

## ■ kynnysarvo (threshold)

- 'varoitussarvo' = tästä lähtien syytä varoa ruuhkaa
- aluksi 64 K
- kynnysarvoon saakka voidaan kasvattaa ruuhkaikkunaa eksponentiaalisesti
- kynnysarvon saavuttamisen jälkeen kasvatetaan ruuhkaikkunaa vain lineaarisesti
  - = kasvatetaan kuittausten jälkeen vain yhdellä
  - edetään hyvin varovaisesti!

## ■ 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

## Uudelleenlähetyssajastimen hallinta

- uudelleenlähetyssajastin (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

## ■ RTT

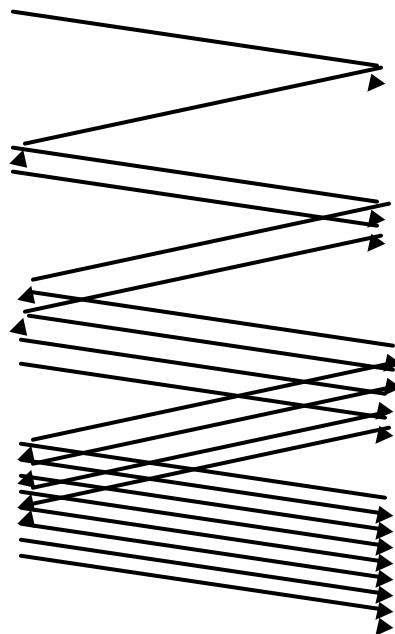
- arvio kiertoviiveelle (round-trip time)
- mitataan jokaisen lähetetyn segmentin kiertoviive  $M$   
 $RTT = \alpha RTT + (1-\alpha)M$ , tyypillisesti  $\alpha = 7/8$
- uudelleenlähetyssajastimen arvo  $\beta RTT$ 
  - aluksi  $\beta$  oli aina 2
  - parannus: otetaan huomioon myös poikkeama  $D$  (deviation)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

- uudelleenlähetysten vaikutus ajastimeen

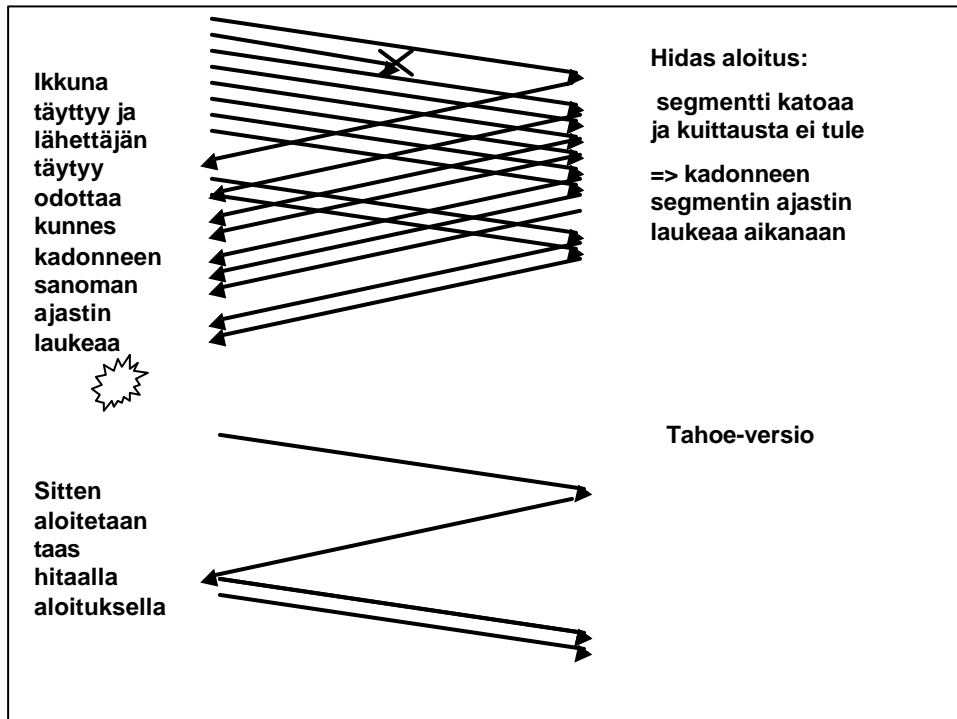
- kumpaan segmenttiin kuittaus kohdistuu?

- Karnin algoritmi

- ei oteta huomioon uudelleenlähetettyjen segmenttien kuittauksia RTT:n laskemisessa



Hidas aloitus:  
Lähetysmäärä kasvaa  
eksponentiaalisesti



## Parannuksia ruuhkanvalvontaan

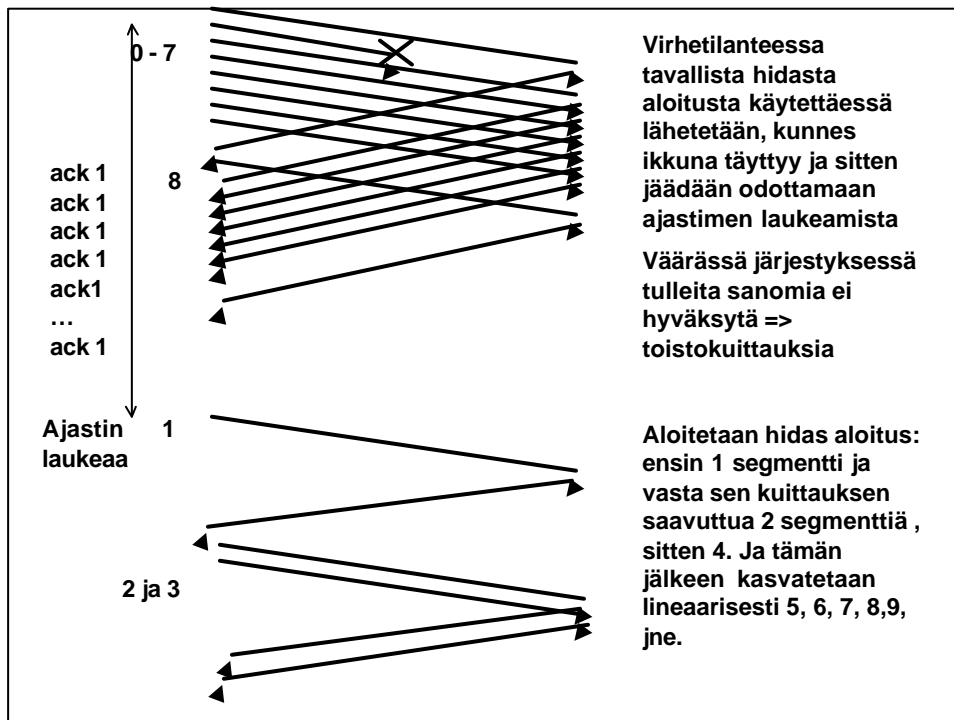
- **Nopea uudelleenlähetys** (Fast Retransmit)
  - ei odoteta ajastimen laukeamista ennen uudelleenlähetystä
  - vastaanottaja kuittaa jokaisen paketin
  - kun vastaanottaja huomaa puuttuvan paketin, se lähettää uudelleen edellisen paketin kuittauksen
    - Duplicate ACK (~ NAK)
  - kun lähettäjä saa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

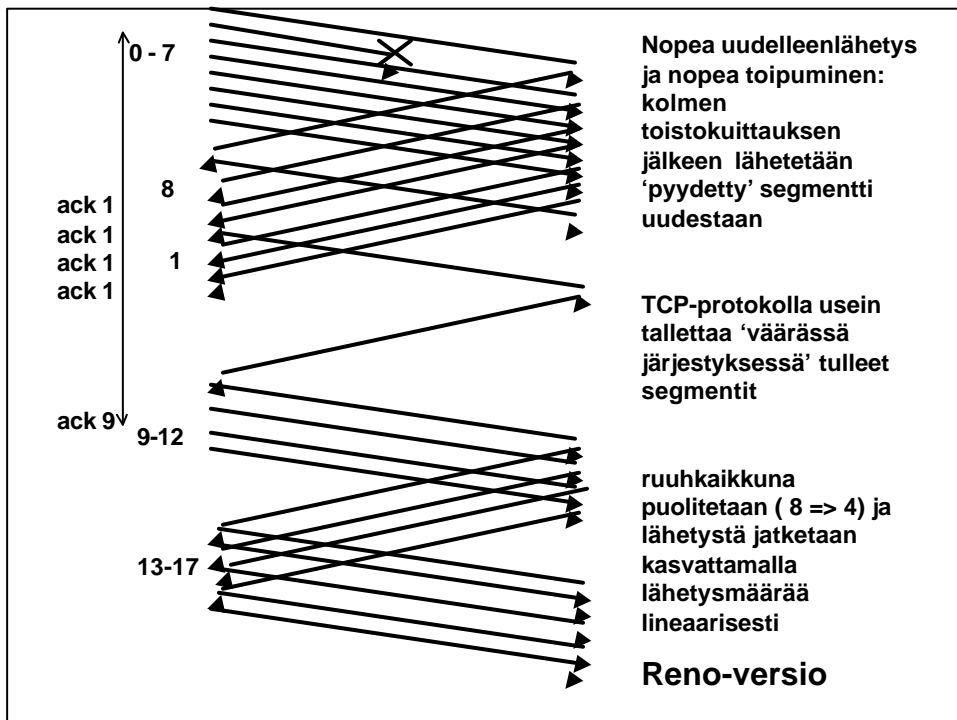
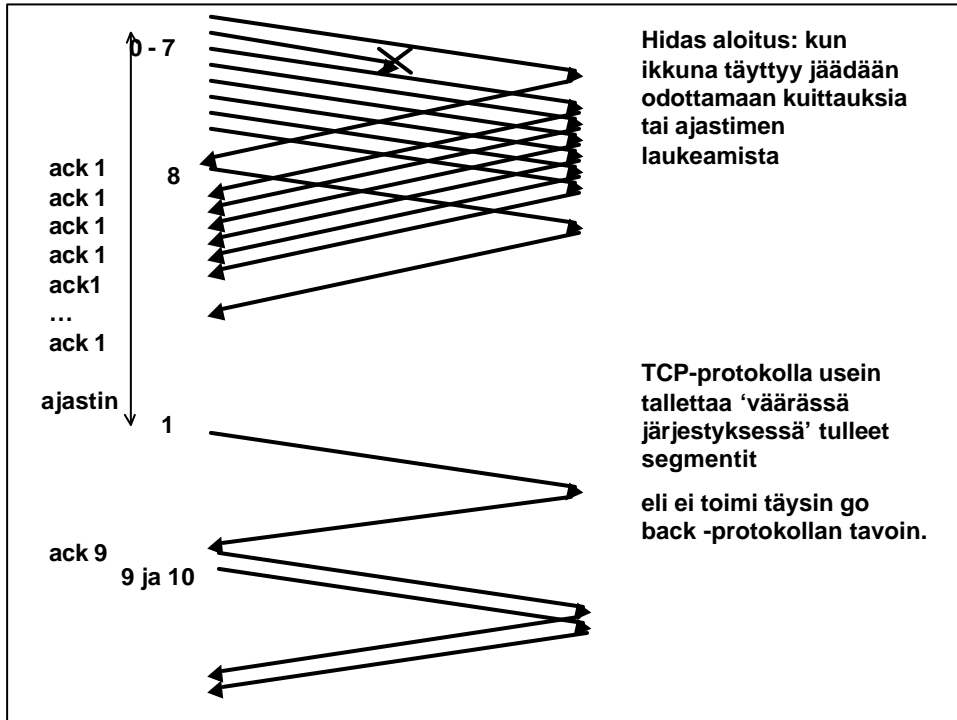
## ■ Nopea toipuminen (Fast Recovery)

- 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?

6.2.2002

87





- 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?

### 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 pienenee ja ruuhkaa ei syntyisi
  - langattomat yhteydet ovat epäluotettavia ja paketteja katoaa
    - kadonneet paketit syytä lähettää nopeasti uudelleen
    - lähetystä pitäisi päinvastoin nopeuttaa!

## TCP-yhteyden hallinta

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

6.2.2002

93

## ■ CONNECT-primitiivi

– parametreina

- IP-osoite ja porttinumero
- suurin hyväksyttävä segmentin koko
- muuta tietoa, esim. salasana



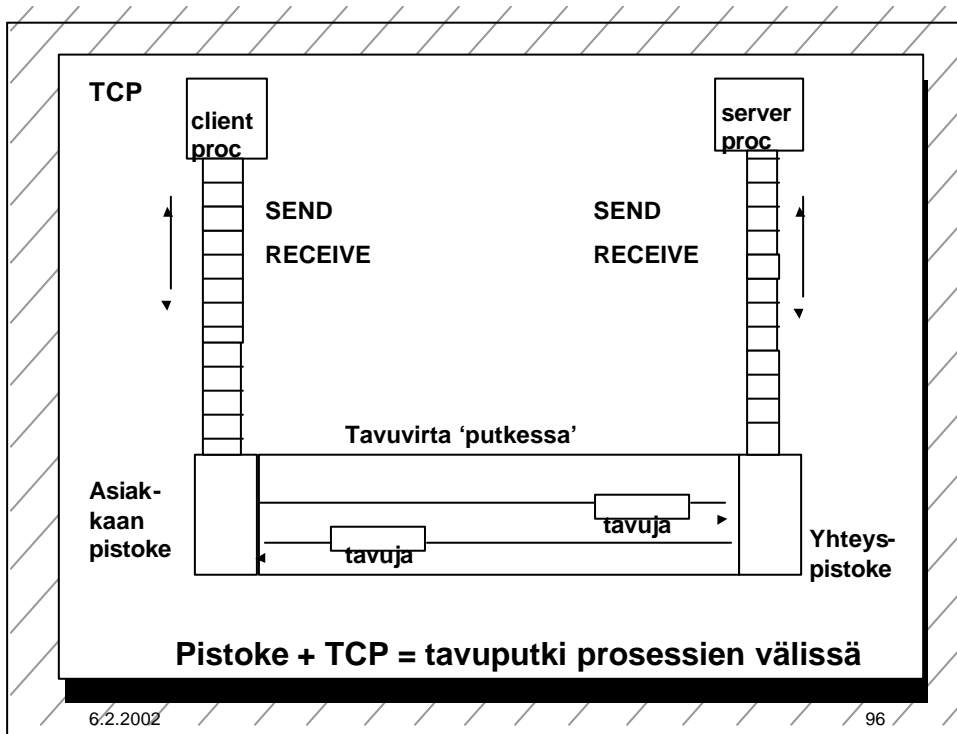
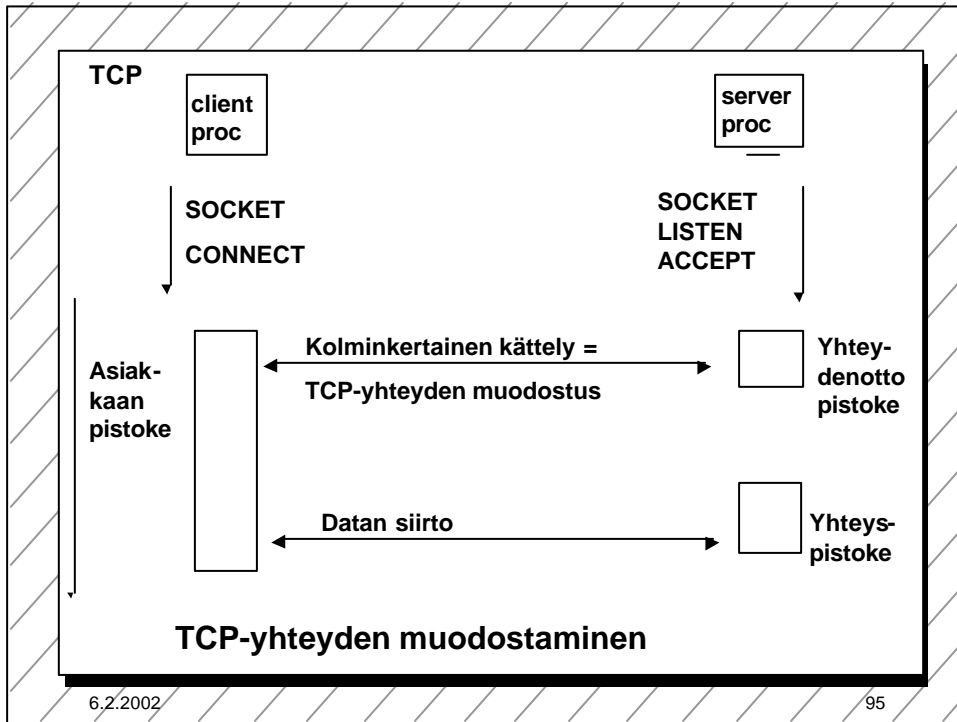
## ■ TCP-segmentti, jossa SYN-segmentti

- SYN = 1
- ACK = 0

6.2.2002

94





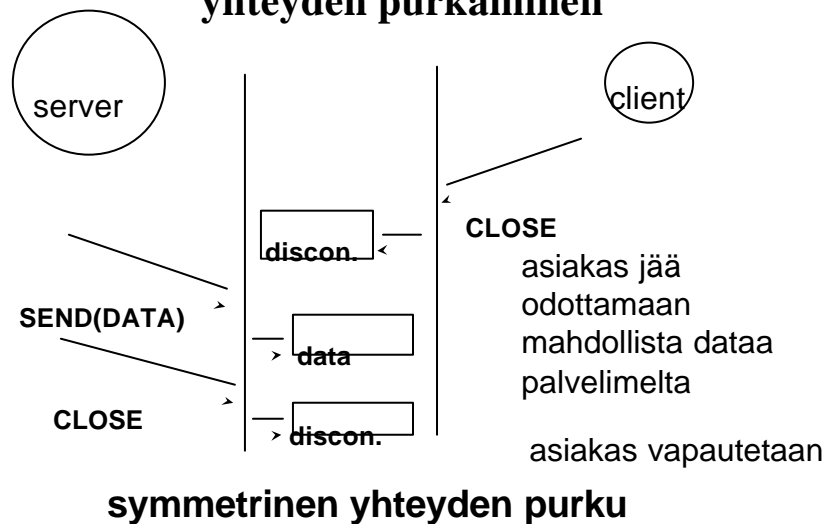
- TCP-yhteys on tavuvirtaa, ei sanomavirtaa
  - lähetettäessä neljä 512 tavun pätkää vastaanottaja saa joko
    - neljä 512 tavun pätkää
    - kaksi 1024 tavun pätkää
    - yhden 2048 tavun pätkän

Segmentit lähetetään neljänä eri IP-pakettina

Ne luovutetaan vastaanottajalle yhdellä READ-kutsulla



## yhteyden purkaminen



## 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)
```

```
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

## 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)

## Pistokeohjelmointi

- Pistokeohjelmointia ja yleensä hajautettujen verkkosovellusten tekemistä opetellaan erillisellä kurssilla
  - **Verkkosovellusten toteuttaminen**  
(järjestetään keväällä 2002)

# 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

- pistokeohjelmointi