

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

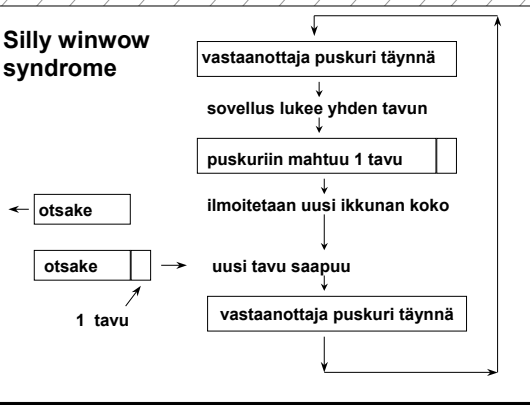
- ### Siirron optimointi
- TCP saa optimoida lähettämisiään
    - ei tarvitse lähettää heti kun data on tullut
    - dataa kerätään puskuuriin ja lähetetään sopivassa tilanteessa
    - PUSH-lipun avulla sovellus ilmoittaa, että data on lähetettävä heti
- 1/31/2003 63

- ### 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
    - kaiutetaan merkki vielä 41 tavun IP-paketilla
  - yhden merkin käsittely =>
    - 162 tavun siirtäminen
    - ja neljän segmentin lähettäminen
- 1/31/2003 64

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

- ### 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 winnow syndrome



1/31/2003

67

## 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

## TCP-otsakkeen kentät

Source port		Destination port	
Sequence number			
Acknowledgement number			
TCP head. length	URG	ACK	PSH
	RST	SYN	FIN
Checksum		Urgent pointer	
Options (0 or more 32 bit words)			
Data (optional)			

1/31/2003

69

## 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

## ■ 6 lippubittiiä

- URG onko pikadataa
  - pikadatan sijainnin ilmoittaa pikadatakenttä (Urgent pointer)
- ACK onko kuittauskenttä käytössä
- PSH onko hetilähetettävää (pushed) dataa
- RST yhteyden uudelleenaluspyyntö (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

- Ikkunan koko (window size)
  - vaihteleva ikkunankoko
  - 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

## ■ 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 uudelleenlähetyksiä

1/31/2003

74

## 3.6. TCP:n ruuhkan valvonta

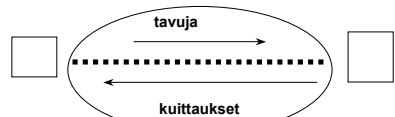
- Liikaa kuormitusta => verkko ruuhkautuu => hidastetaan lähettämistä
- Ruuhkan havaitseminen
  - nykyisin siirtovirheet harvinaisia
    - poikkeuksena langattomat verkot
  - => uudelleenlähetykset johtuvat ruuhkasta
    - uudelleenlähetyksajastimen laukeaminen on merkki ruuhkasta

1/31/2003

75

## ■ 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

## ■ Ruuhkaikkunan koko?

- Lähettäjän on itse pääteltävä ja arvioitava sopiva ruuhkaikkunan koko
  - kukaan muu ei sitä kerro!
  - uudelleenlähetyksajastin laukeaa => on ruuhkaa
  - kuittauksia 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

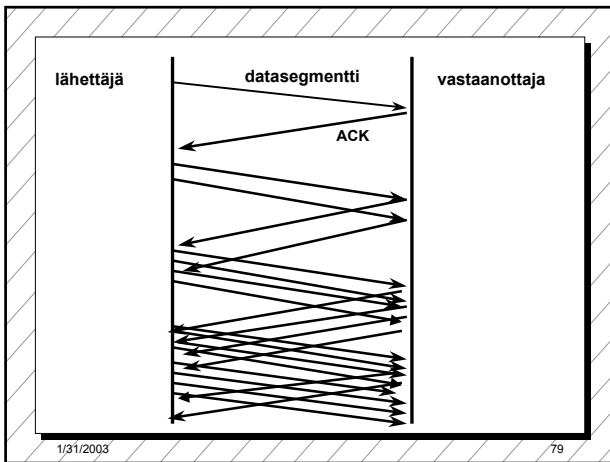
77

## 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



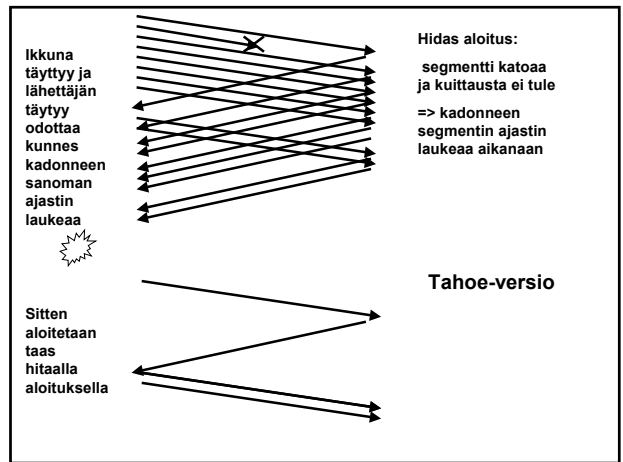
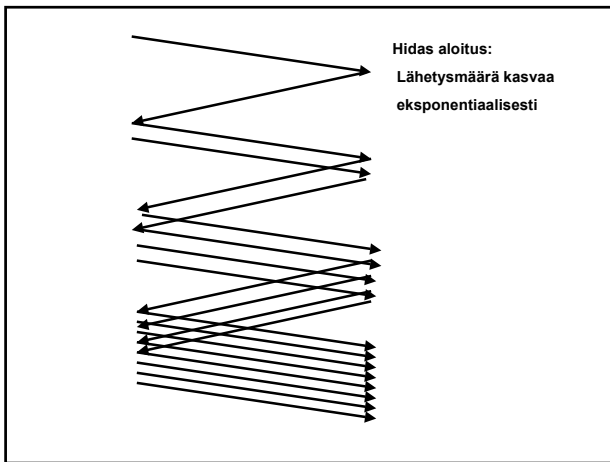
- **kynnysarvo (threshold)**
    - aluksi 64 K
    - 'varoitussarvo' = tästä lähtien syytä varoa ruuhkaa
  - kynnysarvoon saakka voidaan kasvattaa ruuhkaikkunaa eksponentiaalisesti
  - kynnysarvon saavuttamisen jälkeen kasvatetaan ruuhkaikkunaa vain lineaarisesti
    - = kasvatetaan kuittausten jälkeen vain yhdellä
    - edetään hyvin varovaisesti!
- 1/31/2003 80

- **jos ajastin ehtii laueta => ruuhkatilanne**
    - kynnysarvoksi 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
    - kynnysarvon saavuttamisen jälkeen kasvatetaan vain segmentti kerrallaan
    - kunnes taas havaitaan ruuhka ja aloitetaan ruuhkaikkunan uuden arvon etsiminen
- 1/31/2003 81

- ### Uudelleenlähetyksajastimen hallinta
- **uudelleenlähetyksajastin** (retransmission timer)
    - asetetaan aina kun segmentti lähetetään
    - ruuhkaa, jos kuittaus ei saavu ajoissa
  - **mikä on sopiva ajastimen aika?**
    - kuittaus aika vaihtelee suuresti
    - vaihtelu on myös nopeaa
  - **dynaaminen arvo**
    - saadaan jatkuvien verkon suorituskykymittauksien perusteella
- 1/31/2003 82

- **RTT**
    - arvio kiertoviiveelle (round-trip time)
    - mitataan jokaisen lähetetyn segmentin kiertoviive M
$$RTT = \alpha RTT + (1-\alpha)M, \text{ tyypillisesti } \alpha = 7/8$$
  - **uudelleenlähetyksajastimen arvo  $\beta RTT$** 
    - aluksi  $\beta$  oli aina 2
    - parannus: otetaan huomioon myös poikkeama D (deviation) oletetun ja saadun kiertoviiveen välillä  $|RTT-M|$
$$D = \alpha D + (1-\alpha)|RTT-M|$$
    - ajastimen arvo =  $RTT + 4 \cdot D$
- 1/31/2003 83

- **uudelleenlähetyksen vaikutus ajastimeen**
    - kumpaan segmenttiin kuittaus kohdistuu?
  - **Karnin algoritmi**
    - ei oteta huomioon uudelleenlähetyksen segmenttien kuittauksia RTT:n laskemisessa
- 1/31/2003 84



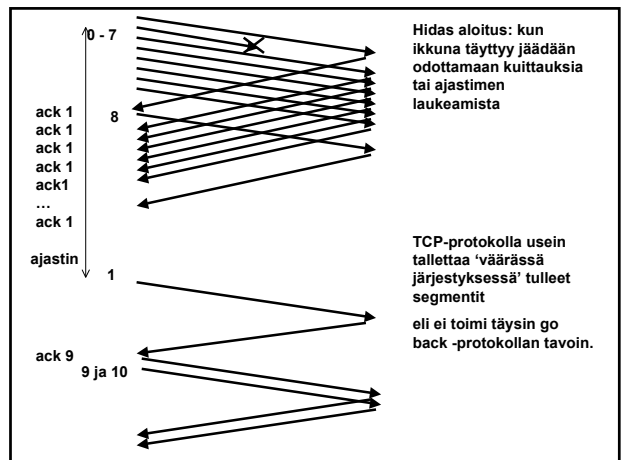
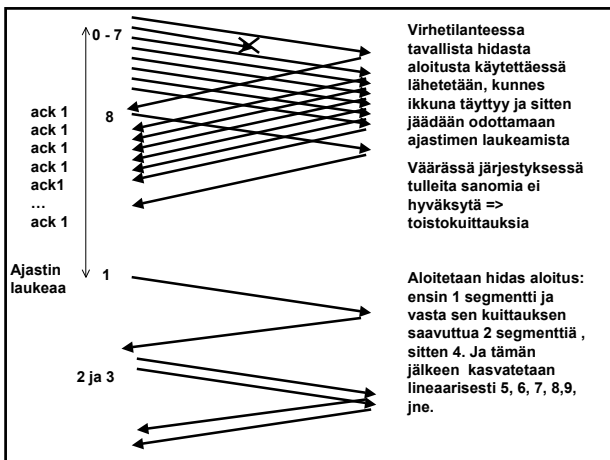
### Parannuksia ruuhkanvalvontaan

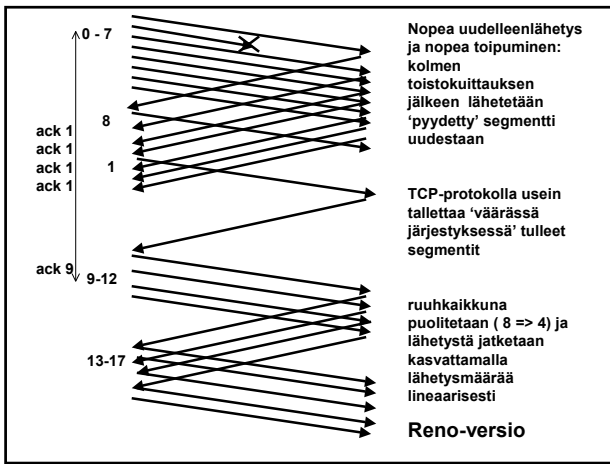
- **Nopea uudelleenlähetys (Fast Retransmit)**
  - ei odoteta ajastimen laukeamista ennen uudelleenlähetystä
  - vastaanottaja kuittaa jokaisen paketin
  - kun vastaanottaja huomaa puuttuvan paketin, se lähettää uudelleen edellisen paketin kuittauksen
    - Duplicate ACK (~ NAK)
  - kun lähettäjä saa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

1/31/2003 87

- **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





- **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

### TCP langattomassa verkossa

- **monet TCP-toteutukset optimoitu luotettaville lankaverkoille => suorituskyky langattomissa verkoissa erittäin huono**
  - ruuhkanvalvonta-algoritmi olettaa ajastimen laukeamisen johtuvan ruuhkasta
    - lähettämistä hidastetaan, jotta verkon kuormitus pieneneisi 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

### TCP-yhteyden hallinta

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

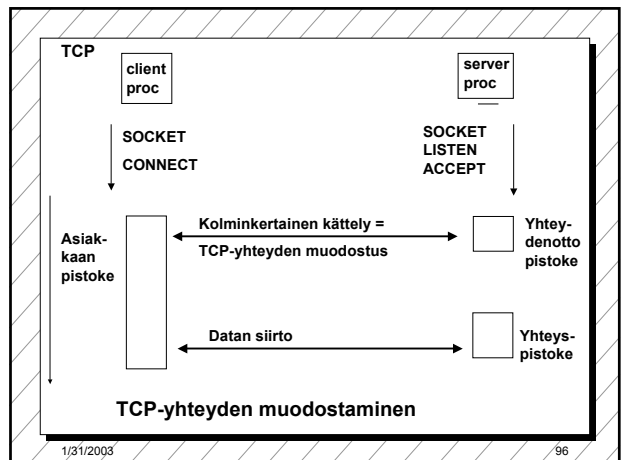
1/31/2003 94

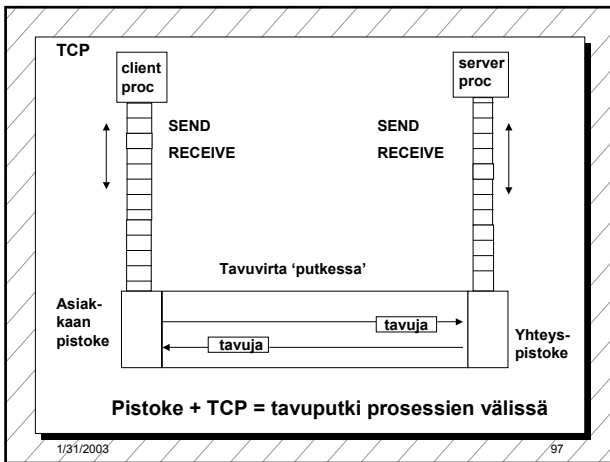
- **CONNECT-primitiivi**
  - parametreina
    - IP-osoite ja porttinumero
    - suurin hyväksyttävä segmentin koko
    - muuta tietoa, esim. salasana

↓

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

1/31/2003 95





■ **TCP-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

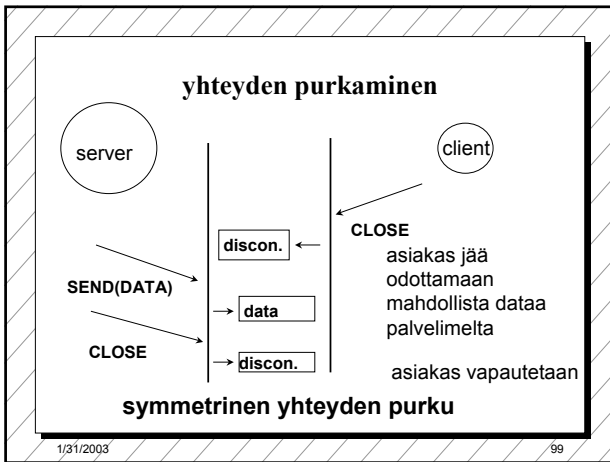
Segmentit lähetetään neljänä eri IP-pakettina

Ne luovutetaan vastaanottajalle yhdellä READ-kutsulla

neljä 512 tavun segmenttiä

yksi 2048 tavun data

1/31/2003 98



**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 (intsocket, struct sockaddr *address, int addr_len)
  
```

1/31/2003 100

```

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

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

1/31/2003 101

**Pistokeohjelmointia Javalla**

- Socket clientSocket = new Socket("hostname", 6789);
- clientSocket.close();
- ServerSocket welcomeSocket = new ServerSocket(6789);
- Socket connectionSocket = welcomeSocket.accept();
- accept()

(esimerkki kirjassa Kurose, Ross, Computer Networking, A Top-Down Approach Featuring the Internet)

1/31/2003 102

## Pistokehjelmointi

- Pistokehjelmointia ja yleensä hajautettujen verkko-sovellusten tekemistä opetellaan erillisellä kurssilla
- Verkkosovellusten toteuttaminen (järjestetään keväällä 2003)

1/31/2003

103

## Yhteenveto

- Kuljetuskerroksen palvelut
  - UDP
  - TCP
    - luotettava tavuvirta
      - yhteyden muodostus ja purku
      - numerointi, tarkistussumma,
      - kuittaus, uudelleenlähetys, Go-back N
      - vuonvalvonta: vastaanottoikkuna (liukuva ikkuna)
      - ruuhkanhallinta: hidas aloitus
- pistokehjelmointi

1/31/2003

104

