

3. Kuljetuskerros

3.1. Kuljetuspalvelu

■ 'End-to-end'

- prosessilta prosessille looginen yhteys
 - portti
- verkkokerros koneelta koneelle
 - IP-osoite

■ peittää verkkokerroksen puutteet

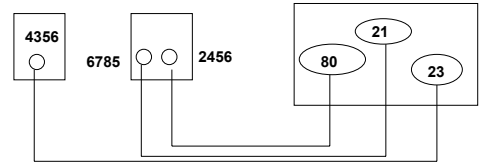
- jos verkkopalvelu ei ole riittävän hyvä, sitä voidaan parantaa kuljetuskerroksella
 - kuljetuskerros huomaa verkkokerroksen kadottamat paketit ja pyytää niiden uudelleenlähetystä

1/31/2003

1

Asiakkaalle kuljetuskerros usein automaattisesti antaa käyttöön jonkin vapaan porttinumeron yhteyden ajaksi

Palvelimilla kiinteät numerot yhteydenottoa varten



Kolme yhteyttä: 4356 <=> 23, 6785 <=> 21, 2456 <=> 80

1/31/2003

4

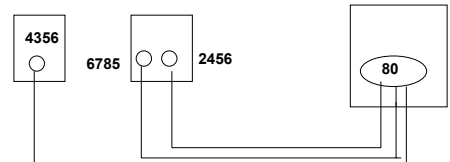
Sovelluksien datavirtojen erottaminen

- IP-osoite
 - osoittaa koneen yksikäsitteisesti
- Sovellusprosessi tunnistetaan porttinumerosta (16 bittiä =>0-65535)
 - jokaisessa lähetetyssä segmentissä on
 - lähettäjän porttinumero
 - vastaanottajan porttinumero
- Yleisillä palvelimilla omat varatut porttinumerot (0-1023)
 - SMTP 25, HTTP 80, jne

1/31/2003

2

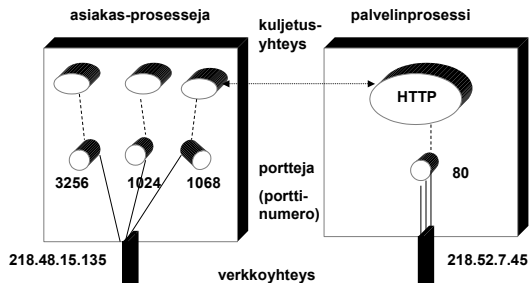
Tarvitaan sekä lähteen että kohteen porttinumerot



Kolme yhteyttä: 4356 <=> 80, 6785 <=> 80, 2456 <=> 80

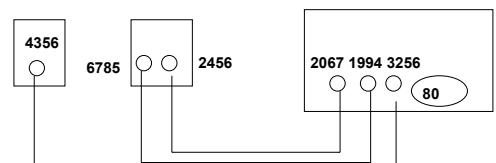
1/31/2003

5



Kuljetusyhteys on looginen prosessilta prosessille yhteys (end-to-end)

Palvelimissa yhteyksille uudet porttinumerot, jotta portti 80 voi ottaa vastaan uusia yhteyspyyntöjä

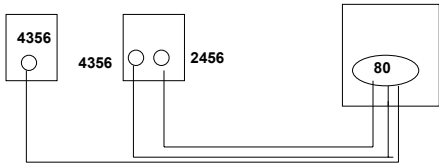


Kolme yhteyttä: 4356 <=> 3256, 6785 <=> 1994, 2456 <=> 2067

1/31/2003

6

Eri koneissa voidaan ottaa sama numero!



Kolme yhteyttä: 4356 <=> 80, 4356 <=> 80, 2456 <=> 80!

Kuljetusyhteydellä käytetään apuna myös IP-osoitetta:

=> koneilla eri IP-osoitteet, joten yhteydet pystytään erottamaan

Internetin kuljetuskerros

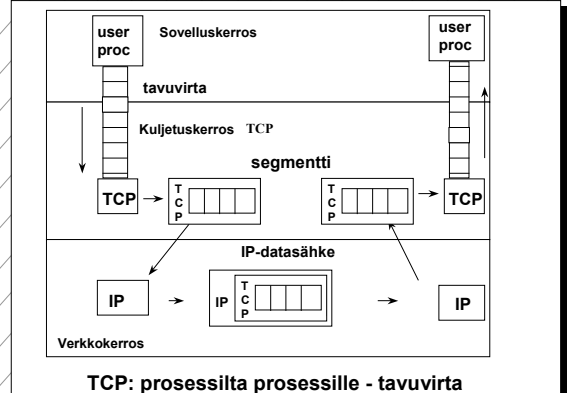
- **UDP (User Datagram Protocol)**
 - yhteydetön, epäluotettava palvelu
- **TCP (Transmission Control Protocol)**
 - yhteydellinen, luotettava palvelu
 - virhevalvonta
 - havaitsee ja korjaa siirrossa syntyneet virheet
 - vuonvalvonta
 - ei ylikuormita vastaanottajaa
 - ruuhkanvalvonta
 - huolehtii ettei verkko pääse ruuhkautumaan

Sovelluksen vaatimuksia kuljetuspalvelulle:

- Virheetön, luotettava
- järjestyksen säilyttävä
- kaksoiskappaleet karsiva
- mielivaltaisen pitkiä sanomia salliva
- vuonvalvonnan mahdollistava

Verkkokerros kuitenkin voi

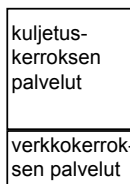
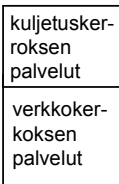
- kadottaa sanomia
- toimittaa sanomat epäjärjestyksessä
- viivyttaa sanomia satunnaisen pitkän ajan
- luovuttaa useita kopioita samasta sanomasta
- rajoittaa sanomien kokoa



TCP: prosessilta prosessille - tavuvirta

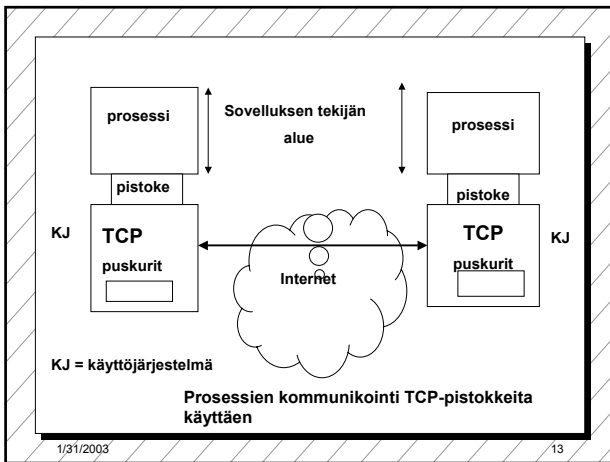
kuljetuspalvelut parantavat verkkopalveluja

Sovelluksen näkemä palvelun laatu (Quality of Service, QoS)



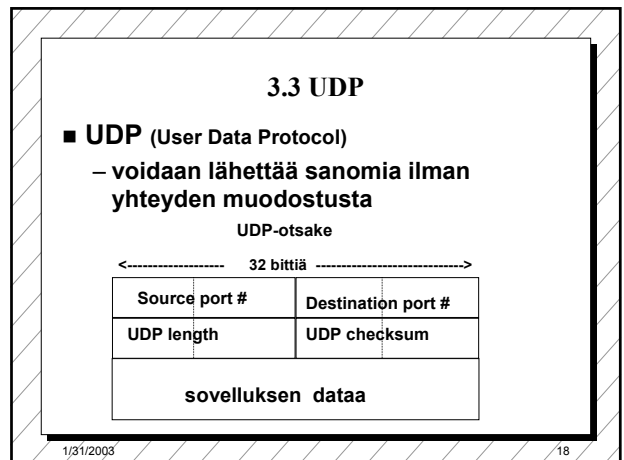
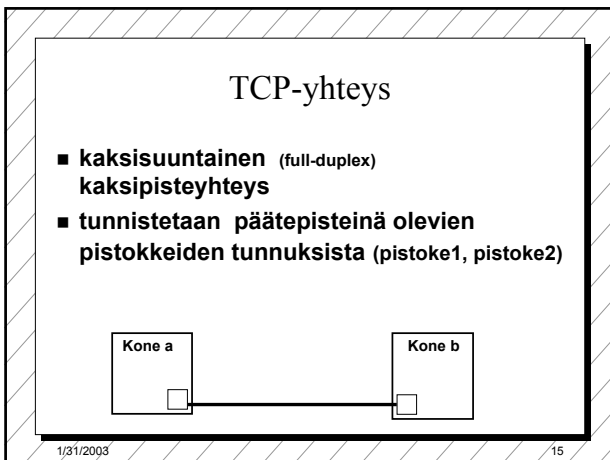
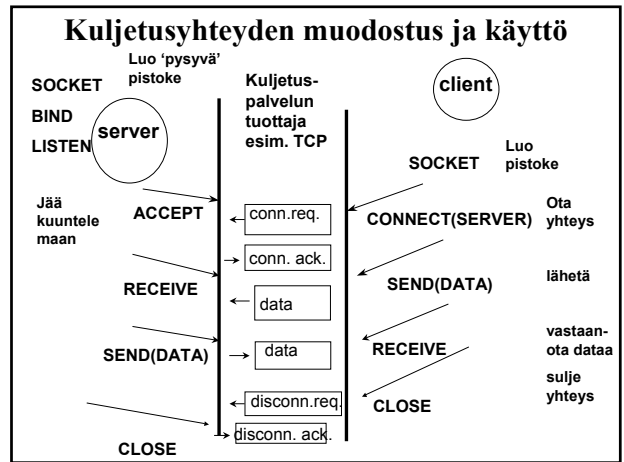
Pistokerajapinta (Socket interface)

- **Verkkopalvelun ja sitä käyttävän sovelluksen rajapinta**
 - yleensä käyttöjärjestelmän tarjoama palvelu
 - pistokerajapinta alunperin Berkeley Unixin mukana, nyt lähes kaikissa käyttöjärjestelmissä
 - miten verkkoprotokollan tarjoamiin palveluihin päästään käsiksi sovelluksesta



- ## TCP:n pistokeprimitiivit
- **SOCKET** luo uusi yhteydenpäätepistepistoke
 - **BIND** anna pistokkeelle osoite
 - **LISTEN** halukas vastaanottamaan yhteyksiä
 - **ACCEPT** jää odottamaan yhteyksiryöksiä
 - **CONNECT** yritä muodostaa yhteys
 - **SEND** lähetä dataa yhteyttä pitkin
 - **RECEIVE** vastaanota dataa yhteydeltä
 - **CLOSE** pura yhteys (symmetrinen)
- 1/31/2003 16

- **pistoke (socket)**
 - TCP-yhteyden päätepiste sovellukselle
 - lähettäjällä ja vastaanottajalla oma pistoke
 - pistokenumero 48 bittiä
 - koneen 32 bitin IP-osoite
 - 16 bitin porttinumero
- 1/31/2003 14



UDP-tarkistussumma

- Virheen havaitsemista varten otsakkeeseen liitetään tarkistussumma
 - kaikki segmentin 16 bitin sanat lasketaan yhteen ja summasta otetaan yhden komplementti
 - = muutetaan ykköset nolliksi ja nollat ykkösiksi
 - vastaanottaja laskee taas kaikkien segmentin sanojen (mukana myös tarkistussumma) summan jos tulokseksi saadaan 16 ykköstä, niin ok!

1/31/2003

19

UDP:n etuja:

- Yhteydetön
 - aikaa ei kulu yhteyden muodostamiseen ja purkamiseen
 - ei tarvita resursseja yhteyden tilatietojen ylläpitoon
- Otsake (= 8 tavua) pieni => pieni yleisrasite => lisää tehokkuutta
- Ruuhkanvalvonta ei säännöstele liikennettä

1/31/2003

22

Esimerkki

- Lasketaan tarkistussumma kolmen tavun mittaiselle sanomalle:

Lähettäjä	vastaanottaja	
1011 0100	1011 0100	1011 0100
0111 0101	0111 0101	1111 0101
1000 1101	1000 1101	1000 1101
=====	0100 1001	0100 1001
1011 0110	=====	=====
↓	1111 1111	0111 1111
0100 1001	OK!	VIRHE!

Yhden komplementti

1/31/2003

20

Tehtäviä:

- Lähetetään 10 tavun viesti UDP:llä.
 - Miten kauan kestää lähettäminen, jos lähetyksenopeus on 56 kbps?
 - $10 \text{ tavua} + 8 \text{ tavua} = 18 * 8 \text{ b} = 144 \text{ bittiä}$
 - $144 \text{ b} / 56 \text{ 000 b/s} = 2.57 \text{ ms}$
 - Miten suuri on etenemisviive, 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)?
 - $8/18 = 0.44$ eli 44 %

1/31/2003

23

- Miksi tarvitaan tarkistussumma?
 - Kaikki siirtoyhteyserrokset eivät suorita tarkistuksia
- UDP-tarkistussumma ei ole kovin tehokas havaitsemaan virheitä!
- Se ei myöskään yritä toipua virheistä!
 - Jotkut toteutukset voivat tuhota virheellisen segmentin
 - jotkut antavat se sovellukselle varoituksen kera

1/31/2003

21

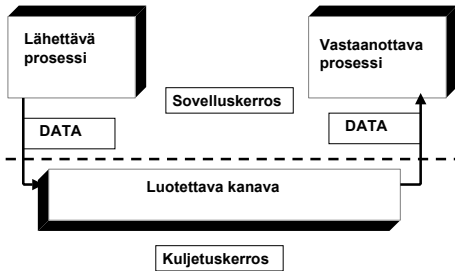
UDP:n käyttö

- Vaikka UDP on epäluotettava, se sopii monien sovellusten tarpeisiin:
 - Remote file server (NFS)
 - multimedia
 - Internet-puhelin
 - verkon hallinta (SNMP)
 - reititys (RIP)
 - nimipalvelu (DNS)
- Miksi nämä sovellukset suosivat UDP:tä?

1/31/2003

24

3.4 Luotettava tiedonsiirto



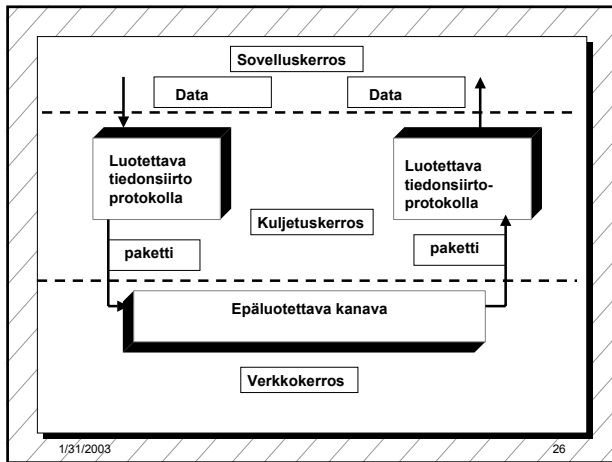
1/31/2003

25

Kohinainen kanava



- sanoma vääristyy => virhetarkistus
- sanoma katoaa => numerointi, ajastin ja uudelleenlähetyt
 - duplikaattien havaitseminen
- sanoma viivästyy => rajallinen elinaika
- sanomien järjestys muuttuu => järjestäminen



1/31/2003

26

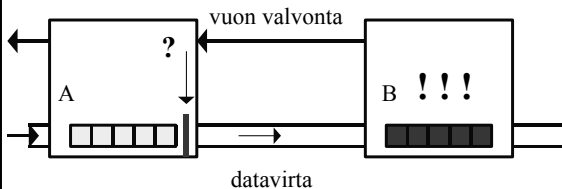
Yksinkertainen Stop and wait -protokolla

- Oletus
 - virheetön siirto => ei huolta virheistä, mutta vuonvalvontaa tarvitaan
- lähettäjä
 - lähettää sanoman
 - odottaa lupaa lähettää seuraava sanoma
- vastaanottaja
 - käsittelee sanoman
 - lähettää tiedon (=antaa luvan) lähettäjälle

1/31/2003

29

Vuon valvonta



- X-ON / X-OFF : GO! | STOP!

Entä jos virheitä?

- Sanomissa virheitä tai sanomat voivat puuttua kokonaan
- Myös kuittaukset voivat kadota
- Tarvitaan
 - virheen havaitseminen ja korjaaminen
 - tarkistussumma
 - kuittaus
 - uudelleenlähetyt
 - sanomien numerointi
 - uudelleenlähetytsajastin

1/31/2003

30

Monimutkaisempi "stop and wait"-protokolla

■ ajastin lähettäjälle

- jos kuittausta ei kuulu, sanoma lähetetään automaattisesti uudelleen
- kuittaus: ACK = 'ok, lähetä seuraava'
- uudelleenlähetys synnyttää kaksoiskappaleita!

■ Sanomanumerointi

- jotta vastaanottaja tunnistaa kaksoiskappaleet
- Miten paljon numeroita tarvitaan?
 - » Numero vie tilaa sanomassa!

Liukuvan ikkunan protokolla

(Sliding Window)

■ Lähetyssikkuna

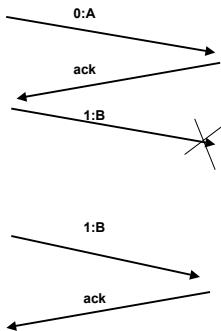
- ikkunan koko
 - montako sanomaa saa korkeintaan olla kuittaamatta
 - järkevä koko riippuu yhteyden tyypistä ja vastaanottajan kapasiteeteista
 - kiinteä koko /vaihteleva koko
- sisältö = mitkä sanomat saa lähettää
 - sanomalla järjestysnumero
 - rajallinen, N bittinä => 2^{*N} arvoa
 - numerot käytettävä järjestyksessä

1/31/2003

34

Lähettäjä:

vastaanottaja:



■ Lähettäjä joutuu odottamaan vasta, kun kaikki ikkunan sanomat on lähetetty

- eli numerot käytetty

■ Kun kuittaus saapuu => ikkuna liikuu

- seuraavat numerot tulevat luvallisiksi

■ eli

- lähettäjä: tietyllä hetkellä sallittujen numeroiden joukko = lähettäjän ikkuna
 - mitkä sanomat saa lähettää "otukäteen" odottamatta kuittausta

1/31/2003

35

Stop and wait -protokollan suorituskyky

■ Esim. satelliittiyhteydellä

- 50 kbps, kiertoviive ~520 ms, sanoma 1000 bittinä
- kanavan käyttöaste < 4%

■ => lähetetään useita sanomia ja sitten vasta odotetaan kuittauksia

- ideaali: lähetykset liukuhihnalla (pipeline)
 - lähetykset ja kuittaukset liimittävät
 - ei mitään odottelua
 - lähetyskanava koko ajan käytössä
- suorituskyky kasvaa

■ Vastaanottajan ikkuna

- kullakin hetkellä sallittujen numeroiden joukko
 - mitä sanomia suostuu vastaanottamaan

- kuittaus muuttaa myös vastaanottajan ikkunan

■ ikkuna pysäyttää sanomien lähetyksen

- seuraava sanomanumero ei ole lähetyssikkunassa

■ ikkuna estää sanoman vastaanoton

- saadun sanoman numero ei ole vastaanottoikkunassa

Kun ikkunan koko on 1

- Aina vain yksi sanoma kuittaamattomana
 - => One Bit Sliding Window -protokolla
 - ~ stop and wait -protokolla
- sanomanumerot 0 ja 1 riittävät
- ACK-sanoma identifioi viimeksi vastaanotetun virheettömän sanoman
 - jotta kuittausduplikaatti ei voi kuitata väärää sanomaa
 - ACK ilmoittaa joko
 - » seuraavaksi odotetun sanoman numeron
 - » viimeksi vastaanotetun sanoman numeron

1/31/2003

37

Valikoiva toisto

- vastaanottaja hyväksyy kaikki kelvolliset sanomat
 - se kuittaa sanomat
 - puskuroi ne ja toimittaa eteenpäin oikeassa järjestyksessä
 - » tarvitaan puskuritilaa
- lähettäjä ei saa kuittausta virheellisestä sanomasta
 - ajastin laukeaa ja sanoma lähetetään uudelleen
 - lähettää uudelleen vain virheellisen sanoman
 - ikkuna liukuu nytkin tasaisesti
 - » yksi puuttuva kuittaus voi pysäyttää lähetyksen

- Entä kun tapahtuu virhe?
 - kaksi eri tapaa hoitaa
 - 1. toisto virheestä lähtien (go back n) (tai paluu n:ään)
 - 2. valikoiva toisto (selective repeat)

1/31/2003

38

Kuittaukset

- ACK
 - kumulatiivinen ACK
 - tähän saakka kaikki ok!
 - Go-Back N
 - yksittäinen ACK
 - vain tämä ok!
 - Valikoiva toisto
- NAK-kuittaus
 - sanoma virheellinen tai puuttuu

1/31/2003

41

Toisto virheestä eli Paluu n:ään ('Go back n')

- virheellisen sanoman havaittuaan
 - vastaanottaja hylkää kaikkia sen jälkeiset sanomat eikä lähetä niistä kuittauksia
 - => sanomat hyväksytään vain oikeassa järjestyksessä
- kun lähettäjä ei saa kuittauksia,
 - sen lähetyksikkuna 'täytyy'
 - eikä se voi enää lähettää
- lähettäjän ajastimet laukeavat aikanaan ja
 - virheellinen sanoma
 - sekä kaikki sen jälkeen lähetetyt sanomat lähetetään uudelleen
- tehoton, jos paljon virheitä ja iso ikkuna

Negatiiviset kuittaukset

- NAK-kuittauksilla voidaan nopeuttaa uudelleenlähettämistä
 - vastaanottaja ilmoittaa heti virheellisestä tai puuttuvasta kehuksesta
 - ei ole tarpeen odottaa ajastimen laukeamista
- hyödyllinen, jos kuittausten saapumisaika vaihtelee paljon
 - ajastinta vaikea asettaa oikein

1/31/2003

42

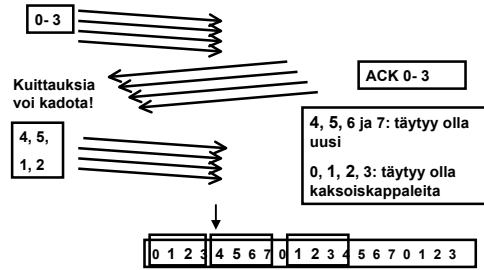
- **NAK-kuittaukset voivat aiheuttaa turhia uudelleenlähetystyksiä**
 - lähetys ja kuittaus menevät ristiin
- **NAK-kuittauksen katoaminen ei haittaa**
- **implisiittinen uudelleenlähetys**
 - ei NAK-kuittauksia
- **explisiittinen uudelleenlähetys**
 - käytetään NAK-kuittauksia

1/31/2003

43

Miksi?

Valikoiva toisto: ikkuna 4, numeroavaruus 8



1/31/2003

46

Ikkunankoko

- **Kun käytetty numeroavaruus on 0, 1, .. n ja eri numeroita siis käytettävissä n+1**
 - yleensä jokin kakkosen potenssi
 - » koska numerokentän koko k bittiä => käytössä 2^{k+1} numeroa
- **ikkunan koko 'go back n':ssä voi olla korkeintaan n**
 - eli oltava ainakin yhtä pienempi kuin numeroavaruus
- **ikkunan koko valikoivassa toistossa voi olla korkeintaan $(n+1)/2$**
 - saa olla korkeintaan puolet numeroavaruudesta

1/31/2003

44

Kaksisuuntainen liikenne

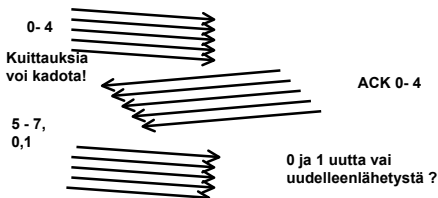
- **datakehys ja kuittauskehys**
- **kehyksessä sekä data että kuittaus**
 - 'piggybacking'
 - tehostaa lähetystä
- **ongelma: kauanko kuittaja odottaa dataa ennen pelkän kuittauksen lähettämistä?**

1/31/2003

47

Miksi?

Valikoiva toisto: ikkuna 5, numeroavaruus 8



1/31/2003

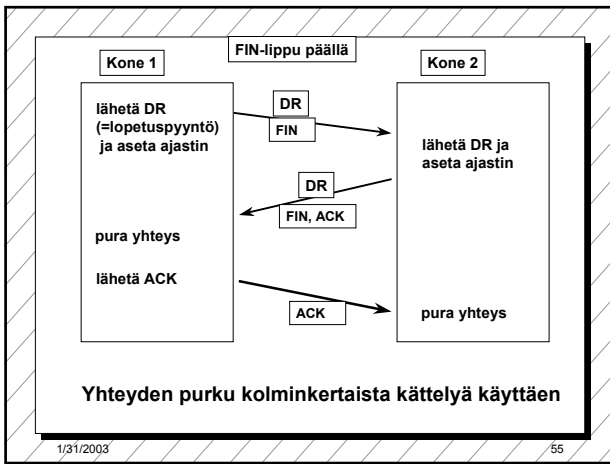
45

3.5. TCP-protokolla

- **yhteyden muodostus ja purku**
- **luotettavan tavuvirran toteuttaminen**
- **vuonvalvonta**
- **siirron optimointi**
- **TCP-segmentti**
- **ruuhkan valvonta**
- **TCP-palvelun käyttö**

1/31/2003

48

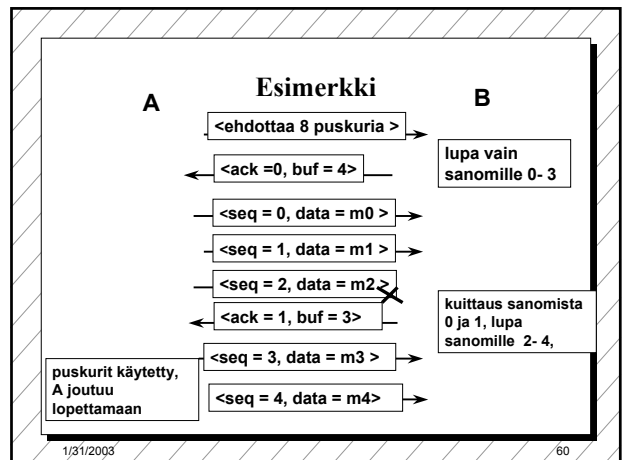


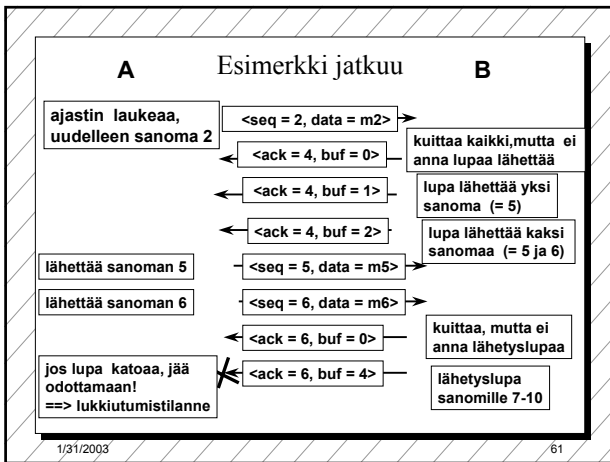
- ## Toistokuittaukset
- **Ensikuittaus**
 - ensimmäinen vastaanotettu sanoman kuittaus
 - ACK(i): sanomaan i saakka kaikki OK!
 - **toistokuittaus (duplicate ACK)**
 - väärässä järjestyksessä saatu segmentti tai virheellinen segmentti => toistetaan uudestaan jo annettu kuittaus
 - NAK-kuittauksen korvike
 - 3 toistokuittaus => segmentti kadonnut tai virheellinen
- 1/31/2003 58

- ## TCP: Virheettömyys ja järjestys
- **Järjestysnumerot**
 - tavuvirta => tavunumerointi
 - segmentin 1. tavun järjestysnumero
 - yhteyden alussa satunnaiset numerot
 - **kuittaukset**
 - kumulatiivinen ACK, ei NAK-kuittaus
 - kuittauksessa seuraavaksi odotettava tavu
 - kuitataan 'tiheästi'
 - vähintään joka toinen
- 1/31/2003 56

- ## TCP:n vuonvalvonta
- **'joustava' liukuva ikkuna** (sliding window) ("credit-vuonvalvonta")
 - **vastaanottaja kertoo, kuinka paljon suostuu vastaanottamaan**
 - => kuittaus irroitettu vuonvalvonnasta
 - puhtaassa liukuvassa ikkunassa kuittaus siirtää ikkunaa
 - AdvertisedWindow-kenttä
 - paljonko saa lähettää = paljonko vastaanottajan puskureihin mahtuu
 - **myös ruuhkan valvonta rajoittaa lähettämistä**
- 1/31/2003 59

- **Go Back N -tyyppinen**
 - virheellisiä tai väärässä järjestyksessä tulleita ei hyväksytä
 - ne voidaan myös tallettaa
 - mutta ei välttämättä lähetä kaikkia virheellisestä lähtien uudestaan
 - **Myös ehdotettu valikoivan toiston tyyppistä kuittamista**
 - SACK-kuittaus, joka kertoo, mitkä segmentit on vastaanotettu ok
- 1/31/2003 57





- Optimointi on usein tarpeen:
- Interaktiivinen editori => merkki lähetetään heti
 - 21 tavun TCP-segmentti => 41 tavun IP-paketti
 - joka kuitataan 40 tavun IP-paketilla
 - ilmoitus uudesta ikkunan koosta 40 tavun IP-paketilla
 - kuitetaan merkki vielä 41 tavun IP-paketilla
 - yhden merkin käsittely =>
 - 162 tavun siirtäminen
 - ja neljän segmentin lähettäminen
- 1/31/2003 64

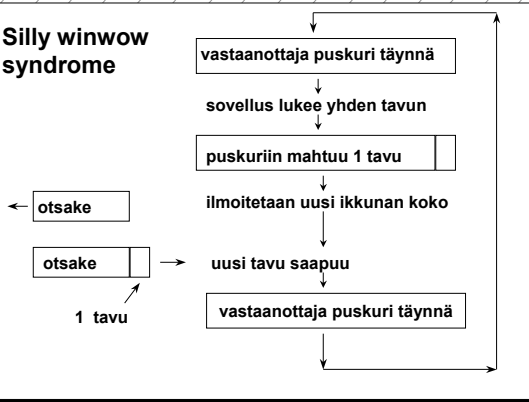
- jos ilmoitus lisäpuskureista katoaa, lähettäjä lukkiutuu odotustilaan
 - vastaanottaja voi luulla, ettei ole lähetettävää
 - lukkiutumisen estämiseksi
 - kun ikkunankoko = 0 lähettäjä ei saa lähettää, paitsi
 - erityistä pikadataa (URG)
 - yhden tavun 'kyselyn', jonka vastaanottaja kuittaa ja samalla ilmoittaa ikkunan koon => estää turhat lukkiutumiset
- 1/31/2003 62

- Ratkaisu: Naglen algoritmi
 - jos data tulee tavuttain
 - lähetä 1. tavu
 - kerää sitä seuraavat tavut puskuuriin ja lähetä vasta kun edellinen lähetys on kuitattu
 - paitsi jos lähetettävää on suurimman segmentin verran tai puolet ikkunan koosta
 - hankala, jos hiirtä liikutellaan Internetin kautta!
- 1/31/2003 65

- Siirron optimointi**
- TCP saa optimoida lähettämisiään
 - ei tarvitse lähettää heti kun data on tullut
 - dataa kerätään puskuuriin ja lähetetään sopivassa tilanteessa
 - PUSH-lipun avulla sovellus ilmoittaa, että data on lähetettävä heti
- 1/31/2003 63

- Silly window syndrome**
- Tilanteessa, jossa
 - lähettäjältä dataa TCP:lle suurina lohkoina
 - vastaanottajalle mahtuu vain tavu kerrallaan
 - voi tuhota TCP:n suorituskyvyn
 - koko data lähetetään tavu kerrallaan
 - joka tavun välissä ilmoitus ikkunan koon kasvattamisesta yhdellä
 - Siis: ei ilmoitusta yhdestä tavusta, lähettäjä ei lähetä yhtä tavua
 - koko segmentti
 - puolet puskurin koosta
- 1/31/2003 66

Silly window syndrome



1/31/2003

67

TPC-segmentin otsakekentät

- **Lähde- ja kohdeportit** (Source port, Destination port)
 - yhteyden päätepisteet
 - portti + koneen IP-osoite => 48 bittinen TSAP
- **Järjestysnumero** (Sequence number)
 - tavut numeroidaan => 32 bittiä
 - segmentin ensimmäisen tavun numero
- **Kuittausnumero** (Acknowledgement number)
 - seuraavaksi odotettu tavu
- **TCP-otsakkeen pituus** (TCP header length)
 - mahdollisten optiokenttien takia
- **6 bitin käyttämätön kenttä**

1/31/2003

70

TCP-segmentti

- **segmentti**
 - 20 tavun otsake
 - + optionaalinen osa
 - dataosa
 - voi puuttua
- **segmentin kokoa rajoittaa**
 - MTU (Maximum transfer unit)
 - verkon rajoitus maksimikoolle (muutama tuhat tavua)
 - IP-paketin dataosa korkeintaan 65535 tavua
- **liian isot segmentit paloitellaan**
 - joka palalle IP-otsake => yleisrasite kasvaa

1/31/2003

68

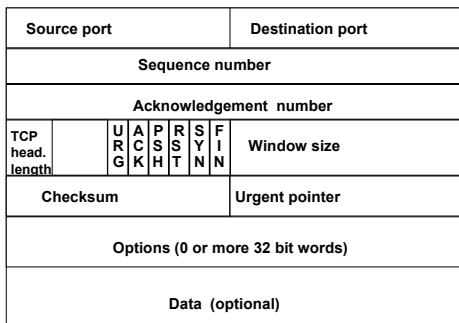
■ 6 lippubittiiä

- **URG** onko pikadataa
pikadatan sijainnin ilmoittaa pikadatakenttä (Urgent pointer)
- **ACK** onko kuittauskenttä käytössä
- **PSH** onko heti lähetettävää (pushed) dataa
- **RST** yhteyden uudelleen alustuspyyntö (reset), yleensä ongelmatilanne
- **SYN** käytetään yhteyttä muodostettaessa
SYN = 1, ACK = 0 connection request
SYN = 1, ACK = 1 connection accepted
- **FIN** käytetään yhteyden purkuun
FIN = 1 ei enää lähetettävää

1/31/2003

71

TCP-otsakkeen kentät



1/31/2003

69

- **Ikkunan koko** (window size)
 - vaihteleva ikkunan koko
 - kuittaus irroitettu lähetysluvasta
- **Tarkistussumma** (Checksum)
 - lasketaan otsakkeelle, datalle ja ns. pseudo-otsakkeelle

1/31/2003

72

pseudo-otsake

Source IP address		
Destination IP address		
00000000	Protocol = 6	TCP/UDP segmentin pituus

Auttaa havaitsemaan väärään osoitteeseen toimitetut paketit.

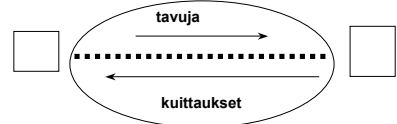
Sisältää IP-otsakkeen tietoja!

1/31/2003

73

■ ruuhkaikkuna

- “paljonko tavuja (segmenttejä) lähettäjällä saa korkeintaan olla verkossa liikkeellä”
 - paljonko lähettäjä saa kuormittaa verkkoa
- kuittaus => ko. tavut jo poistuneet verkosta



1/31/2003

76

■ Optiokenttä (options)

- voidaan lisätä piirteitä, joita ei ole varsinaisessa otsakkeessa
 - suurin hyväksyttävä datakenttä
 - ikkunan koon moninkertaistaminen (window scale)
 - nopeille ja pitkän viipeen linjoille 64 ktavun ikkunan koko on liian pieni
 - valikoivan toiston käyttö ‘go back N’:n tilalla
 - vähentää turhia uudelleenlähetystyksiä

1/31/2003

74

■ Ruuhkaikkunan koko?

- Lähettäjän on itse pääteltävä ja arvioitava sopiva ruuhkaikkunan koko
 - kukaan muu ei sitä kerro!
 - uudelleenlähetyksajastin laukeaa => on ruuhkaa
 - kuittaukset tulevat tasaisesti => ei ole ruuhkaa
- Internet-verkon kuormitus voi vaihdella paljon
- Dynaaminen ruuhkaikkunan koko:
 - ruuhkaikkunaa kasvatetaan, kunnes törmätään ruuhkaan
 - ensin kasvatetaan melko nopeasti, sitten varovaisemmin
 - sen jälkeen ruuhkaikkunaa pienennetään reilusti
 - ja aletaan uudestaan kasvattaa ruuhkaikkunaa

1/31/2003

77

3.6. TCP:n ruuhkan valvonta

- Liikaa kuormitusta => verkko ruuhkautuu => hidastetaan lähettämistä
- Ruuhkan havaitseminen
 - nykyisin siirtovirheet harvinaisia
 - poikkeuksena langattomat verkot
 - => uudelleenlähetystykset johtuvat ruuhkasta
 - uudelleenlähetyksajastimen laukeaminen on merkki ruuhkasta

1/31/2003

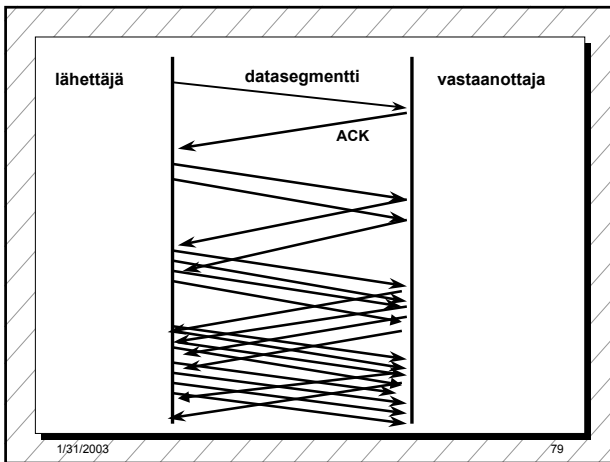
75

Hitaan aloituksen algoritmi (slow start)

- Algoritmi pyrkii löytämään sopivan ikkunan koon yhteyden alussa tai ruuhkatilanteen jälkeen mahdollisimman nopeasti
 - ei ole niin kovin hidas, vaan alussa eksponentiaalinen!
 - alussa ruuhkaikkuna = yksi segmentti
 - kuitattu ruuhkaikkunallinen kasvattaa ruuhkaikkunan kaksinkertaiseksi

1/31/2003

78



Uudelleenlähetysohjauksen hallinta

- uudelleenlähetysohjauksen (retransmission timer)
 - asetetaan aina kun segmentti lähetetään
 - ruuhkaa, jos kuittaus ei saavu ajoissa
- mikä on sopiva ohjauksen aika?
 - kuittaus aika vaihtelee suuresti
 - vaihtelu on myös nopeaa
- dynaaminen arvo
 - saadaan jatkuvien verkon suorituskykymittauksien perusteella

1/31/2003 82

- kynnyksiarvo (threshold)
 - aluksi 64 K
 - 'varoitustarve' = tästä lähtien syytä varoa ruuhkaa
- kynnyksiarvoon saakka voidaan kasvattaa ruuhkaikkunaa eksponentiaalisesti
- kynnyksiarvon saavuttamisen jälkeen kasvatetaan ruuhkaikkunaa vain lineaarisesti
 - = kasvatetaan kuittausten jälkeen vain yhdellä
 - edetään hyvin varovaisesti!

1/31/2003 80

- RTT
 - arvio kiertoviiveelle (round-trip time)
 - mitataan jokaisen lähetetyn segmentin kiertoviive M
 - $RTT = \alpha RTT + (1-\alpha)M$, tyypillisesti $\alpha = 7/8$
- uudelleenlähetysohjauksen arvo βRTT
 - aluksi β oli aina 2
 - parannus: otetaan huomioon myös poikkeama D (deviation) oletetun ja saadun kiertoviiveen välillä $|RTT-M|$
 - $D = \alpha D + (1-\alpha)|RTT-M|$
 - ohjauksen arvo = $RTT + 4 \cdot D$

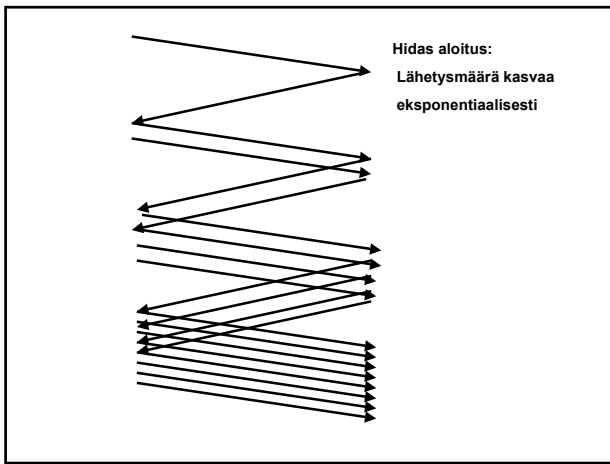
1/31/2003 83

- jos ohjauksen ehtii laua => ruuhkatilanne
 - kynnyksiarvoksi puolet nykyisestä ruuhkaikkunan arvosta
 - hitaalla aloituksella etsitään taas uusi sopiva ruuhkaikkunan arvo
 - ruuhkaikkunan arvoksi 1 segmentti
 - ruuhkaikkunaa kasvatetaan aluksi eksponentiaalisesti eli kaksinkertaistetaan kun ikkunallinen on kuitattu
 - kynnyksiarvon saavuttamisen jälkeen kasvatetaan vain segmentti kerrallaan
 - kunnes taas havaitaan ruuhka ja aloitetaan ruuhkaikkunan uuden arvon etsiminen

1/31/2003 81

- uudelleenlähetysohjauksen vaikutus ohjaukseen
 - kumpaan segmenttiin kuittaus kohdistuu?
- Karnin algoritmi
 - ei oteta huomioon uudelleenlähetysohjauksen kuittausta RTT:n laskemisessa

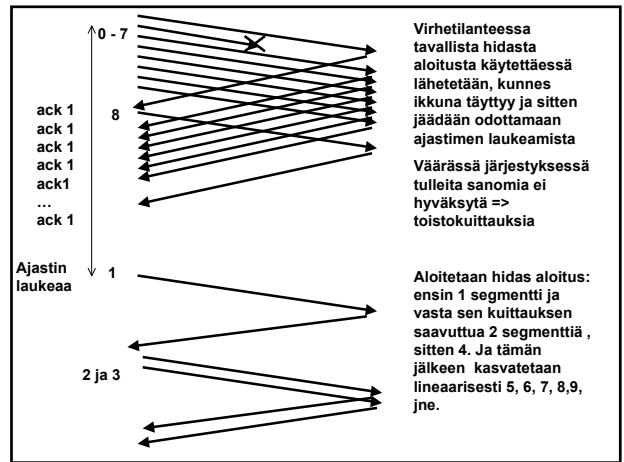
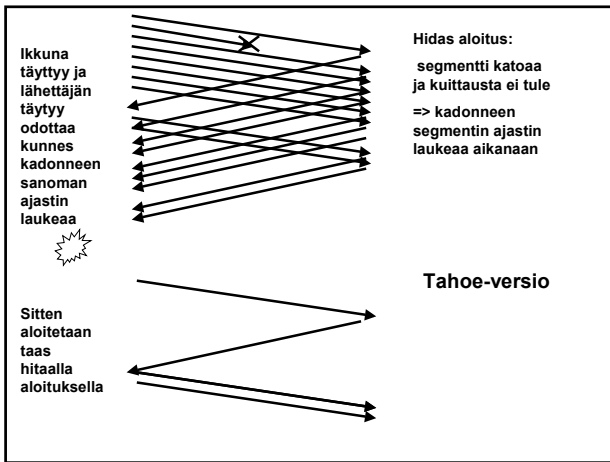
1/31/2003 84



■ **Nopea toipuminen (Fast Recovery)**

- kun kadonnut paketti huomataan nopealla toipumisella, ei aloiteta alusta hitaalla aloituksella
 - vaan pudotetaan ruuhkaikkuna puoleen
 - ja jatketaan normaalilla lineaarisella kasvattamisella
- Mitä hyötyä tästä on?
- Miksi voidaan huoletta tehdä näin?

1/31/2003 88

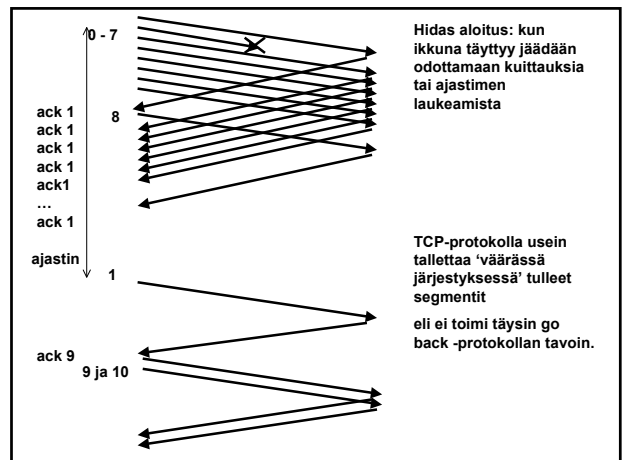


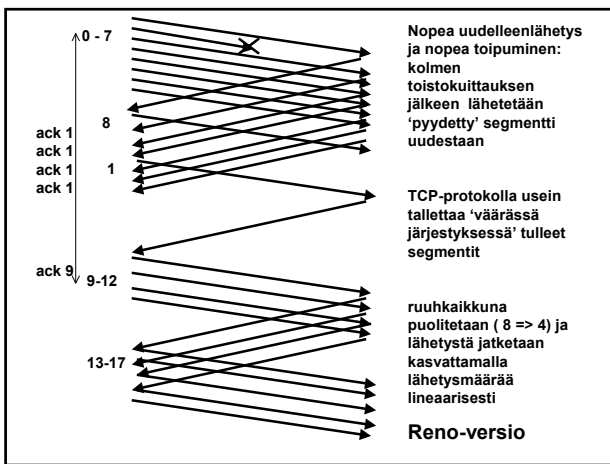
Parannuksia ruuhkanvalvontaan

■ **Nopea uudelleenlähetys (Fast Retransmit)**

- ei odoteta ajastimen laukeamista ennen uudelleenlähetystä
- vastaanottaja kuittaa jokaisen paketin
- kun vastaanottaja huomaa puuttuvan paketin, se lähettää uudelleen edellisen paketin kuittauksen
 - Duplicate ACK (~ NAK)
- kun lähettäjä saa useita (3) peräkkäisiä saman paketin toistokuitauksia => se havaitsee tästä paketin puuttuvan ja lähettää sen heti uudelleen
- => nopeampi uudelleenlähetys

1/31/2003 87





TCP-yhteyden hallinta

- yhteys muodostetaan kolminkertaisella kättelyllä
- passiivinen osapuoli kuuntelee
 - SOCKET
 - BIND
 - LISTEN
 - ACCEPT
- aktiivinen osapuoli aloittaa yhteydenmuodostuksen
 - CONNECT

1/31/2003 94

- hidas aloitus ja ruuhkan valvonta ongelmallisia langattomassa yhteydessä
 - Miksi?
- Lisäparannuksia ruuhkanhallintaan
 - esim. Vegas
 - ruuhkan ennustaminen ennen ajastimen laukeamista
 - ruuhkaikkunaa ei kasvateta aina ruuhkaan asti
 - RED (random early detection)
 - entä UDP?

1/31/2003 92

- **CONNECT-primitiivi**
 - parametreina
 - IP-osoite ja porttinumero
 - suurin hyväksyttävä segmentin koko
 - muuta tietoa, esim. salasana

↓

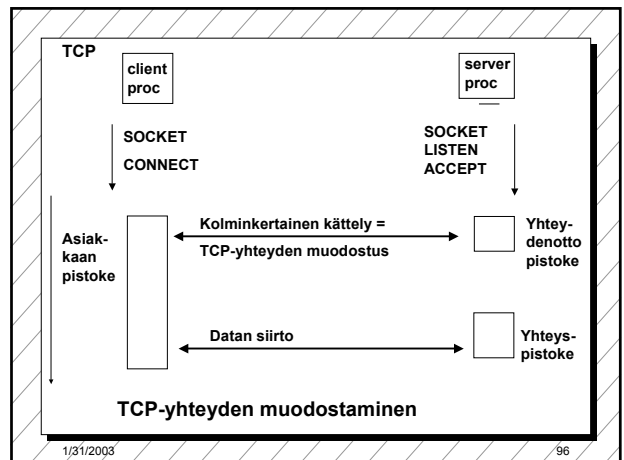
- **TCP-segmentti, jossa SYN-segmentti**
 - SYN = 1
 - ACK = 0

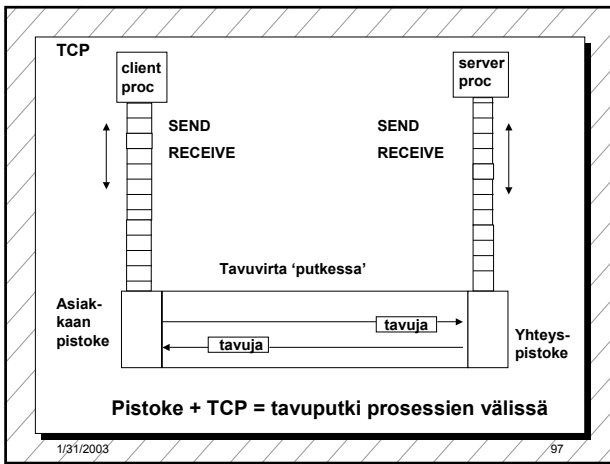
1/31/2003 95

TCP langattomassa verkossa

- monet TCP-toteutukset optimoitu luotettaville langaverkoille => suorituskyky langattomissa verkoissa erittäin huono
 - ruuhkanvalvonta-algoritmi olettaa ajastimen laukeamisen johtuvan ruuhkasta
 - lähettämistä hidastetaan, jotta verkon kuormitus pienenee ja ruuhkaa ei syntyisi
 - langattomat yhteydet ovat epäluotettavia ja paketteja katoaa
 - kadonneet paketit syytä lähettää nopeasti uudelleen
 - lähetystä pitäisi päinvastoin nopeuttaa!

1/31/2003 93





C-rutiineina

```

int socket(int domain, int type, int protocol)
palvelin:
int bind (int socket, struct sockaddr *address,
int addr_len)
int listen(int socket, int backlog)
int accept(int socket, struct sockaddr *address,
int *addr_len)
asiakas:
int connect (int socket, struct sockaddr *address,
int addr_len)

```

1/31/2003 100

- TCP-yhteys on tavuvirtaa, ei sanomavirtaa
 - lähetettäessä neljä 512 tavun pätkää vastaanottaja saa joko
 - neljä 512 tavun pätkää
 - kaksi 1024 tavun pätkää
 - yhden 2048 tavun pätkän
- Segmentit lähetetään neljänä eri IP-pakettina Ne luovutetaan vastaanottajalle yhdellä READ-kutsulla
-
- neljä 512 tavun segmenttiä yksi 2048 tavun data
- 1/31/2003 98

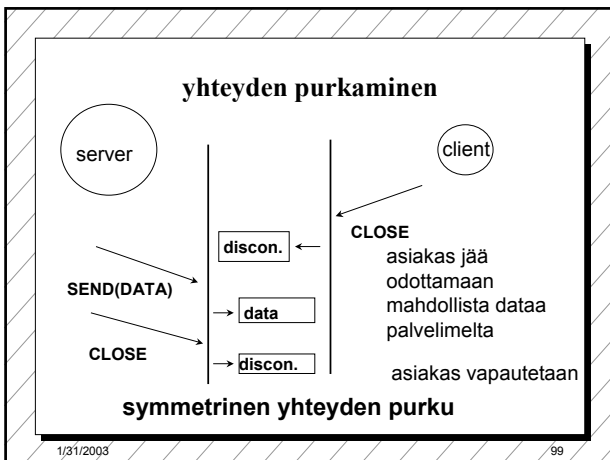
```

int send(int socket, char *message, int msg_len,
int flags)
sanoman lähetys annetun pistokkeen kautta

int recv(int socket, char *buffer, int buf_len, int
flags)
sanoma vastaanotto annetusta pistokkeesta ilmoitettuun puskuriin

```

1/31/2003 101



Pistokeohjelmointia Javalla

- `Socket clientSocket = new Socket("hostname", 6789);`
- `clientSocket.close();`
- `ServerSocket welcomeSocket = new ServerSocket(6789);`
- `Socket connectionSocket = welcomeSocket.accept();`
- (esimerkki kirjassa Kurose, Ross, Computer Networking, A Top-Down Approach Featuring the Internet)

1/31/2003 102

Pistokeohjelmointi

- Pistokeohjelmointia ja yleensä hajautettujen verkkosovellusten tekemistä opetellaan erillisellä kurssilla

- Verkkosovellusten toteuttaminen (järjestetään keväällä 2003)

1/31/2003

103

Yhteenveto

- Kuljetuserroksen palvelut

- UDP

- TCP

- luotettava tavuvirta

- yhteyden muodostus ja purku
- numerointi, tarkistussumma,
- kuittaus, uudelleenlähetykset, Go-back N
- vuonvalvonta: vastaanottoikkuna (liukuva ikkuna)
- ruuhkanhallinta: hidas aloitus

- pistokeohjelmointi

1/31/2003

104

