

TCP

- ◆ TCP:n peruspiirteiden toiminta tarkemmin
 - osin vain harjoitustehtävissä
- ◆ TCP:n uusia piirteitä
 - SACK
 - Window scaling
 - time stamping
 - RED (Random Early Detection)
 - ECN (Explicit Congestion Notification)

TCP-otsakkeen kentät

Source port		Destination port	
Sequence number			
Acknowledgement number			
TCP Head. length	U	A	P
	R	S	F
Window size		Window size	
Checksum		Urgent pointer	
Options (0 or more 32 bit words)			
Data (optional)			

TCP-optiot

- ◆ Optio-kenttä
 - erilaisia valinnaisia piirteitä varten
 - Option pituus on 40 tavua
 - » data offset -kenttä = 4 bittiä kertoo otsakkeen pituuden 32 bitin sanoina => $15 \cdot 4$ tavua = 60 tavua
 - » 60 tavua -20 tavua vakio-otsaketta => enintään 40 tavua optioita varten
 - käytetään ilmoittamaan maksimi segmentin koko
 - muita uusia piirteitä
 - » aikaleimaus (timestamp)
 - » ikkunan skaalaus (window scaling factor)

Erilaisia suorituskykyongelmia

- ◆ TCP-protokolla käytössä hyvin erilaisissa ympäristöissä
 - » pitkään viipeen satelliittiyhteyksillä
 - » erittäin nopeilla yhteyksillä
 - » langattomilla yhteyksillä
- ◆ => suorituskykyongelmia
 - otsakkeen kentät liian pieniä
 - » ikkunan koko 16 bittiä
 - ◆ rajoittaa lähetyksenopeutta satelliittiyhteyksillä
 - » järjestysnumero 32 bittiä
 - ◆ rajoittaa lähetyksenopeutta erittäin nopeilla yhteyksillä

Ikkunan skaalaus (Window scale factor)

- ◆ ikkunan koko = 16 bittiä => 65536 tavua
 - » vuonvalvonnassa
 - » kertoo vastaanottajan ikkunan = kuinka monta tavua voi lähettää ennenkuin täytyy jättää odottamaan kuittausta
- ◆ jos käytössä ikkunan skaalaus -optio, ikkunakentän arvo kerrotaan 2^*F , jossa F on skaalausoption arvo.
 - » Suurin F:n arvo on 14
- ◆ käytetään vain yhteyden aloituspyynnössä

Aikaleima (timestamp)

- ◆ Kaksi eri optiota
 - Timestamp Value
 - » lähteissä segmenteissä,
 - Timestamp Echo Reply
 - » kuittauksessa
 - » sama kuin kuitatun segmentin Timestamp arvo
- ◆ Voidaan käyttää missä tahansa datasegmentissä
- ◆ => kiertoviiveiden jatkuva tarkkailu helpompaa

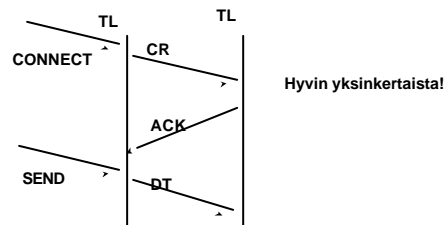
Yhteyden muodostus

- ◆ perusmalli hyvin yksinkertainen
- ◆ ongelmana viivästetyneet kaksoiskappaleet
 - » esim. yhteys pankkiin laskun maksamiseksi
 - » lasku maksetaan useaan kertaan
- ◆ =>yksikäsitteisen yhteyden muodostaminen vaikeaa

12.2.2001

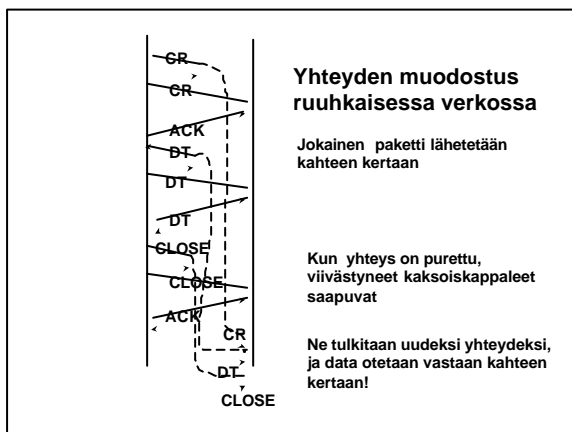
7

Yhteydenmuodostus: perusmalli



12.2.2001

8



Ongelman ratkaisuehdotuksia:

- ◆ kertakäyttöiset kuljetusosoitteet
 - » nimipalvelu?
- ◆ yhteystunnus jokaiselle yhteydelle
 - yhteyden purkamisen jälkeen sen TPDU:t epäkelvoja
 - » lista epäkelvoista yhteystunnuksista
 - kuinka kauan historiatietoja säilytettävä?
 - entä jos kone kaatuu ja unohtaa tietonsa?
- ◆ **rajallinen elinikä paketeille**
 - » elinaikalaskuri, hyppylaskuri

12.2.2001

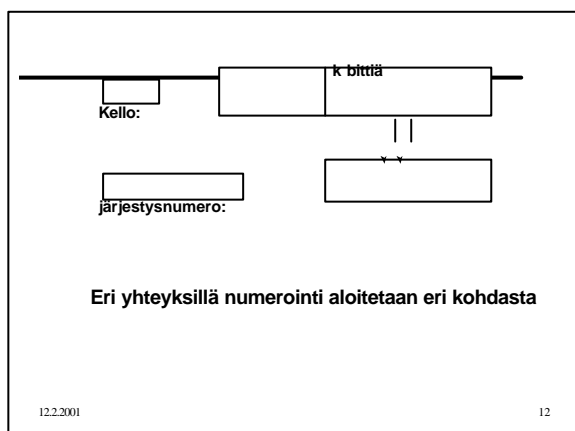
10

Tomlinsonin menetelmä

- ◆ koneessa vikasietoinen kello
 - » etenevä laskuri
 - » vaikka kone kaatuu, laskuri toimii
- ◆ yhteyttä muodostettaessa
 - » kellon bitit ilmoittavat numeroinnin aloituskohdan
 - » bittejä riittävästi (32 bittiä)
 - ◆ jotta uudelleen käyttöön vasta riittävän pitkän ajan päästä
 - ◆ vanhoilla numeroilla varustetut segmentit ehtivät hävitä
- ◆ yhteydellä ei koskaan kahta saman numeroista segmenttiä

12.2.2001

11



12.2.2001

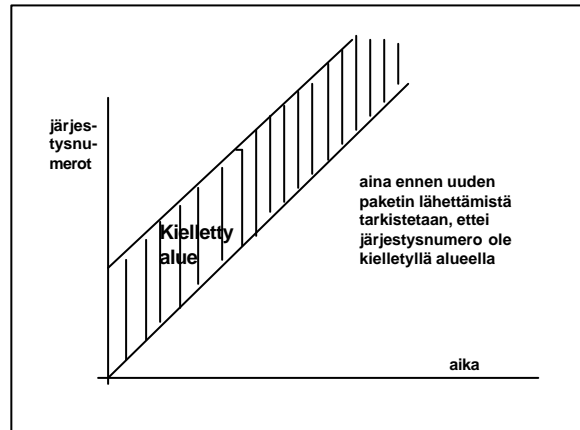
12

◆ **Kun kone kaatuu**

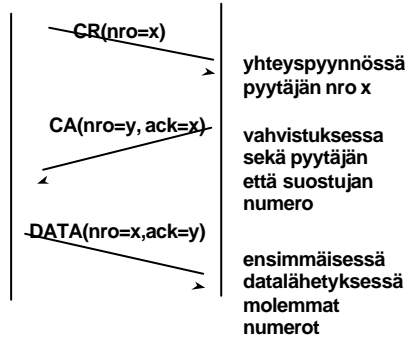
- » se kadottaa tiedon viimeksi käytetystä järjestysnumerosta
- » uusi numero ei saa olla sama kuin jonkun vielä elossa olevan TPDU:n

◆ **Miten selvittää tilanteesta?**

- odotetaan T aikayksikköä
 - » kaikki aikaisemmat TPDU:t varmasti kadonneet ja voidaan aloittaa, mistä numerosta tahansa
 - » entä, jos T on pitkä
- 'kielletyn alueen' (forbidden region) käyttö
 - » ei oteta käyttöön numeroita, joiden duplikaatit voivat olla vielä elossa



Kolminkertainen kättely

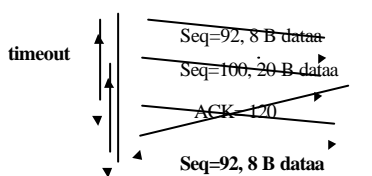


Kuittaukset TCP:ssä

- TCP käyttää kumulatiivista kuittausta
 - kuittaus varmistaa lähettäjälle, että kaikki segmentit kuitattuun segmenttiin saakka ovat saapuneet kunnolla perille
 - väärässä järjestyksessä saapuneita segmenttejä ei kuitata
 - ei käytetä NAK-kuittausta
 - "duplicate ACK" = virheitilanteissa lähetetään uudelleen kuittaus samasta jo kuitatusta segmentistä

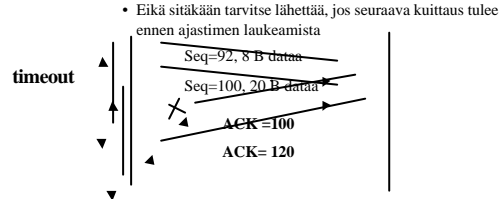
Uudelleenlähettäminen

- TCP lähettää segmentin uudestaan, kun ajastin laukeaa
 - TCP ei automaattisesti lähetä kaikkia puuttuvan segmentin jälkeisiä segmenttejä uudelleen



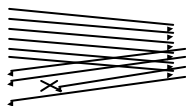
Vain osin "Go-Back-N"-tyyppinen

- Oletetaan, että segmentit 1-N tulevat oikein perille ja kuittaus esim. segmenttiin 1 katoaa. Jos muut kuittaukset tulevat perille, enintään yksi segmentti 1 uudelleenlähetetään.



Kuittaukset voivat kadota

- ◆ Erilliset kuittaukset eli pelkät ACK:it eivät sisällä yhtään tavua dataa, joten niitä ei numeroida eikä kuitata.
 - Kuittauksia voi helposti hävitä



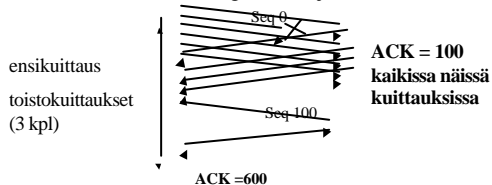
Kumulatiivisissa kuittauksissa seuraava kuittaus paikkaa hävinneen informaation

“Duplicate Ack”

- ◆ ensikuittaus (first-time ACK)
 - segmentin ensimmäinen kuittaus
 - tähän saakka kaikki on kunnossa
- ◆ toistokuittaus (duplicate ACK)
 - vastaanottaja kuittaa viimeksi saatua hyväksyttyä segmenttiä aina kun saa virheellisen tai väärässä järjestyksessä tulevan segmentin
 - NAKin korvike, jolla ilmoitetaan ongelmista lähettäjälle

Nopea uudelleenlähetys (Fast retransmit)

- ◆ Kun lähettäjä vastaanottaa 3 toistokuittausa samalle segmentille, se lähettää heti puuttuvan segmentin uudestaan
 - eikä odota segmentin ajastimen laukeamista

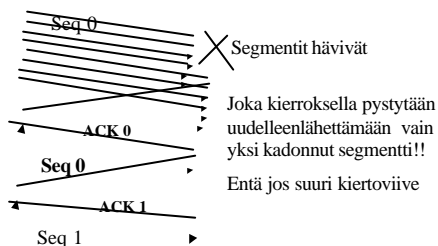


Viivästetty ACK (Delayed ACK)

- Ei tarvitse välttämättä kuitata jokaista segmenttiä
 - kuitenkin kuitattava ainakin joka toinen ja viive saa olla korkeintaan 500 ms,
 - ◆ usein noin 200ms
- Hyöty: kuittaus kulkee datan mukana
 - samalla kertaa ikkunan muutos, kuittaus ja kaiutus
- Haitta: kiertoviiveen laskeminen, pakettien kellotus

Suorituskykyongelmia!

- ◆ Kun useita segmenttejä katoaa ‘samasta ikkunasta’



SACK (Selective Acknowledgement)

- RFC 2018
TCP Selective Acknowledgement Options.
M. Mathis, J. Mahdavi, S. Floyd, A. Romanow. October 1996.
(Status: **PROPOSED STANDARD**)

INTERNET DRAFT
Mark Allman, Ethan Blanton. "A Conservative SACK-based Loss Recovery Algorithm for TCP".
(draft-allman-tcp-sack-02.txt), January, 2001

- ◆ Kumulatiivinen kuittaus paljastaa aina vain yhden puuttuvan kerrallaan

- ◆ SACK paljastaa kaikki puuttuvat
 - » ilmoittamalla, mitkä segmenttivälit on jo vastaanotettu

- ◆ Esim. Segmentin koko 1000 tavua
 - 1. segmentti katoaa ja muut tulevat perille
 - ◆ segmentin 2 kuittaus: ACK 0, 1000: 2000
 - ◆ segmentin 10 kuittaus: ACK 0, 1000: 10000
 - 1. ja 3. segmentti katoavat
 - ◆ segmentin 10 kuittaus:
 - ◆ ACK 0, 1000:2000 3000:10000

SACK-optiot

- ◆ SACK- permitted yhteyden muodostuksessa eli vain SYN-segmentissä ilmoittamaan, että yhteydellä voidaan käyttää SACK-kuittauksia
 - ◆ (type = 4, length = 2)
- ◆ SACK-optio
 - kuljettaa lisäinformaatiota saapuneista segmenteistä eli kertoo, mitkä 'tavupätkät' ovat jo valmiina vastaanottajan puskurissa
 - kuljetetaan TCP-segmentin optio-osassa

TCP:n SACK-optio

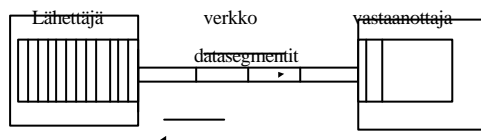


4 lohkoa mahtuu yhteen TCP-segmenttiin, jossa optiolle on varattu 40 tavua, jos ei käytetä muita optioita kuten aikaleimaa (timestamp).

Vain neuvoo-antava!

- ◆ Ohjeellista tietoa lähettäjälle
 - vastaanottaja voi tarvittaessa poistaa SACK-optiossa ilmoittamia tavuja puskureistaan
- ◆ Jos vastaanottaja käyttää SACK-optiota, niin sitä on käytettävä aina kun vastaanottajalla on puskureissaan epäjärjestyksessä olevaa dataa
 - tällöin kaikissa ACK:ssa on oltava ajantasalla oleva tieto siitä, mitkä tavut on jo puskureissa

TCP ruuhkanvalvonta



ACK toimii lähetyksen tahdistajana putkesta poistunut dataa, joten voidaan lähettää sama määrä lisää

TCP self-clocking

- ◆ TCP tahdistaa itse oman lähetyksensä ACK:ien avulla
 - nopeutta voi rajoittaa
 - » verkko
 - ◆ ruuhkan takia syytä vielä pienentää lähetyksenopeutta
 - » vastaanottaja
 - ◆ lähetyksenopeus ok
 - lähettäjä ei voi tietää kumpi