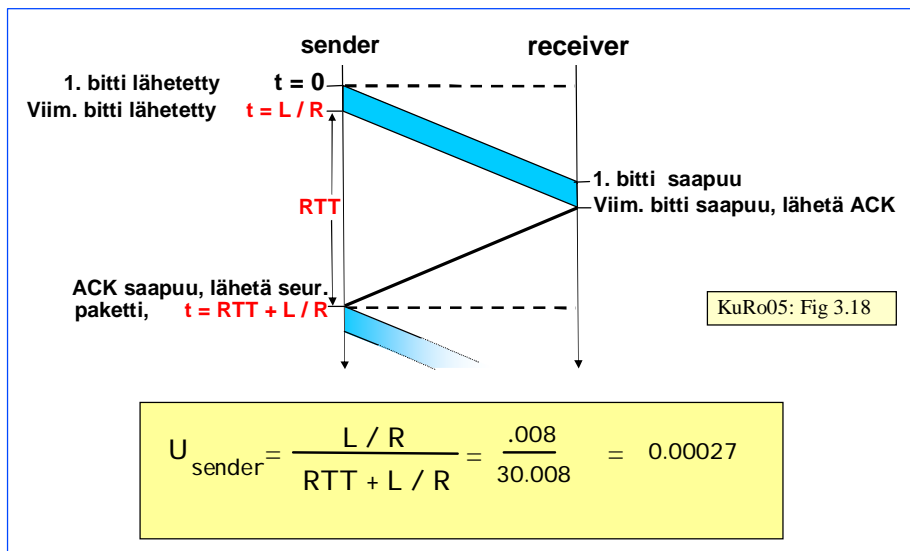
$$U_{\text{sender}} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

Käyttöaste (utilization): se osa kokonaisajasta, jolloin lähettäjä lähettää

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



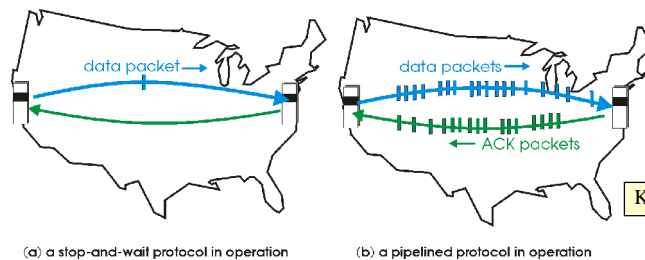
rdt3.0: stop-and-wait tehokkuus



rdtX.X: Liukuhinnaprotokollat

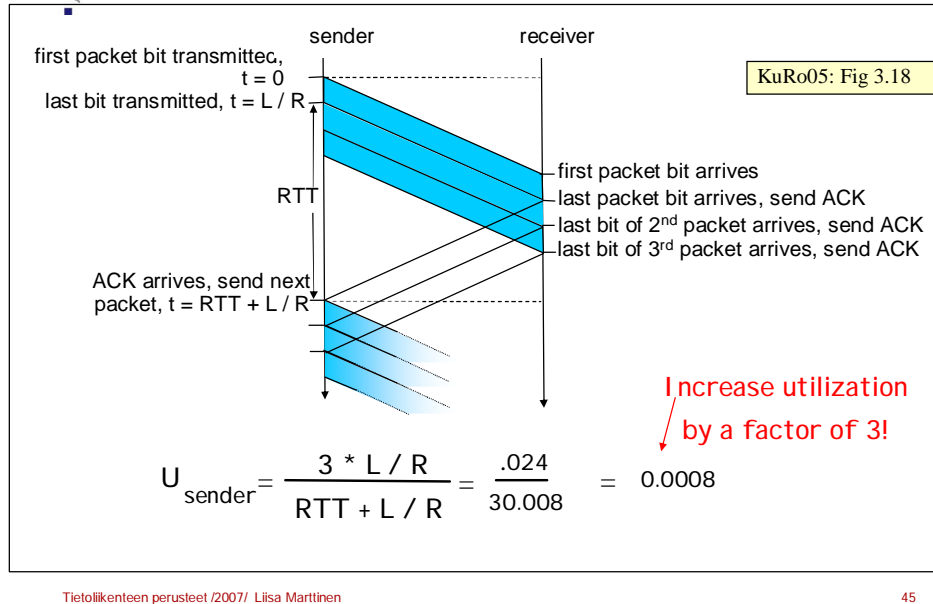
Lähettäjä saa lähettää useita paketteja, vaikka ei ole saanut kuittauksia edeltäviin

- n Numerointi (0,1) ei enää riitä, lisää numeroita tarvitaan
- n Tarvitaan puskurointia molemmissa päissä





Liukuhihnoitus: käyttöasteen kasvattaminen



Liukuva ikkuna (sliding window)

■ Ikkuna = tietyllä hetkellä sallitut pakettinumerot

- Riippuu yhteyden tyypistä ja puskuroinnista
- Rajallinen kenttä pakettinumerolle
- Vuonvalvonta: vaihteleva ikkunan koko N

■ Lähetysikkuna

- Ikkunan koko = montako pakettia saa olla kuittaamatta
- Mitkä pakettinumerot on käytetty, mutta kuittaamatta
- Mitä pakettinumeroita voi vielä käyttää

■ Lähettäjän on odotettava, jos kaikki numerot on käytetty

■ Kun kuittaus saapuu, ikkuna liikuu

- Seuraavat numerot tulevat luvallisiksi



Liukuva ikkuna

n Vastaanottoikkuna

- n Mitkä pakettinumerot otettu vastaan, mutta kuittaamatta
- n Mitä pakettinumeroita lähettäjä saa vielä käyttää

n Jos saadussa paketissa on ikkunan viimeinen numero

- n Ikkuna pysäyttää pakettien lähetyksen vastapäätä
- n Ikkuna estää uusien pakettien vastaanoton

n Paketin kuittaus liu'uttaa myös vastaanottajan ikkunaa

- n Hyväksytään uusia pakettinumeroita



Kun ikkunan koko on 1

n Vain yksi paketti kuittaamattomana

- n One Bit Sliding Window -protokolla
- n = stop-and-wait -protokolla

n Pakettinumerot 0 ja 1 riittävät

n ACK ilmoittaa

- n Joko seuraavaksi odotetun paketin numeron
- n Tai viimeksi vastaanotetun virheettömän paketin numeron

n ACK sisältää paketin numeron

- n Kuittausdublikaatti ei voi kuitata väärää paketteja



Virhetilanteen käsittely

- n Entä, kun huomataan virhe?
 - n Monta muuta pakettia jo matkalla
- n Pakettiin ei tule kuittausta
 - n Paketti katosi tai virheellinen
 - n Kuittaus katosi tai virheellinen
 - n => Ajastin laukeaa aikanaan
- n "Go-Back-N" (paluu N:ään)
 - n Paketit uudelleenlähetetään virheellisestä lähtien
- n Selective Repeat (Valikoiva toisto)
 - n Lähetetään vain virheelliset paketit



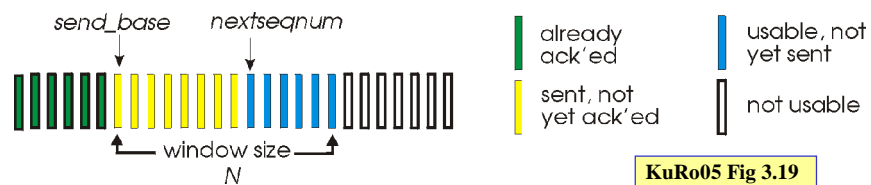
Go-Back-N

- n Vastaanottaja hyväksyy paketit vain järjestyksessä
 - n Kuittaa järjestyksessä tulleen virheettömän paketin
 - n Hylkää kaikki puuttuvan tai virheellisen paketin jälkeiset paketit eikä lähetä niistä kuittauksia
- n Kun lähettäjä ei saa pakettiin kuittausta
 - n Lähetyssikkuna täyttyy ja estää uusien pakettien lähettämisen
 - n Lähettäjän ajastimet laukeavat
 - n Lähettäjä lähettää uudestaan kaikki viimeisen kuittauksen jälkeiset paketit
 - n Näiden kuittaukset siirtävät taas lähetyssikkunaa
- n **Tehoton, jos paljon virheitä ja iso ikkkuna**



Go-Back-N

- Parannus: nopea reagointi puuttuvaan
- 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
 - Tuplakuitaus (duplicate ACK)

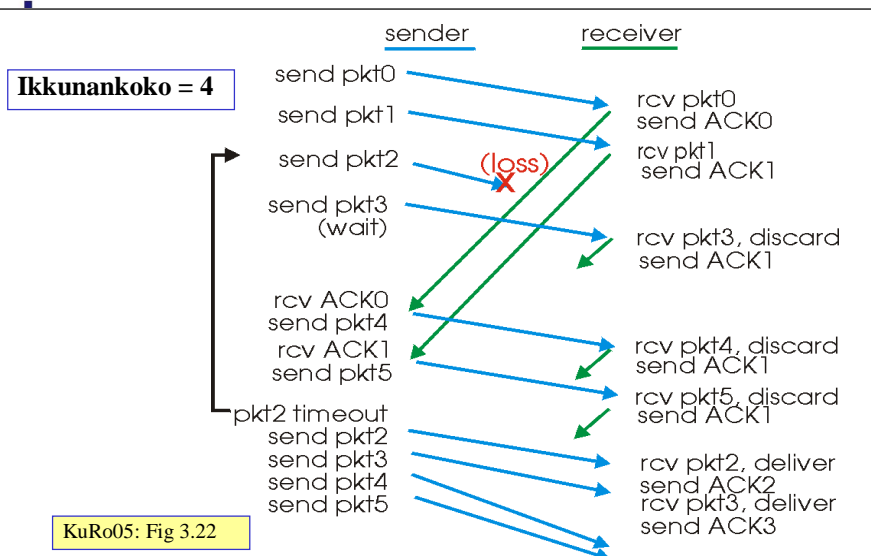


Tietoliikenteen perusteet /2007/ Liisa Marttinen

51



Go-Back-N: Esimerkki



Tietoliikenteen perusteet /2007/ Liisa Marttinen

52

Go-Back-N: lähettäjän tilakaavio

Lähetyspyyntö sovellukselta: lähetetään, jos ikkuna sallii

rdt_send(data)

```
if (nextseqnum < base+N) {
  sndpkt[nextseqnum] = make_pkt(nextseqnum,data,chksum)
  udt_send(sndpkt[nextseqnum])
  if (base == nextseqnum)
    start_timer
  nextseqnum++
}
else refuse_data(data)
```

Tässä vain yksi ajastin!

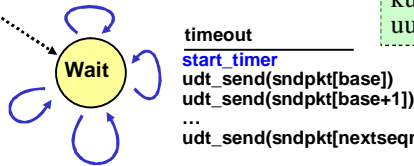
alkuarvot

Δ
base=1
nextseqnum=1

Ajastin laukeaa, lähetä kaikki kuittaamattomat uudestaan

rdt_rcv(rcvpkt) && corrupt(rcvpkt)
 Δ

Korruptoitunut kuittaus hylätään



rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
base = getacknum(rcvpkt)+1
if (base == nextseqnum)
stop_timer
else start_timer

Kuittaus siirtää ikkunaa

KuRo05: Fig 3.20

Sender

Tietoliikenteen perusteet /2007/ Liisa Marttinen

53

Go-Back-N: Vastaanottajan tilakaavio

receiver

rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && hasseqnum(rcvpkt,expectedseqnum)

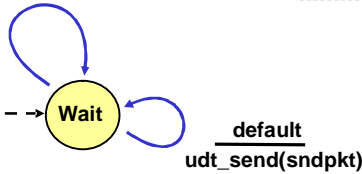
Paketti ok ja odotettu numero

extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pkt(expectedseqnum,ACK,chksum)
udt_send(sndpkt)
expectedseqnum++

Data sovellukselle, kuittaus lähettäjälle
Seuraava numero++

Δ

expectedseqnum=1
sndpkt = make_pkt(expectedseqnum,ACK,chksum)



Muuten sama kuittaus uudestaan

alkuarvot

KuRo05: Fig 3.21

Tietoliikenteen perusteet /2007/ Liisa Marttinen

54



Valikoiva toisto (Selective Repeat)

n Valikoiva uudelleenlähetyks

- n Lähetä uudelleen vain virheellinen /puuttuva paketti

n Kuittaus jokaiselle kelvolliselle paketille

n Paketit sovellukselle oikeassa järjestyksessä

- n Vastaanottajalla oltava puskuritilaa pakettien järjestämiseen

n Jos lähettäjä ei saa kuittausta paketista

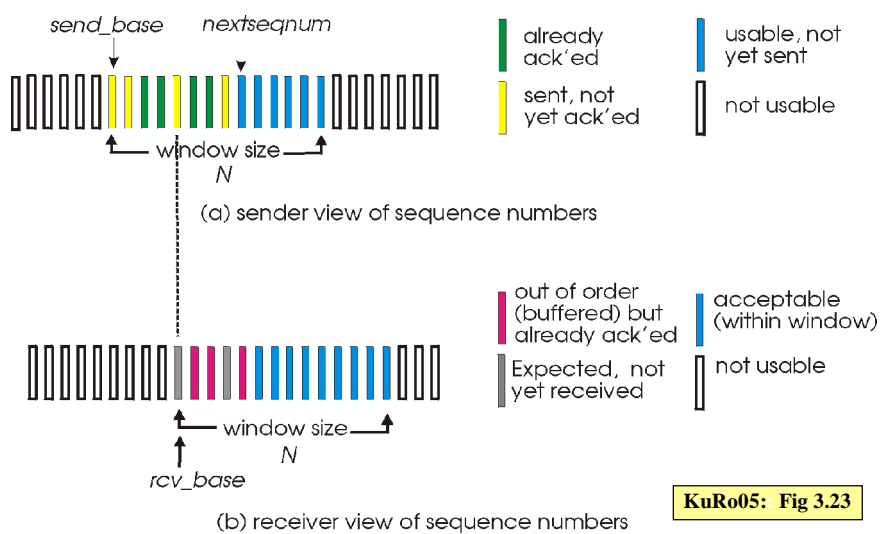
- n Lähetyssikkunan täytyminen pysäyttää lähettämisen
- n Aikanaan ajastin laukeaa ja aiheuttaa uudelleenlähetyksen
- n Jokaisella paketilla on oma ajastin

n Ikkuna liukuu nytkin tasaisesti

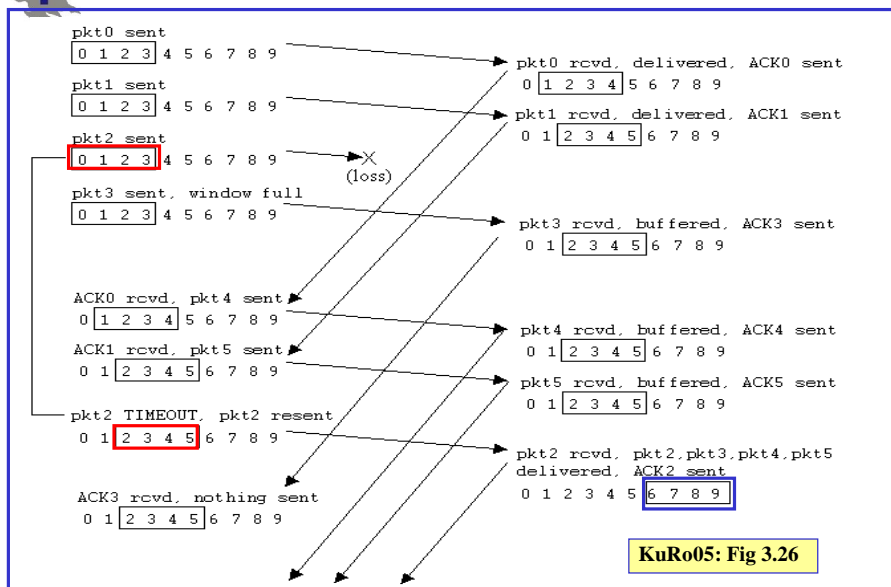
- n Yksi puuttuva kuittaus voi pysäyttää lähetyksen
- n Kun puuttuva paketti saatu, ikkuna liukuu kaikkien kuitattujen yli



Valikoiva toisto



Esimerkki



Tietoliikenteen perusteet /2007/ Liisa Marttinen

57



Ikkunankoko

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

Go-Back-n: n

Valikoiva toisto: $(n+1)/2$

Tietoliikenteen perusteet /2007/ Liisa Marttinen

58



Yhteenvedo menetelmistä

n Ks. KuRo05 Table 3.1

n Tarkistussumma

n Ajastin

n Järjestysnumero

n Kuittaukset

n Positiiviset ACK

n Negatiiviset NAK

n Ikkunat, liukuhihnoitus

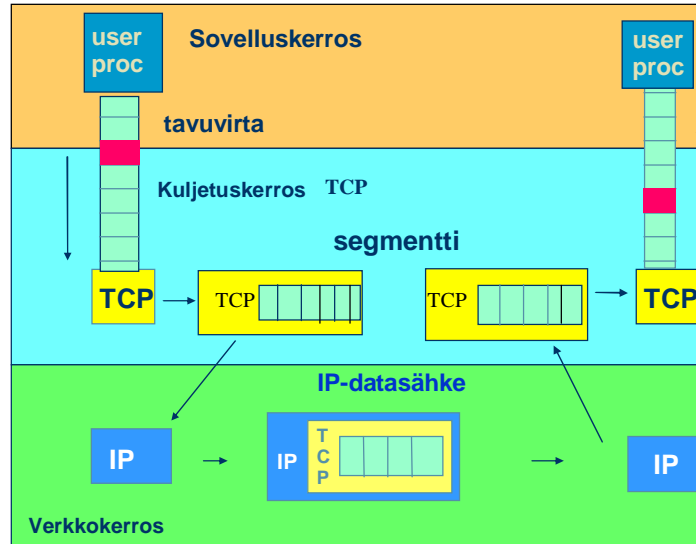


Kuljetuskerros

Yhteydellinen kuljetuspalvelu TCP

FRC 793, RFC 1122,
RFG 1323, RFC 2018,
FRC 2581

TCP: prosessilta prosessille -tavuvirta



Tietoliikenteen perusteet /2007/ Liisa Marttinen

61

TCP-protokolla

- n **Päästä-päähän kuljetuspalvelu**
 - n Yksi lähettäjä, yksi vastaanottaja
 - n Reitittimet eivät ole kiinnostuneita kuljetustason protokollasta
- n **Yhteydellinen (connection-oriented)**
 - n Yhteydenmuodostus
 - n Isäntäkoneissa: puskuritila, ikkunakoko, tavunumerointi
 - n Yhteyden purku
- n **Kaksisuuntainen (full duplex)**
 - n Yksi yhteys, jossa yhtä aikaa liikennettä molempiin suuntiin
- n **Luotettava, järjestyksen säilyttävä tavuvirta**
 - n Ei sanomaraajoja
 - n Tavunumerointi
 - n Kumulatiiviset kuittaukset

Tietoliikenteen perusteet /2007/ Liisa Marttinen

62



TCP-protokolla

- n Vuonvalvonta, ruuhkanhallinta (-valvonta)
 - n Lähettäjä ei voi tukahduttaa vastaanottajaa eikä reitittämiä
- n Liukuvan ikkunan protokolla
 - n Vuonvalvonta ja ruuhkanhallinta vaikuttavat ikkunankokoon
- n Puskurointia molemmissa päissä
 - n Uudelleenlähetystä varten
 - n Jotta saadaan annettua sovellukselle järjestyksessä

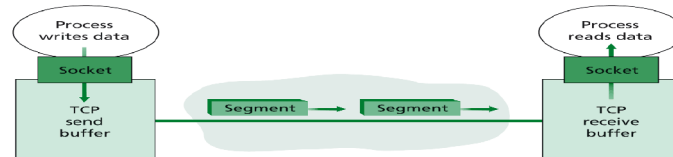


Figure 3.28 ♦ TCP send and receive buffers

Tietoliikenteen perusteet /2007/ Liisa Marttinen

63



TCP-protokolla

- n Segmentillä maksimikoko
 - n MSS (maximum segment size) = paljonko dataa segmentissä
- n Varmistaa, että tässä koneessa ei tarvita lisäpilkkomista paketeiksi
- n Linkkikerroksen fyysiset ominaisuudet vaikuttavat MSS:N arvoon
 - n MTU (maximum transfer unit)
 - n Ethernet MTU = 1500 => MSS = 1460, sillä TCP:n ja IP:n osoitteet (20 +20 tavua) vievät oman tilansa

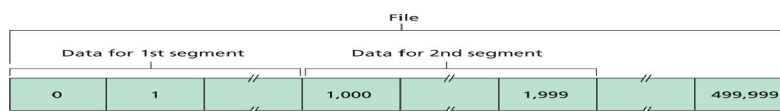


Figure 3.30 ♦ Dividing file data into TCP segments

Tietoliikenteen perusteet /2007/ Liisa Marttinen

64

TCP-segmentti

Otsake aina vähintään 20 B
Options-osa tarvittaessa

Segmentti- ja kuittaus-
numerot tavunumeroina

Ikkunankoko: paljonko tilaa
vastaanottopuskurissa (tavua)

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

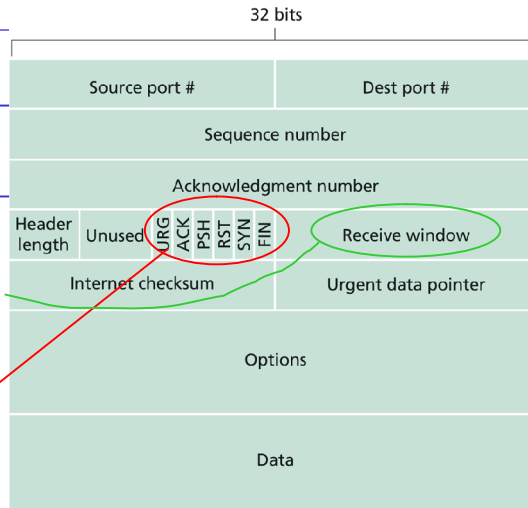


Figure 3.29 ♦ TCP segment structure

Tietoliikenteen perusteet /2007/ Liisa Marttinen

65

Tavunumerointi

Tavuvirtaa ...

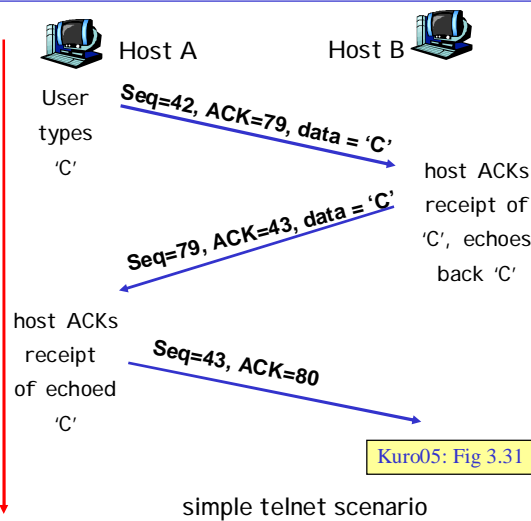
Segmentit voivat olla
erikokoisia

Segmentin 'numero' =

- ensimmäisen tavun numero
- alkuarvot sovitaan yhteyttä muodostettaessa

Kuittaus

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



Kuro05: Fig 3.31

simple telnet scenario

Tietoliikenteen perusteet /2007/ Liisa Marttinen

66



TCP: lähetys (simplified)

NextSeqNum = InitialSeqNum; SendBase = InitialSeqNum

```
loop (forever) {
  switch(event)
```

Vuonvalvonta ja/tai ruuhkanhallinta voi estää lähettämisen!

```
  event: data received from application above
  create TCP segment with sequence number NextSeqNum
  if (timer currently not running) start timer
  pass segment to IP
  NextSeqNum = NextSeqNum + length(data)
```

Riittääkö 1 segmentti?

```
  event: timer timeout
  retransmit not-yet-acknowledged segment with
  smallest sequence number
  start timer
```

Ajastimen arvo?

Yksi vai monta?

```
  event: ACK received, with ACK field value of y
  if (y > SendBase) {
    SendBase = y
    if (there are currently not-yet-acknowledged segments)
      start timer
  }
```

Toistokuittaukset?

```
} /* end of loop forever */
```

Kuro05: Fig 3.33

Tietoliikenteen perusteet /2007/ Liisa Marttinen

Comment:

- SendBase-1: last cumulatively ack'ed byte

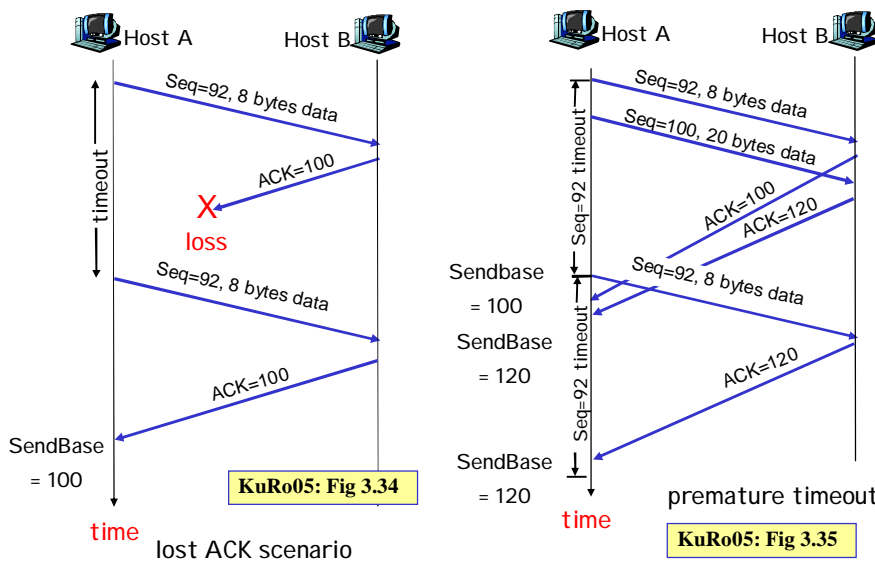
Example:

- SendBase-1 = 71; y= 73, so the rcvr wants 73+ ; y > SendBase, so that new data is acked

67



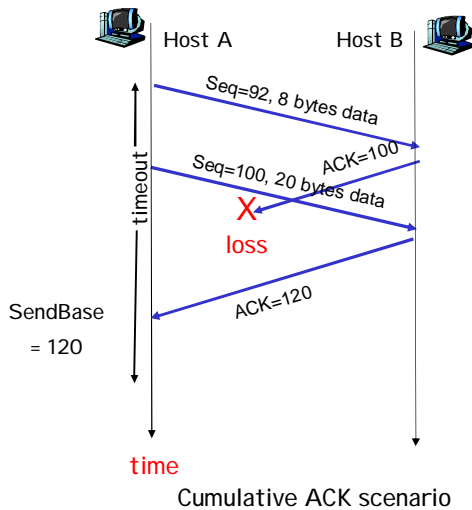
TCP: uudelleenlähetyks



Tietoliikenteen perusteet /2007/ Liisa Marttinen

68

TCP: uudelleenlähetys (2)



KuRo05: Fig 3.36

TCP ACK generation [RFC 1122, RFC 2581]

Event at Receiver

TCP Receiver action

Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed

Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK

Arrival of in-order segment with expected seq #. One other segment has ACK pending

Immediately send single cumulative ACK, ACKing both in-order segments

Arrival of out-of-order segment higher-than-expected seq. # . Gap detected

Immediately send *duplicate ACK*, indicating seq. # of next expected byte

Arrival of segment that partially or completely fills gap

Immediate send ACK, provided that segment starts at lower end of gap

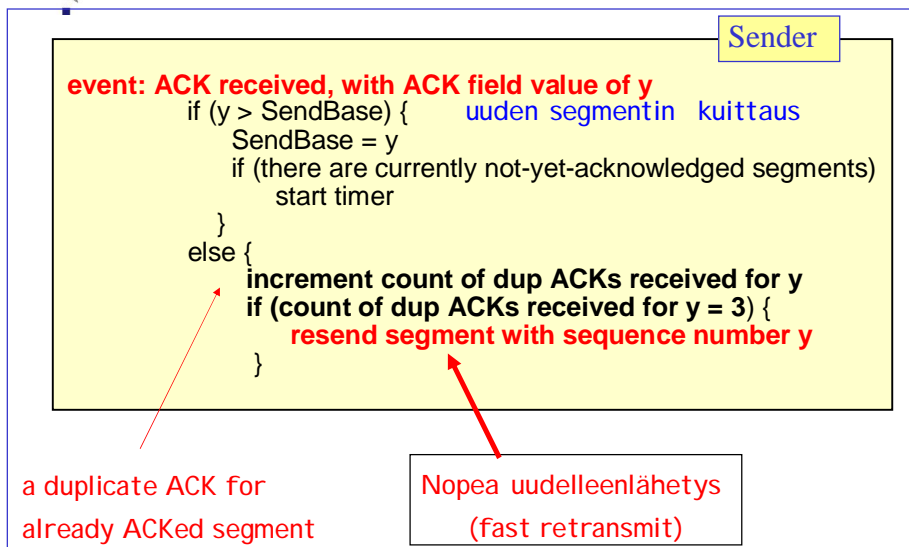


Nopea uudelleenlähetys (fast retransmit)

- n Timeout suhteellisen pitkä
 - n => aika iso viive ennen uudelleenlähetystä
- n Vastaanottaja ilmoittaa puuttuvasta segmentistä toistokuittauksilla (duplicate ACK)
 - n Liukuhinoituksen vuoksi useita segmenttejä voi olla kuittaamatta
 - n Jos välistä puuttuu segmentti, seurauksena on useita ACK-kuittauksia
- n Jos lähettäjä saa 3 samaa segmenttiä kuittaavaa **toistokuittaus**, se olettaa, että seuraava segmentti on kadonnut
 - n Ja lähettää puuttuvan segmentin heti
- n Nopea uudelleenlähetys = lähetä uudelleen jo ennen kuin ajastin laukeaa eli kolmen tuplakuittauksen jälkeen.



Nopea uudelleenlähetys





Paluu n:ään vai valikoiva toisto?

n Kumpaa TCP käyttää?

- n Liukuvan ikkunan protokolla
- n Hybridi, tavallaan 'best-of-both'

n Go-Back-N-tyyppinen

- n Virheellisiä tai väärässä järjestyksessä tulleita ei hyväksytä, mutta ne yleensä talletetaan puskuriiin
 - Kumulatiivinen ACK
 - Kaikkia virheellisestä lähtien ei tarvitse lähettää uudestaan

n Valikoiva toisto

- n Kumulatiivinen ACK ei kelpaa
- n SACK-kuittaus (selective ACK), joka kertoo, mitkä segmentit vastaanotettu (RFC 2018), ei yleisesti käytössä



Vuonvalvonta

n Jotta lähettäjä ei tukahduta vastaanottoa

- n Siirtonopeus sovitettava vastaanottavan sovelluksen mukaan

n Kuittaus on irroitettu vuonvalvonnasta

n Liukuva ikkuna, koko vaihtelee

- n Kuittaus siirtää ikkunaa
- n Vuonvalvonta määrää ikkunankoon
- n Kun ikkunankoko = 0, ei saa lähettää!

n Vastaanottaja kertoo, montako tavua puskuureihin vielä mahtuu

- n TCP-segmentin otsakkeen kenttä **Receive window**
- n Sovellus lukee tavut silloin kun haluaa
- n Koko on mukana jokaisessa TCP-segmentissä (molempiin suuntiin)

n Myös ruuhkanhallinta rajoittaa lähettämistä

Vuonvalvonta

- n Kun ikkunankoko ==0, milloin voi taas lähettää?
- n Jatka lähettämällä tavun kokoisia segmenttejä
 - n Kysely
- n Kuittaus antaa ajantasalla olevan tiedon vastaanottajan puskuritulasta
 - n Edellisen ACK:n toistokuittaus => ei tilaa
 - n Normaali ACK, kun vapaata vähintään täydelle TCP-segmentille
- n Miksi lähettäjä ei vain odota, että vastaanottaja kertoo, kun tilaa jälleen tulee
 - n Entä, jos tämä kuittaus katoaa!
 - n Lähettäjä odottaa turhaan ja vastaanottaja luulee, ettei sillä ole lähetettävää, lukkiutuminen!

Tietoliikenteen perusteet /2007/ Liisa Marttinen

75

Yhteyden muodostus

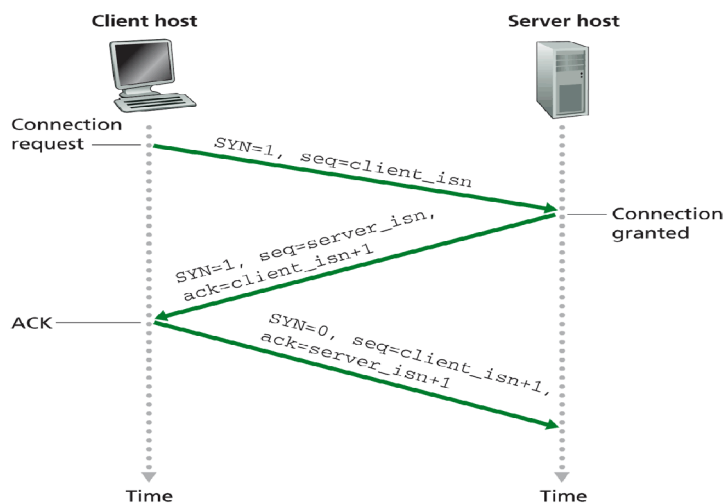


Figure 3.38 ♦ TCP three-way handshake: segment exchange

Tietoliikenteen perusteet /2007/ Liisa Marttinen

76

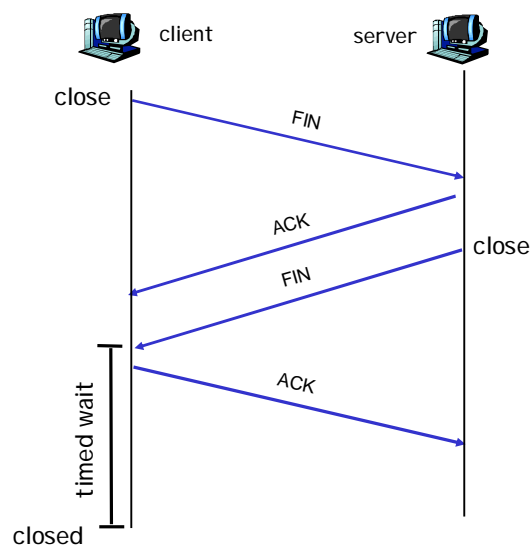


Yhteyden muodostus

- n **Kolmivaiheinen kättely** (three-way handshake)
 - n 3 segmenttiä: SYN – SYNACK – SYNACK
 - n Otsakkeen bittikentät
 - n Viimeinen voi sisältää dataa (piggyback)
 - n Jos porttiin ei liity prosessia, vastaukseksi RST-segmentti eli yhteyttä ei voida muodostaa
- n Varaa puskuritilaa
 - n Lähettäjä puskuroid uudelleenlähetystä varten
 - n Vastaanottaja saatujen pakettien järjestämistä varten
- n Sovitaan tavunumeroinnin alkuarvoista
 - n Kuittauksia vasten
- n Ilmoita oma vastaanottoikkunan koko
 - n Vuonvalvontaa varten



Yhteyden purku





Yhteyden purku

- n Molemmat suunnat puretaan erikseen
 - n 4 segmenttiä: FIN – ACK, FIN – ACK
- n Yhteys on kokonaan purettu, kun molemmat suunnat purettu
- n Purku käyttää ajastimia
 - n Joidenkin erikoistilanteiden hallintaan
 - n Noin 2 * paketin maksimaalinen elinikä
 - n Tyypillisesti 30, 60 tai 120 s



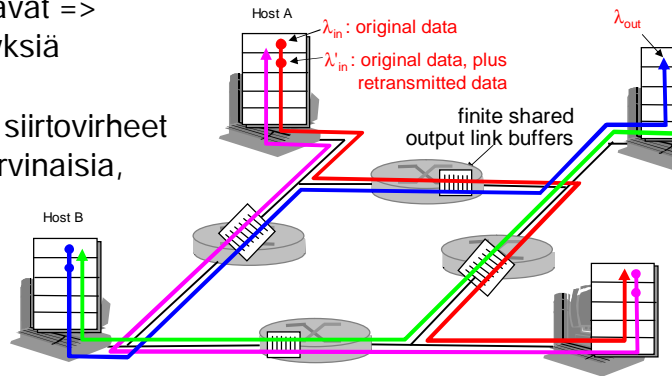
Kuljetuskerros

Ruuhkanhallinta TCP:ssä



Ruuhka

- n Reitittimelle tulee paketteja nopeammin kuin se ehtii välittää niitä eteenpäin
 - n Pitkiä jonotusviipeitä
 - n Pakettien hävittämistä (puskuritila loppuu)
- n Ajastimet laukeavat => uudelleenlähetystä
- n Lankaverkoissa siirtovirheet ovat nykyisin harvinaisia, paketin katoamisen syynä lähes aina ruuhka



Tietoliikenteen perusteet /2007/ Liisa Marttinen

81



Miten ratkaista ongelma?

- n Lähettäjän on hidastettava vauhtia
- n Verkkoavusteinen ruuhkanvalvonta (network assisted congestion control)
 - n Reitittimet antavat tietoa ruuhkasta
 - n Lisäbitit kertomassa ruuhkasta tai kenttä, joka ilmoittaa yhdeydelle sallitun lähetyksenopeuden (explicit rate)
- n Päästä-päähän ruuhkanvalvonta (end-to-end congestion control)
 - n Reitittimet eivät kerro ruuhkaantumisesta isäntäkoneille
 - n Isäntäkoneet huomaavat itse ruuhkan lisääntyneestä pakettien katoamisesta ja uudelleenlähetyksistä
 - n TCP käyttää tätä
- n Tällä kurssilla käsitellään vain TCP:n ruuhkanhallinta
 - n TCP Reno -algoritmi
 - n Ruuhkanhallintaan kehitetty runsaasti erilaisia algoritmeja

Tietoliikenteen perusteet /2007/ Liisa Marttinen

82



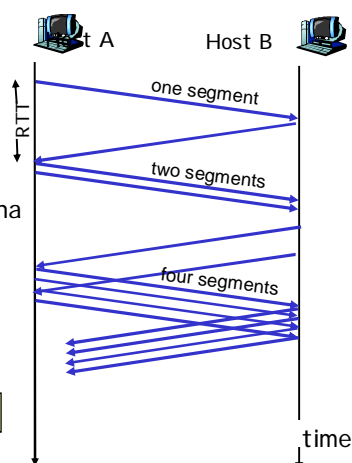
Ruuhkaikkuna (congestion window)

- n Internetin hetkellinen kuormitus vaihtelee
- n Ruuhkaikkuna
 - n Paljonko lähettäjä saa tietyllä hetkellä kuormittaa verkkoa
 - n Paljonko lähettäjällä saa olla kuittaamattomia segmenttejä
- n Lähettäjän pääteltävä itse sopiva ikkunankoko
 - n Jos uudelleenlähetyksajastin laukeaa, on ruuhkaa
 - n Jos kuittaukset tulevat tasaisesti, ei ole ruuhkaa
- n Dynaaminen ruuhkaikkunan koko
 - n Kasvata ikkunaa ensin nopeasti, kunnes törmätään ruuhkaan
 - n Pienennä sitten ikkunaa reilusti ja kasvata varovasti
- n Lähetyssikkunan raja voi tulla vastaan ensin
 - n Kuittamatta saa olla **min(lähetyssikkuna, ruuhkaikkuna)**



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

- n Aluksi ruuhkaikkuna = yksi segmentti
 - n Alussa hidas siirtonopeus = MSS/RTT
- n Kukin kuitaus kasvattaa yhdellä ruuhkaikkunan kokoa **hidas aloitus**
 - n Eksponentiaalinen kasvu
 - n Ikkuna kaksinkertaistuu yhden RTT:n aikana
- n Jos uudelleenlähetyks, puolita ruuhkaikkunan koko
 - n Multiplicative decrease
- n Sen jälkeen kasvata ikkunaa yksi segmentti/RTT **ruuhkanvälttely**
 - n Lineaarinen kasvu (Additive increase)
 - n Ruuhkan välttely (congestion avoidance)
- n Siirtonopeus = $CongWin / RTT$ tavua/sek





Kynnysarvo (threshold)

n = ~ **varoitus**: tästä lähtien syytä varoa ruuhkaa

n Aluksi threshold = 64 KB

n Ajastimen laukeamisen (timeout) jälkeen

n Threshold = CongWin / 2

n Timeout = 2 * timeout

n Kynnysarvoon saakka ikkunan kasvatus eksponentilaalista

n Yhtä kuitattua segmenttiä kohden saa lähettää kaksi uutta

n Eli kaksinkertaistuu yhden RTT:n aikana

n Sen jälkeen ikkuna kasvaa lineaarisesti

ruuhkanvälttely

n Kasvaa yhdellä yhden RTT:n aikana

n Miksi näin?



TCP Reno: Tarkennus

n **Saatu 3 ACK-kaksoiskuittausta** (double ACK)

n Verkko pystyy välittämään dataa!

n Ei siis (pahaa) ruuhkaa, ehkä paketissa bittivirhe tai paketti kadonnut jostain muusta syystä

n **Nopea uudelleenlähetys** (fast retransmit)

n **Nopea toipuminen** (fast recovery)

n Puolita ruuhkaikkunan koko ja kasvaa sitten lineaarisesti

n **Timeout (= uusi hidas aloitus)**

n Verkko pahasti ruuhkautunut!

n Pudota ikkunankoko arvoon 1

Hidas aloitus

n Kasvata eskponentiaalisesti kynnysarvoon asti

n Kasvata sitten lineaarisesti

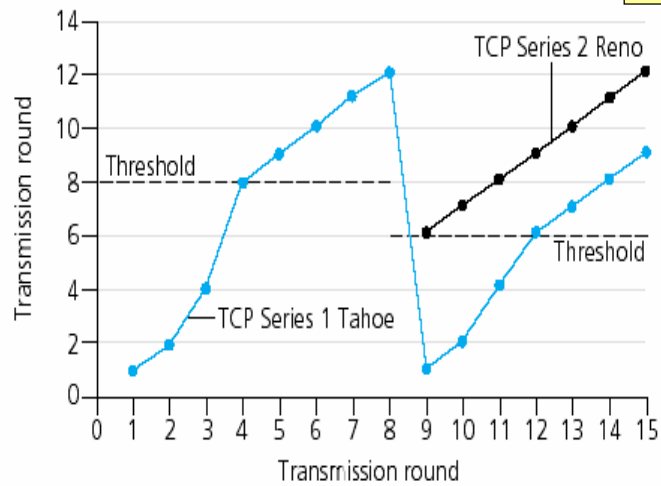
ruuhkanvälttely

Vanha TCP Tahoe pudotti aina kokoon 1.



TCP Tahoe vs. TCP Reno

KuRo05: Fig 3.51



Tietoliikenteen perusteet /2007/ Liisa Marttinen

87



TCP Reno Ruuhkanhallinta

State	Event	TCP Sender Action	Commentary
Slow Start (SS)	ACK receipt for previously unacked data	$CongWin = CongWin + MSS$, If ($CongWin > Threshold$) set state to "Congestion Avoidance"	Resulting in a doubling of CongWin every RTT
Congestion Avoidance (CA)	ACK receipt for previously unacked data	$CongWin = CongWin + MSS * (MSS / CongWin)$	Additive increase, resulting in increase of CongWin by 1 MSS every RTT
SS or CA	Loss event detected by triple duplicate ACK	$Threshold = CongWin / 2$, $CongWin = Threshold$, Set state to "Congestion Avoidance"	Fast recovery, implementing multiplicative decrease. CongWin will not drop below 1 MSS.
SS or CA	Timeout	$Threshold = CongWin / 2$, $CongWin = 1 MSS$, Set state to "Slow Start"	Enter slow start
SS or CA	Duplicate ACK	Increment duplicate ACK count for segment being acked	CongWin and Threshold not changed

Tietoliikenteen perusteet /2007/ Liisa Marttinen

88



Ajastimen arvo?

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

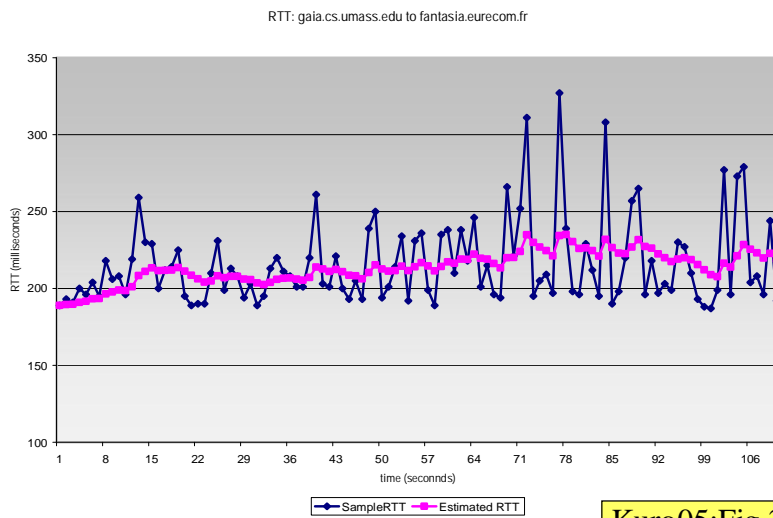


Ajastimen arvo?

- n $Timeoutinterval = EstimatedRTT + 4 DevRTT$
- n **Arvioi kiertoviive** eli EstimatedRTT
 - n Mittaa jokaisen lähetetyn segmentin kiertoviive tai noin RTT:n välein normaalien lähetyksen aikana.
 - n Laske painotettu arvo, tyypillisesti $\alpha = 1/8 = 0.125$
- n $EstimatedRTT = (1-\alpha) * EstimatedRTT + \alpha * SampleRTT$
- n **Huomioi poikkeama**
tyypillisesti $\beta = 0.25$
- n $DevRTT = (1-\beta) * DevRTT + \beta * |SampleRTT - EstimatedRTT|$



Esimerkki



Tietoliikenteen perusteet /2007/ Liisa Marttinen

91



Onko TCP reilu?

- n Jokainen reitittimessä kulkeva yhteys kärsii ruuhkasta
- n Vain TCP-yhteydet kiltisti vähentävät lähetystään
 - n AIMD: additive increase, multiplicative decrease
- n Sovellus voi avata monta rinnakkaista yhteyttä
 - n Onko tämä reilua muita kohtaan?
- n UDP?
 - n Ei ruuhkanhallintaa, ei välitä ruuhkasta eikä vähennä lähetystä
 - n Multimediasovellukset käyttävät UDP:tä: työntävät dataa vakionopeudella ja sietävät katoamisia
- n TCP-ystävällinen reititin?

Tietoliikenteen perusteet /2007/ Liisa Marttinen

92



Kertauskysymyksiä

- n TCP vs. UDP?
- n Miten tehdä kuljetuspalvelusta luotettava?
- n Keskeisimmät TCP:n otsakkeessa olevat tiedot?
- n Mitä tapahtuu TCP:n yhteydenmuodostuksessa?
- n Vuonvalvonta?
- n Ruuhkanhallinta?



ks. Kurssikirja s. 285