

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:

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

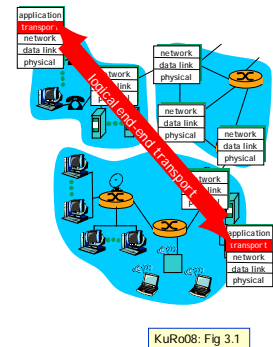


Kuljetuskerros

Kuljetuspalvelu

Kuljetuskerros

- n Tarjoaa kuljetuspalvelun prosessien välille
- n Vain isäntäkonelissa
Lähetys: Pilko sovelluskerroksen sanoma pienemmiksi **segmenteiksi**, jotka verkkokerros toimittaa perille.
Vastaanotto: Kokoaa segmentit sanomaksi, jonka sovellus lukee.
- n Verkkokerros reitittää koneesta koneelle
- n Segmentin koko s.e. verkkokerros pystyisi välttämään sellaisena



KuRo08: 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

Sovelluksen vaatimukset

Kuljetuskerros pyrkii palauttamaan verkkokerroksen puutteita

Verkkokerroksen palvelut

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")

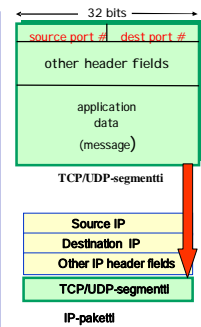
Mikä kone /Mikä prosessi?

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

Mikä kone /Mikä prosessi?

Lähetys (asiakas)

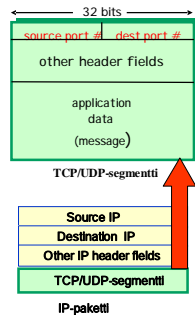
- Kuljetuskerros
 - Segmentin otsakkeessa lähde- ja kohdeprosessin porttinumero
 - Antaa segmentin verkkokerroksen välitettäväksi
 - TCP: huolehtii myös luotettavuudesta
 - UDP: tarjoaa pelkän välityspalvelun
- Verkkokerros
 - Paketin otsakkeessa lähde- ja kohdekoneen IP-osoite ja reitittimet osaaavat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



Mikä kone /Mikä prosessi?

Vastaanotto (palvelija)

- Verkkokerros
 - Vastaanottaa IP-paketin
 - Poistaa verkkokerroksen otsakkeet
 - Luovuttaa paketissa olleen segmentin kuljetuskerrokselle
- Kuljetuskerros
 - Poistaa kuljetuskerroksen otsakkeet
 - Kokoaa yhteenkuuluvat segmentit sanomiksi (tavuvirraksi)
 - Ohjaa sanoman (tavuvirran) oikealle prosessille (eli oikeaan pistokkeeseen) porttinumeron avulla
 - TCP: huolehtii myös luotettavuudesta
 - UDP: tarjoaa pelkän välityspalvelun



Kaksi www-asiakasta ja palvelija

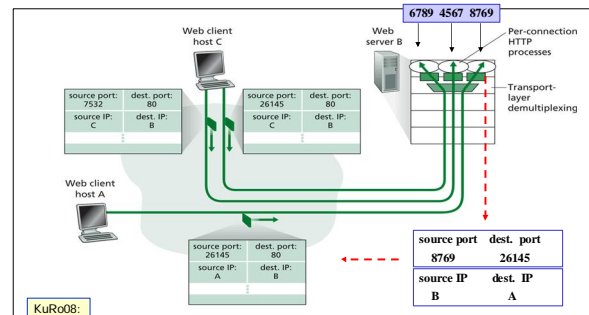


Figure 3.5 Two clients, using the same destination port number (80) to communicate with the same Web server application

Kuljetuskerros

Yhteydetön
kuljetuspalvelu
UDP

UDP (User Datagram Protocol) (RFC 768)

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



Käyttö

Vaikka UDP ei takaa luotettavuutta, se sopii silti monen sovelluksen tarpeisiin

Alkujaan oli vain TCP, UDP luotu myöhemmin

Miksi UDP?

- Pieni yleisrasite, koska pieni otsake
- Ei tilatietojen tallettamista eikä lähetys- ja vastaanottopuskureita
- Sovellus voi sietää virheitä
- Reaaliaikavaatimuksia: data lähtee heti verkkoon, koska ei yhteydenmuodostusta eikä vuon- tai ruuhkanvalvontaa

Tarvittava luotettavuus on räätälöitävissä sovellukseen

Sovellusprotokolla: oma sanonumerointi, uudelleenlähetys

Tietoliikenteen perusteet / 2008/ Liisa Marttinen

13



Kuljetusprotokolla TCP

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)

Lähetettäjä hilljentää vauhtia, jos vastaanottaja ei ehdi käsitellä

Ruuhkanvalvonta (congestion control)

Lähetettäjä hilljentää vauhtia, jos reitittimet eivät ehdi käsitellä

Tietoliikenteen perusteet / 2008/ Liisa Marttinen

14



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	UDP or TCP
Internet telephony	typically proprietary	UDP or TCP
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

Kuro08:

Miksi nämä sovellukset suosivat UDP:tä?

Tietoliikenteen perusteet / 2008/ Liisa Marttinen

15



UDP-segmentin rakenne

Porttinumerot

Koska on prosessien välinen palvelu

Length

Segmentin kokonaispituus otsake (8 B) mukaanluettuna

Checksum (optionaalinen)

Bitvirheen havaitsemiseen

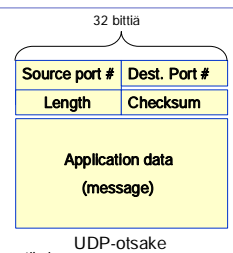
UDP ei yritä toipua, hävittää virheellisen segmentin

Data

Pitkä sanoma pilkottuna useaksi segmentiksi

IP-osoitteet vasta verkkokerroksen otsakkeessa

Näitä tarvitaan reitityksessä



UDP-otsake

Tietoliikenteen perusteet / 2008/ Liisa Marttinen

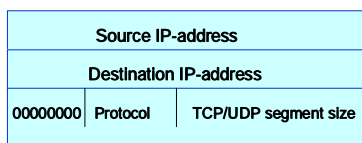
16



Kuljetuskerroksen pseudo-otsake

Käyttö vain isäntäkoneen sisäisesti. Ei lähetetä verkkoon.

- UDP laskee tarkistussumman otsakkeelle, datalle ja ns. pseudo-otsakkeelle, joka sisältää IP-otsakkeen tietoja
- Varmistus, että segmentti on tullut oikeaan koneeseen ja oikeaan porttiin



Tietoliikenteen perusteet / 2008/ Liisa Marttinen

17



UDP-tarkistussumma

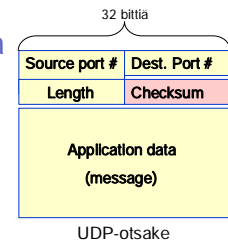
Lähetys

Summaa 16 bitin kokonaisuudet (otsake + pseudo-otsake mukana), ylivuotobitit lasketaan mukaan, talleta yhden komplementtina

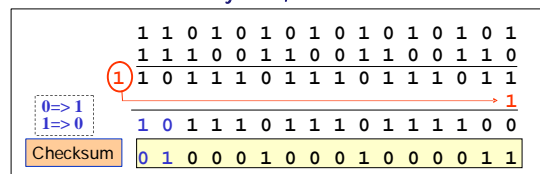
Vastaanotto

Summaa 16 b kokonaisuudet (myös tarkistussumma).

Jos tuloksena on 16 ykköstä, niin OK!



UDP-otsake



Tietoliikenteen perusteet / 2008/ Liisa Marttinen

18

Esimerkki

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

Lähettäjä:	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
-----	-----	-----
①1011 0110	11111 1110	1 0111 1110
-----	-----	-----
1011 0111	1111 1111	0111 1111
-----	-----	-----
0100 1000	OKI	Virhe!

Yhden komplementti

Miksi UDP-tarkistussumma

- Kaikki linkkikerrokset eivät suorita tarkistuksia!
 - Ethernet huolehtii kyllä
- UDP-tarkistussumma ei ole kovin tehokas havaitsemaan virheitä!
- UDP ei yritä toipua virheistä!
 - Jotkut toteutukset voivat tuhota virheellisen segmentin
 - Jotkut antavat sen sovellukselle varoituksen kera
- Lisärasite?
 - Ei tarvitse käyttää, jos ei halua. Tällöin lähettäjä laittaa tarkistussummaksi pelkkiä nollia

Tehtäviä:

- Lähetetään 10 tavun viesti UDP:llä.
 - Miten kauan kestää lähettäminen, jos lähetyksenopeus on 56 kbps?

$$10 \text{ tavua} + 8 \text{ tavua} = 18 * 8 \text{ b} = 144 \text{ bittia}$$

$$144 \text{ b} / 56 \text{ 000 b/s} = 2.57 \text{ ms}$$
 - Miten suuri on etenemisvirhe, jos etäisyys lähettäjältä vastaanottajalle on 1000 km?

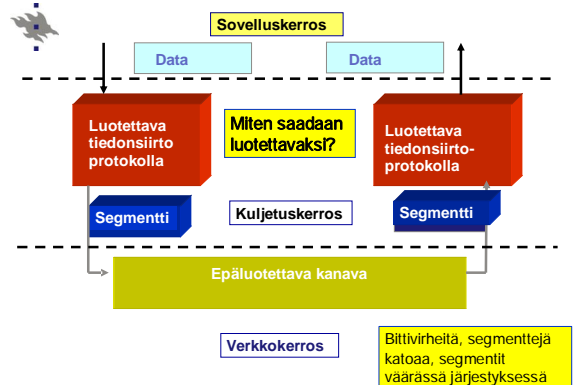
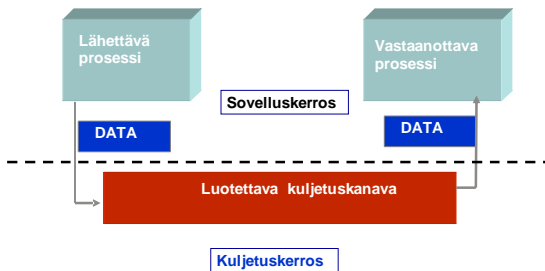
$$1000 \text{ km} / 200 \text{ 000 km/s} = 5 \text{ ms}$$
 - Miten suuri on UDP-otsakkeen aiheuttama yleisrasite (overhead)?

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

Kuljetuskerros

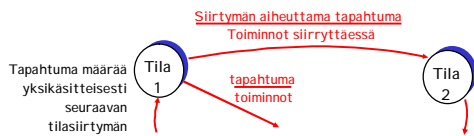
Luotettavan kuljetuspalvelun periaatteet

Luotettava tiedonsiirto



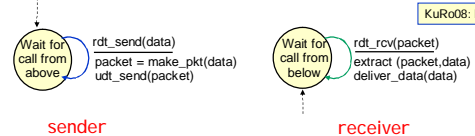
Kuinka saada luotettavaksi?

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



Versio rdt1.0: Ideaalitalanne

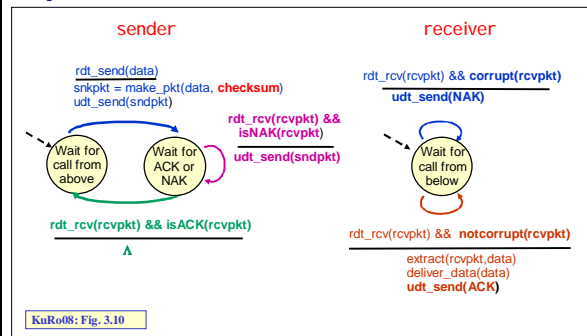
- Oletus: **siirtokanava on täysin luotettava**
 - Ei bittivirheitä, ei katoavia paketteja, ei epäjärjestystä
- Luotettava kuljetuspalvelu =
 - Lähettäjä kirjoittaa datan kanavaan,
 - Vastaanottaja lukee datan kanavasta



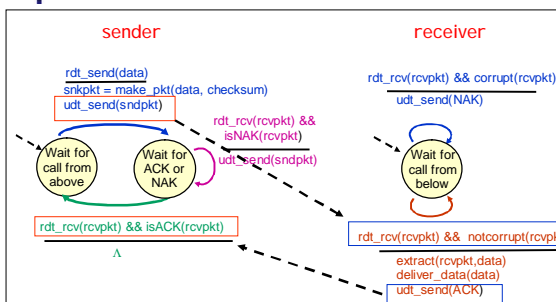
Versio rdt2.0: Bittivirheitä

- Oletus: Voi olla **vain bittivirheitä**
 - Bitti voi kääntyä siirron aikana
 - Siirto ei kadota paketteja**
- 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: **kultausanoma** (ACK / NAK)
 - Uudelleenlähetytys**: dataa puskuroitava
- Stop-and-wait-protokolla
 - Lähettäjä odottaa kuittausta ennenkuin lähettää seuraavan

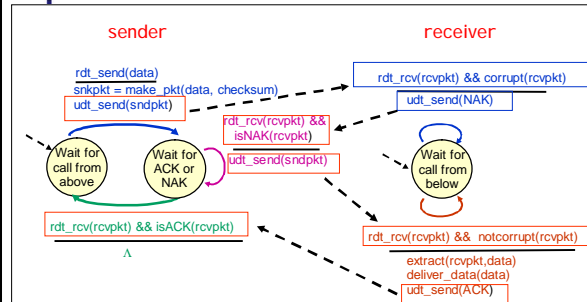
rdt2.0:



rdt2.0: Toiminta, kun ei ole virhettä



rdt2.0: Toiminta virheilänteessä



rdt2.0: Missä voi mennä väärin?

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

Versio rdt2.1: enemmän bittivirheitä

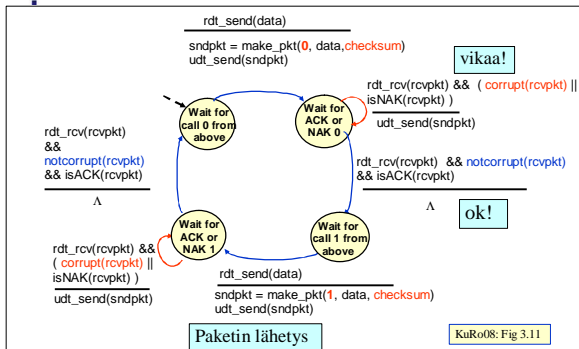
Lähetäjä:

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

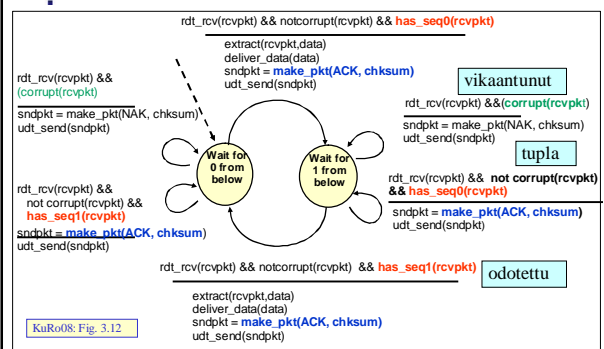
Vastaanottaja:

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

rdt2.1: Lähetäjän tilakaavio



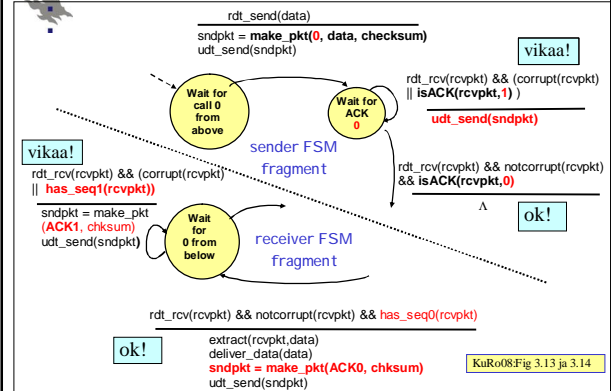
rdt2.1: vastaanottajan tilakaavio



Versio rdt2.2: vain ACK-kuitaus käytössä

- ┆ Sama toiminnallisuus kuin edellä
- ┆ Käyttää vain ACK-kuitauksia
 - ┆ Vastaanottaja kuittaa viimeksi kunnossa saamansa paketin
 - ┆ Kuittaukseseen on liitettävä kuitattavan paketin numero
- ┆ Jos samalle paketille (nro X) tulee useita ACK-kuitauksia (*duplicate ACK*), niin sitä seuraava paketti (nro X+1) joko puuttuu tai on virheellinen
 - ┆ -NAK-kuitaus
 - ┆ Lähetetään uudelleen sanoma X+1

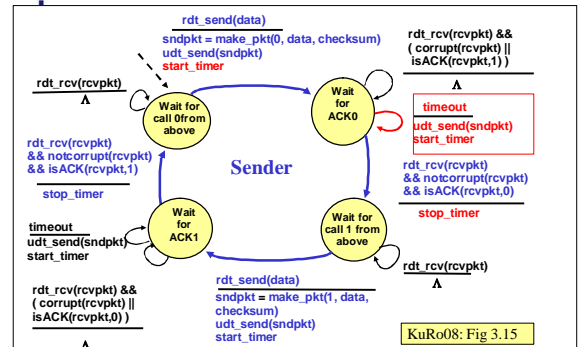
rdt2.2



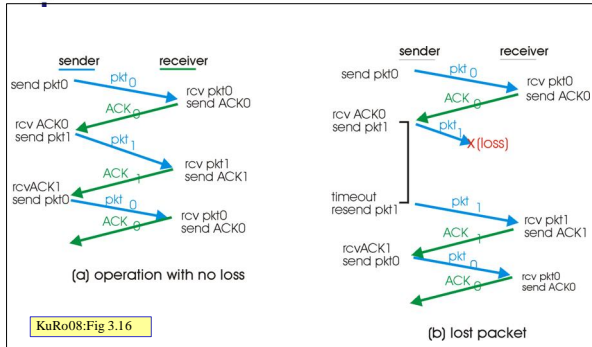
Versio rdt3.0: paketteja voi kadota!!

- Oletus: Siirtokanava voi kadottaa paketteja**
Sekä datapaketteja että kuittauspaketteja voi kadota.
Tarkistussumma, pakettinumero, ACK eivät vielä riitä! **Miksi?**
- Lähetäjä odottaa jonkin aikaa kuittausta
Jos ei saavu, lähettää paketin uudelleen
Ajastin laukaisee uudelleenlähteyksen
- Jos paketti (data / kuittaus) kuitenkin vain viivästyy eikä olekaan kadonnut
Syntyy duplikaatti, joka havaitaan sanonumeroinnin avulla
Kuittauksessa mukana kuitatun paketin numero

rdt3.0

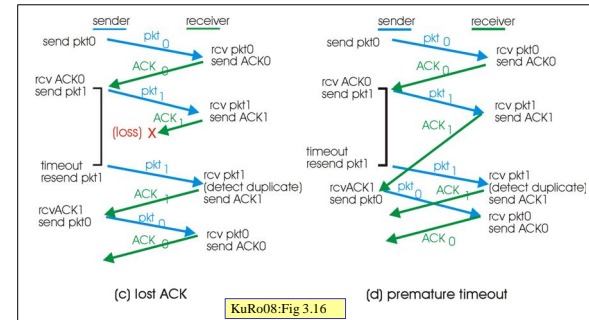


rdt3.0 toiminnassa



KuRo08:Fig 3.16

rdt3.0 toiminnassa



KuRo08:Fig 3.16

rdt3.0: Tehokkuus?

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

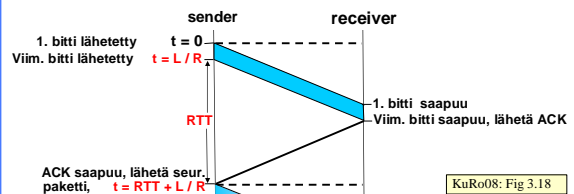
$$T_{\text{transmit}} = \frac{L \text{ (packet length in bits)}}{R \text{ (transmission rate, bps)}} = \frac{8\text{kb}/\text{pkt}}{10^{**}9 \text{ b/sec}} = 8 \text{ microsec}$$

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

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

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

rdt3.0: stop-and-wait tehokkuus



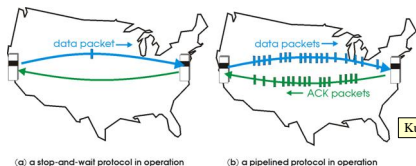
KuRo08: Fig 3.18

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

rdtX.X: Liukuhinnaprotokollat

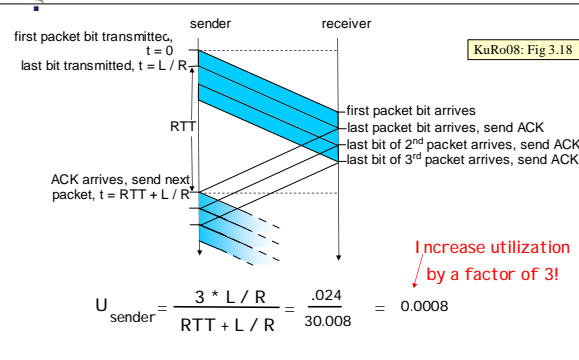
Lähettäjä saa lähettää useita paketteja, vaikka ei ole saanut kuittauksia edeltävin

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



KuRo08: Fig 3.17

Liukuhinnointi: käyttöasteen kasvattaminen



KuRo08: Fig 3.18

Liukuva ikkuna (sliding window)

n Säätelee pakettien lähettämistä ja vastaanottoa

- n Kerroo millä pakettinumerolla on lähetetty/vastaanotettu, mistä saatu/lähetetty kuittauksset ja millä numerolla voi vielä lähettää/vastaanottaa paketteja

n Ikkunan koko riippuu yhteyden tyypistä ja puskurien koosta

...10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 ...

n Lähetyssikkuna (sender window)

- n Ikkunan koko = montako pakettia saa olla kuittaamatta
- n Mitkä pakettinumerot on käytetty, mutta kuittaamatta
- n Mitä pakettinumeroita voi vielä käyttää
- n Lähettäjän on odotettava, jos kaikki ikkunan numerot on käytetty
- n Kun kuittaus saapuu, ikkuna liikuu
- n Seuraavat numerot tulevat luvallisiksi

Liukuva ikkuna (2)

n Vastaanottoikkuna (receiver window)

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

n Jos saadussa paketissa on ikkunan viimeinen numero

- n Ikkuna pysäyttää pakettien lähetyksen vastapästä
- 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 (esim. TCP)
- n Tai viimeksi vastaanotetun virheettömän paketin numeron

n ACK sisältää paketin numeron

- n Kuittausduplikaatti 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 (Valkoiva toisto)

- n Lähetetään vain virheettiset 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

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

Tehoton, jos paljon virheitä ja iso ikkuna

Tietoliikenteen perusteet /2008/ Liisa Marttinen

49

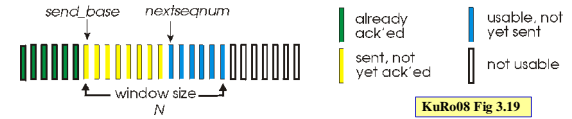
Go-Back-N

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

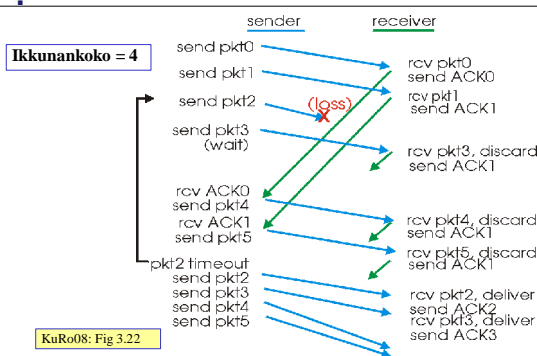


KuRo08 Fig 3.19

Tietoliikenteen perusteet /2008/ Liisa Marttinen

50

Go-Back-N: Esimerkki

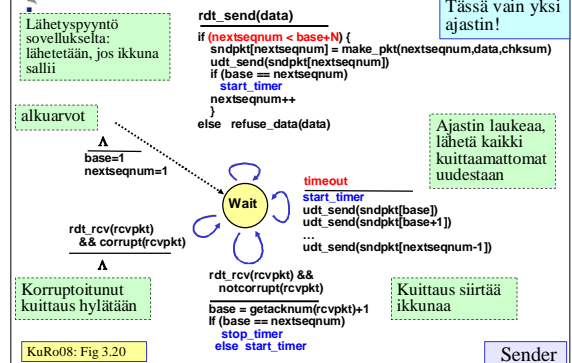


KuRo08: Fig 3.22

Tietoliikenteen perusteet /2008/ Liisa Marttinen

51

Go-Back-N: lähettäjän tilakaavio

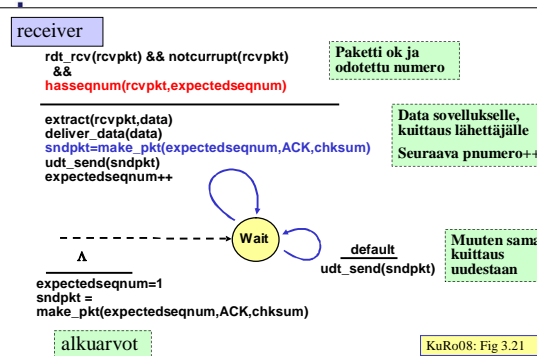


KuRo08: Fig 3.20

Tietoliikenteen perusteet /2008/ Liisa Marttinen

52

Go-Back-N: Vastaanottajan tilakaavio



KuRo08: Fig 3.21

Tietoliikenteen perusteet /2008/ Liisa Marttinen

53

Valikoiva toisto (Selective Repeat)

Valikoiva uudelleenlähetys

- Lähetä uudelleen vain virheellinen /puuttuva paketti

Kuittaus jokaiselle kelvolliselle paketille

Paketit sovellukselle oikeassa järjestyksessä

- Vastaanottajalla oltava puskuri tilaa pakettien järjestämiseen

Jos lähettäjä ei saa kuittausta paketista

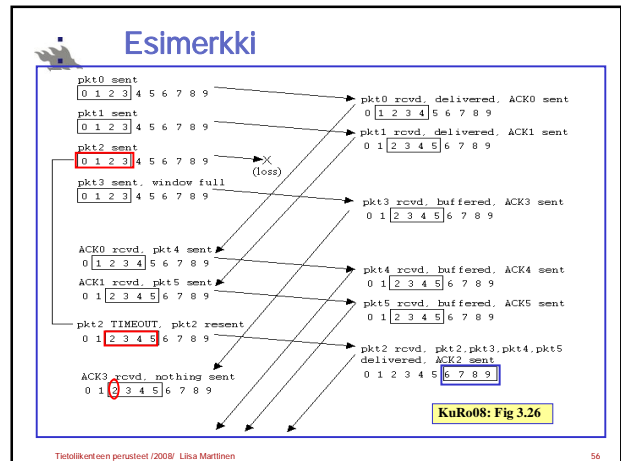
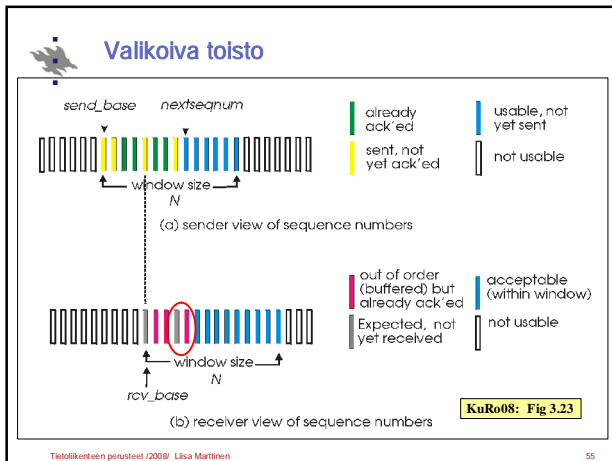
- Lähetysikkunan täytyminen pysäyttää lähettämisen
- Aikanaan ajastin laukeaa ja aiheuttaa uudelleenlähetysten
- Jokaisella paketilla on oma ajastin

Ikkuna liikuu nytkin tasaisesti

- Yksi puuttuva kuittaus voi pysäyttää lähetysten
- Kun puuttuva paketti saatu, ikkuna liikuu kaikkien kuittattujen yli

Tietoliikenteen perusteet /2008/ Liisa Marttinen

54



Ikkunankoko

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

Go-Back-n: n

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

MIKSI NÄIN?

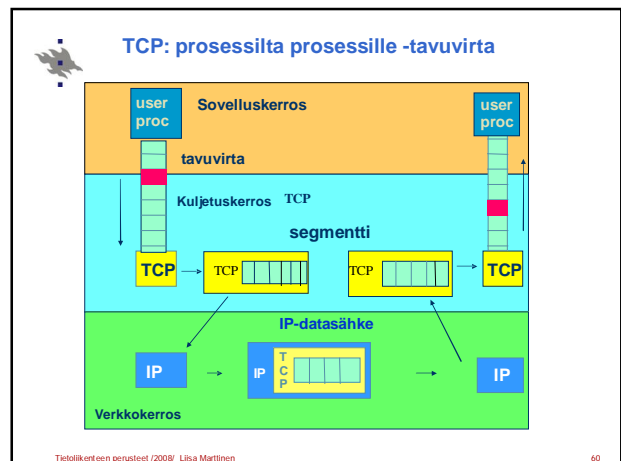
Harjoitustehtävinä!

- ### Yhteenveto menetelmistä
- Ks. KuRo08 Table 3.1
 - Tarkistussumma
 - Ajastin
 - Järjestysnumero
 - Kuittaukset
 - Positiiviset ACK, tuplakuittaukset
 - Negatiiviset NAK
 - Ikkunat, liukuihnoitus

Kuljetuskerros

Yhteydellinen kuljetuspalvelu TCP

RFC 793, RFC 1122, RFC 1323, RFC 2018, RFC 2581



TCP-protokolla

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

TCP-protokolla

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

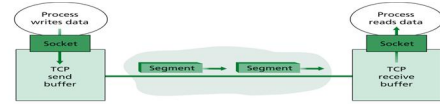


Figure 3.28 TCP send and receive buffers

TCP-protokolla

- Segmentillä maksimikoko**
 - MSS (maximum segment size) = paljonko dataa segmentissä
- Varmistaa, että tässä koneessa ei tarvita lisäpiikkomista paketeiksi**
- Linkkierroksen fyysiset ominaisuudet vaikuttavat MSS:n arvoon**
 - MTU (maximum transfer unit)
 - 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

TCP-segmentti

Otsake aina vähintään 20 B
Optio-osa tarvittaessa

Segmentti- ja kuittausnumerot **tavunumeroina**

Ikkunankoko: paljonko tilaa vastaanottopuskurissa (tavua)

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

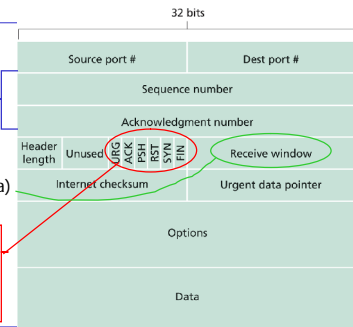


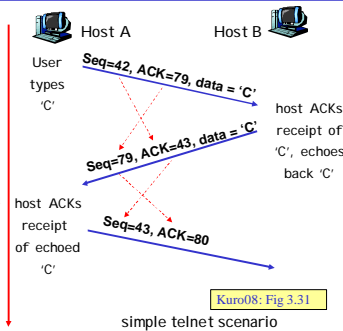
Figure 3.29 TCP segment structure

Tavunumerointi

Tavuvirtaa ...
Segmentit voivat olla erikokoisia (= <= MSS)

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

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



Kuro08: Fig 3.31

simple telnet scenario

TCP: lähetyksen (simplified)

```

NextSeqNum = InitialSeqNum; SendBase = InitialSeqNum
loop (forever) {
  switch(event) {
    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)
    event: timer timeout
    retransmit not-yet-acknowledged segment with smallest sequence number
    start timer
    event: ACK received, with ACK field value of y
    if (there are currently not-yet-acknowledged segments)
      start timer
  } /* end of loop forever */

```

Vuonvalvonta ja/tai ruuhkanhallinta voi estää lähettämisen!

Riittääkö 1 segmentti?

Ajastimen arvo?

Yksi vai monta?

Toistokuittaukset?

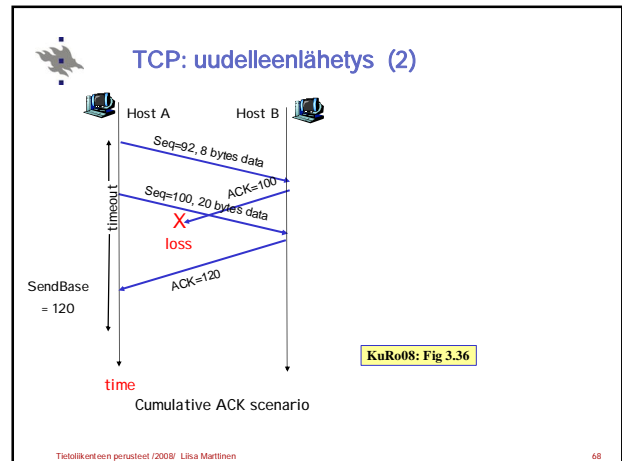
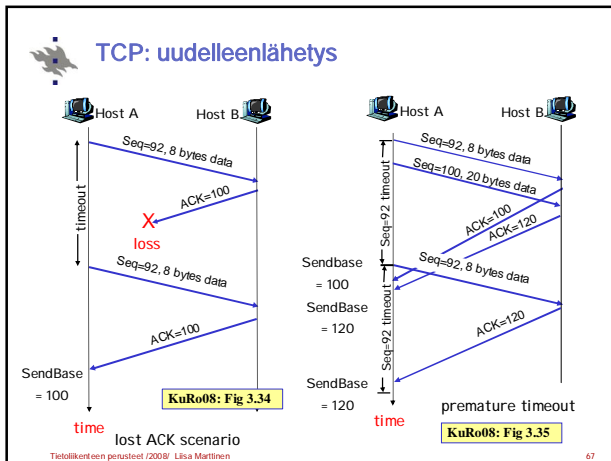
Kuro08: Fig 3.33

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



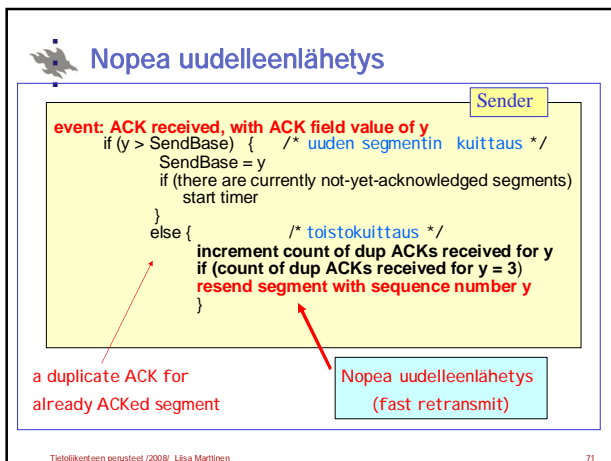
TCP ACK generation [RFC 1122, RFC 2581]

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 Joka toinen kuitattava!
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

TCP:n kuitattava vähintään joka toinen segmentti ja kuittausta saa viivyttää korkeintaan 500 ms.

Tietoliikenteen perusteet /2008/ Liisa Marttinen 69

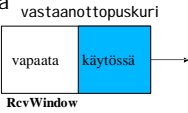
- ### Nopea uudelleenlähetyt (fast retransmit)
- Timeout suhteellisen pitkä
 - n => aika iso viive ennen uudelleenlähetytstä
 - Vastaanottaja ilmoittaa puuttuvasta segmentistä toistokuitauksilla (duplicate ACK)
 - n Liukuhinnoituksen vuoksi useita segmenttejä voi olla kuittaamatta
 - n Jos välistä puuttuu segmentti, seurauksena on useita ACK-kuitauksia
 - Jos lähettäjä saa 3 samaa segmenttiä kuittaavaa toistokuittausta, se olettaa, että seuraava segmentti on kadonnut (siis kaikkiaan 4 samaa)
 - n Ja lähettää puuttuvan segmentin heti
 - Nopea uudelleenlähetyt = lähetyt uudelleen jo ennen kuin ajastin laukeaa eli kolmen tuplakuitauksen jälkeen.
- Tietoliikenteen perusteet /2008/ Liisa Marttinen 70



- ### Paluu n:ään vai valikoiva toisto?
- Kumpaa TCP käyttää?
 - n Liukuvan ikkunan protokolla
 - n Hybridi, tavallaan 'best-of-both'
 - Go-Back-N-tyyppinen
 - n Virheellisiä tai väärässä järjestyksessä tulleita ei hyväksytä, mutta ne yleensä talletetaan puskurin
 - Kumulatiivinen ACK
 - Kaikkia virheellisestä lähtien ei tarvitse lähettää uudestaan
 - Valikoiva toisto
 - n Kumulatiivinen ACK ei kelpaa
 - n SACK-kuittaus (selective ACK), joka kertoo, mitkä segmentit vastaanotettu (RFC 2018), ei yleisesti käytössä
- Tietoliikenteen perusteet /2008/ Liisa Marttinen 72

Vuonvalvonta

- Jotta lähettäjä ei tukahduta vastaanottoa
 - Siirtonopeus sovitettava vastaanottavan sovelluksen mukaan
 - Kuittaus on irroitettu vuonvalvonnasta
 - Lukuva Ikkuna, koko vaihtelee**
 - Kuittaus siirtää ikkunaa
 - Vuonvalvonta määrää ikkunankoon**
 - Kun ikkunankoko = 0, ei saa lähettää!
- Vastaanottaja kertoo, montako tavua puskureihin vielä mahtuu
 - TCP-segmentin otsakkeen kenttä **Receive window**
 - Sovellus lukee tavut silloin kun haluaa**
 - Koko on mukana jokaisessa TCP-segmentissä (molempiin suuntiin)**
 - Myös ruuhkanhallinta rajoittaa lähettämistä**

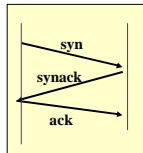


Vuonvalvonta

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

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 RST-segmentti eli yhteyttä ei voida muodostaa
- Varaa puskuritilaa**
 - Lähettäjä puskuroid uudelleenlähetystä varten
 - Vastaanottaja saatujen pakettien järjestämistä varten
- Sovitaan tavunumeroinnin alkuarvoista**
 - Kuittauksia varten
- Ilmoita oma vastaanottoikkunan koko**
 - Vuonvalvontaa varten



Yhteyden muodostus

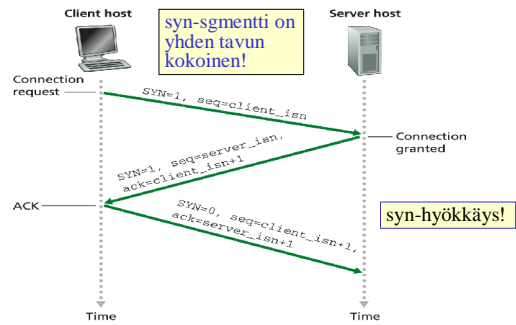
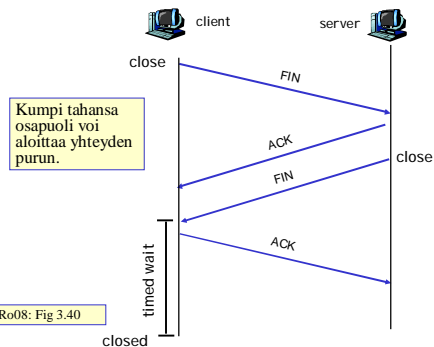


Figure 3.39 TCP three-way handshake: segment exchange

Yhteyden purku

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

Yhteyden purku



KuRo08: Fig 3.40

Kuljetuskerros

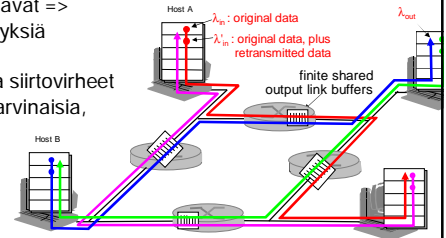
Ruuhkanhallinta TCP:ssä

Tietoliikenteen perusteet / 2008/ Liisa Marttinen

79

Ruuhka

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



Tietoliikenteen perusteet / 2008/ Liisa Marttinen

80

Miten ratkaista ongelma?

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

Tietoliikenteen perusteet / 2008/ Liisa Marttinen

81

Ruuhkaikkuna (congestion window)

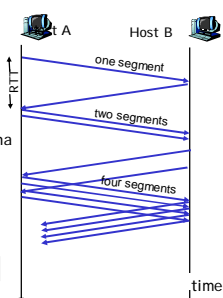
- Internetin hetkellinen kuormitus vaihtelee
- Ruuhkaikkuna
 - Paljonko lähettäjä saa tietyllä hetkellä kuormittaa verkkoa
 - Paljonko lähettäjällä saa olla kuitaamattomia segmenttejä
- Lähettäjän päätettävä itse sopiva ikkunankoko
 - Jos uudelleenlähetysajastin laukeaa, on ruuhkaa
 - Jos kuitaukset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

Tietoliikenteen perusteet / 2008/ Liisa Marttinen

82

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

- Aluksi ruuhkaikkuna = yksi segmentti
 - Alussa hidas siirtonopeus = MSS/RTT
- Kukin kuitaus kasvattaa yhdellä ruuhkaikkunan kokoa **hidas aloitus**
 - Eksponeentiaalinen kasvu
 - Ikkuna kaksinkertaistuu yhden RTT:n aikana
- Jos uudelleenlähetys, ruuhkaikkunan kooksi 1 segmentti
 - Multiplicative decrease
- Sen jälkeen kasvata ikkunaa yksi segmentti/RTT **ruuhkanvälttely**
 - Lineaarinen kasvu (Additive increase)
 - Ruuhkan välttely (congestion avoidance)
- Siirtonopeus = $\text{CognWin} / \text{RTT}$ tavua/sek



Tietoliikenteen perusteet / 2008/ Liisa Marttinen

83

Kynnysarvo (threshold)

- $n \approx$ varoitus: tästä lähtien syytä varoa ruuhkaa
 - Aluksi threshold = 64 KB
- Ajastimen laukeamisen (timeout) ja jälkeen
 - Threshold = $\text{CongWin} / 2$
- Kynnysarvoon saakka ikkunan kasvatus eksponentiaalista
 - Yhtä kuitattua segmenttiä kohden saa lähettää kaksi uutta
 - Eli kaksinkertaistuu yhden RTT:n aikana
- Sen jälkeen ikkuna kasvaa lineaarisesti **ruuhkanvälttely**
 - Kasvaa yhdellä yhden RTT:n aikana
- Miksi näin?

Tietoliikenteen perusteet / 2008/ Liisa Marttinen

84

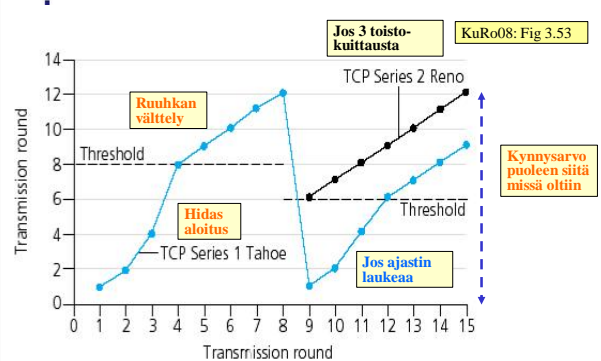
TCP Reno: Tarkennus

- Saatu 3 ACK-kaksokuitausta (double ACK) (4 samaa kuitausta!)**
 - 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 koko ja kasvata sitten lineaarisesti
 - Timeout (= uusi hidas aloitus)**
 - Verkko pahasti ruuhkautunut! **Hidas aloitus**
 - Pudota ikkunankoko arvoon 1
 - Kasvata eksponentiaalisesti kynnyсарvoon asti
 - Kasvata sitten lineaarisesti **ruuhkanvälttely**
- Vanha TCP Tahoe pudotti aina kokoon 1.**

Tietoliikenteen perusteet /2008/ Liisa Marttinen

85

TCP Tahoe vs. TCP Reno



Tietoliikenteen perusteet /2008/ Liisa Marttinen

86

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

KuRo08: Table 3.3

Tietoliikenteen perusteet /2008/ Liisa Marttinen

87

Ajastimen arvo?

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

Tietoliikenteen perusteet /2008/ Liisa Marttinen

88

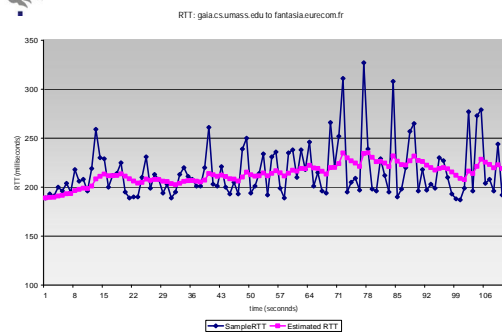
Ajastimen arvo?

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

Tietoliikenteen perusteet /2008/ Liisa Marttinen

89

Esimerkki



Kuro05:Fig.3.32 RTT samples and RTT estimates

Tietoliikenteen perusteet /2008/ Liisa Marttinen

90



Ajastinajan kaksinkertaistaminen

- n Aina kun saadaan lähetettäväksi sovellukselta dataa tai saadaan kuittaus jo lähetettyyn segmenttiin, otetaan käyttöön tuorein estimoitu ajastimen arvo.
- n **Ajastimen laukeaminen =>**
 - n Kun segmentti lähetetään uudelleen, kaksinkertaistetaan ajastimen arvo.
 - n Jos sama segmentti joudutaan lähettämään useaan kertaan uudestaan, niin ajastimen arvo aina kaksinkertaistetaan.

Esimerkki: Lähetetään segmentti 100 ja ajastimen arvo 2.5 sekuntia.

Ajastin laukeaa ja lähetetään segmentti 100 uudestaan ja nyt ajastimen arvo on 5 sekuntia.

Kun kuittaus ei tule 5 sekunnin sisällä, niin ajastin taas laukeaa ja segmentti 100 lähetetään vielä kerran. Nyt ajastimen arvoksi asetetaan 10 sekuntia.

Tähän saadaan kuittaus ajoissa ja siirrytään lähettämään segmenttiä 1100. Ajastimen arvoksi asetetaan tuorein estimoitu arvo 3.2 sekuntia.

Tietoliikenteen perusteet /2008/ Liisa Marttinen

91



Onko TCP reilu? Kohdellaanko TCP-yhteyksiä reilusti?

- 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 /2008/ 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. 293

Tietoliikenteen perusteet /2008/ Liisa Marttinen

93