

Ruuhkanvalvonta on hankalaa!

- ◆ Sitä varten on koko ajan kehitetty yhä parempia menetelmiä
 - uudelleenlähetyssajastimen arvo
 - » RTT:n varianssin arviointi
 - » Karnin algoritmi
 - » exponential retransmission timer backoff
 - lähetyssikkunan hallinta
 - » slow start
 - » congestion avoidance
 - » fast retransmit
 - » fast recovery

Lähetettynä voi olla vain rajallinen määrä kuittaamatonta dataa ('Flight size')

- ◆ vastaanottoikkuna (receiver window, **rwnd**)
 - vastaanottaja ilmoittaa lähettämiensä segmenttien ikkunakentässä
 - vastaanottaja voi vapaasti kasvattaa tai pienentää
 - vuonvalvontaa varten
- ◆ ruuhkaikkuna (congestion window, **cwnd**)
 - lähettäjä saa korkeintaan lähettää verkkoon, jotta verkko ei tukkeutuisi
 - ruuhkanhallintaa varten
- ◆ **min(rwnd, cwnd)** rajoittaa lähettämistä

Ruuhkaikkunan arvo eri tilanteissa

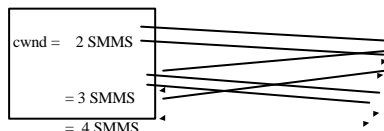
- initial window (IW)
 - » ruuhkaikkunan arvo heti kolminkertaisen kättelyn jälkeen
 - ◆ korkeintaan kaksi segmenttiä tai 2* suurin määrä tavuja, jonka lähettäjä voi kerralla lähettää (SMSS)
- loss window (LW)
 - » ikkunan arvo, kun TCP on havainnut, uudelleenlähetyssajastimen lauettua, segmentin kadonneeksi
- restart window (RW)
 - » kun lähetyks käynnistetään uudelleen joutilaana olon jälkeen

SMSS (sender Maximum Segment Size)

- ◆ Suurin segmentti, jonka lähettäjä saa lähettää
 - ilman otsake- ja optiotavuja
 - » pelkästään datatavut mukana
 - voi olla verkon rajoitus, reitin rajoitus, vastaanottajan yhteydenmuodostuksessa ilmoittama rajoitus tai oletusarvoinen pienin segmentin maksimikoko (=536 tavua dataa)

Slow start

- ◆ Hitaan aloituksen aikana
 - Ruuhkaikkunaa cwnd kasvatetaan korkeintaan maksimilähetysmäärällä (SMSS) jokaista uutta dataa kuittaavaa ACKia kohden



Limited Transmit

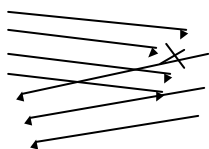
- ◆ RFC 3042: Enhancing TCP's Loss Recovery Using Limited Transmit.

M. Allman, H. Balakrishnan, S. Floyd. January 2001
(Status: PROPOSED STANDARD)

- ◆ Lähettäjä ei saa kolmea toistokuittausta =>
 - odotettava aina ajastimen laukeamista ja
 - suoritettava hidas aloitus
 - => hidastaa usein turhaan lähettämistä

Lähetäjä ei saa kuittauksia,

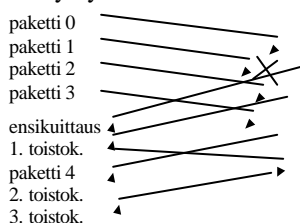
- ◆ Jos ruuhkaikkuna on hyvin pieni,
 - ei voi tulla kolmea toistokuittausta, jos ruuhkaikkuna sallii vain neljä kuittaamatonta lähetystä



Ratkaisu:

- ◆ Kun lähettäjä saa toistokuittauksen, se saa aina lähettää yhden **uuden paketin** verkkoon
 - » kuittaus kertoo, että verkosta poistettu paketti, joten verkkoon siis mahtuu!
- ◆ Kun saman paketin toistokuittauksia tulee kolme, niin suoritetaan nopea uudelleenlähetys ja nopea toipuminen (fast recovery)

- ◆ Vaikka ruuhkaikkuna on pieni, niin rajoitetulla lähetyksellä saadaan tarvittaessa syntymään kolme toistokuittausta



Miksi lähetetään uusi paketti?

- ◆ Miksi ei heti ensimmäisen toistokuittauksen jälkeen lähetä uudestaan sitä jo lähetettyä kuittaamatonta pakettia?
- ◆ Koska ei vielä olla varmoja siitä, että paketti on todella kadonnut.
 - Se voi olla vain viivästynyt
 - tai paketit ovat matkalla joutuneet väärään järjestykseen
- ◆ => näin vältetään turhia uudelleenlähetystsiä

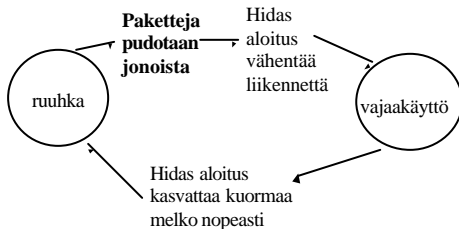
RED (Random Early Detection)

- ◆ Aktiivinen, ennaltaehkäisevä puskurijonon hallinta parantaa TCP:n suorituskykyä
 - proaktiivinen <=> reaktiivinen
- ◆ Ongelma:
 - Kun ruuhka on syntymässä, puskurien jonot kasvavat ja lopulta puskurit täyttyvät ja paketteja joudutaan hävittämään
 - » yleensä 'pudotetaan' viimeksi saapuvat paketit
 - ◆ tail-drop
 - tässä vaiheessa usein poistetaan paljon paketteja monelta lähettäjältä

Globaalin tahtiituksen ongelma (Global Synchronization)

- ◆ Samanaikaisesti usean TCP-lähetyksen uudelleenlähetysajastin laukeaa ja useat TCP:t vähentävät hyvin voimakkaasti lähettämistään hitaan aloituksen takia.
- ◆ => verkon vajaakäyttöisyys
- ◆ lähettäjät kasvattavat lähettämistään hitaassa ajoituksessa melko nopeasti ja jossain määrin samassa tahdissa
- ◆ => ruuhka verkossa

Verkon suorituskyky on huono!



Oskilloidaan koko ajan ruuhkan ja vajaakäytön välillä eikä yhteyksillä saavuteta tasaista kuittausien tahdistamaa 'lähetysputkea'.

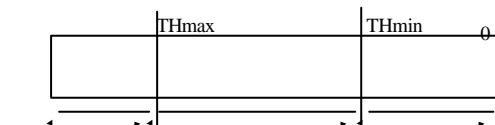
Tehokkaampi puskurijonojen hallinta

- ◆ Puskureiden koon kasvattaminen ei ratkaise ongelmaa.
 - Miksi ei?
- ◆ Pitää ennakoida ruuhkatilanteen kehittyminen ja reagoida siihen ennenkuin tilanne ehtii niin pahaksi, että joudutaan poistamaan paljon paketteja samalla kertaa.
- ◆ => Aktiivinen puskurijonon hallinta => RED

Miten RED toimii?

- ◆ Jonon pituutta tarkkaillaan koko ajan
 - aina kun jonoon tulee paketti, niin
 - » jos jonon pituus < minimi kynnyspituus, paketti laitetaan jonoon
 - » jos jonon pituus >= maksimi kynnyspituus, paketti hävitetään
 - » jos jonon pituus minimi ja maksimi kynnyksarvojen välissä, niin todennäköisyydellä P paketti hävitetään ja todennäköisyydellä 1-P laitetaan jonoon
 - » hävittämistodennäköisyys kasvaa, kun jononpituus kasvaa

Jonon pituuden vaikutus



RED-puskuri

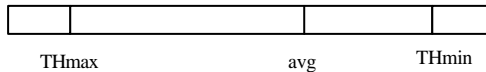
- ◆ Pyritään pitämään jonon pituus koko ajan annetuissa rajoissa hävittämällä paketteja kun jonon pituus kasvaa
 - eli voi tulla ruuhka
- ◆ Hävittämistodennäköisyys kasvaa, kun jono kasvaa.
- ◆ Hävittämistodennäköisyys kasvaa, kun paketteja ei ole hävitetty

Kun paketti saapuu FIFO-tyyppiseen ulostulojonoon.

- ◆ Lasketaan keskimääräinen jononpituus painotettuna keskiarvona aikaisemmista jononpituuksista
- ◆ $avg <- (1-wq)*avg + wq * q$
- ◆ jos jonon pituus $q = 0$
 - $m <- f(time - q_time)$
 - $avg <- (1-wq)**m * avg$
 - arvioidaan, kuinka monta pikkupakettia (= m) olisi voitu siirtää sinä aikana, jonka jono on ollut tyhjänä
- ◆ $wq \sim 0.002 \Rightarrow avq$ reagoi hitaasti jonon pituuden muutoksiin

Hävittämistodennäköisyyden laskeminen

- ◆ Jos jononpituus avq on välillä $Thmin$ ja $Thmax$, on laskettava hävittämistodennäköisyys P
- ◆ mitä pitempi jono, sitä suurempi P
 - $P_b = P_{max} (avq - Thmin) / (Thmax - Thmin)$



- ◆ Suositus $P_{max} = 0.02 \Rightarrow$ kun $avq = 1/2(Thmax - Thmin)$, niin 2 pakettia sadasta hävitetään.

- ◆ Lisäksi pyritään tekemään hylkäämiset tasavälein

– estämään hylkäysryöpyt
| ||| ||| | ||| |||

- ◆ mitä pitempään ei ole hylätty yhtään, sitä suurempi hylkäystodennäköisyys
 - avq :n ollessa $Thmin$ in ja $Thmax$ in välissä muuttuja **count** laskee peräkkäisiä hylkäämättömiä paketteja
 - $Pa < Pb / (1 - count * Pb)$
 - $Pa =$ hylkäämiskriteerinä käytetty todennäköisyys (P)

- ◆ ruuhkan estämiseen
- ◆ erityisesti globaalien tahdistumisen estämiseen
- ◆ ryöppyisen liikenteen sorsiminen estämiseen
 - » liikenneryöpyt usein ruuhkan syy
 - » ryöppyisen liikenteen lähteet joutuvat kokemaan pakettien hylkäämistä tasaisen liikenteen lähteitä enemmän
- ◆ rajoittaa keskimääräistä jononpituutta \Rightarrow rajoittaa keskimääräistä viivettä

ECN (Explicit Congestion Notification)

- ◆ **The Addition of Explicit Congestion Notification (ECN) to IP**,
K. K. Ramakrishnan, Sally Floyd, D. Black.
(draft-ietf-tsvwg-ecn-01.txt), January 2001.
(STATUS: INTERNET DRAFT)
- ◆ **TCP and Explicit Congestion Notification**.
ACM Computer S. Floyd., Communications Review, 24,
October 1994

Lisäys IP arkkitehtuuriin!

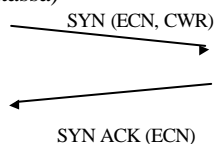
- ◆ Käyttöön 2 bittiä, joita käytetään ruuhkasta ilmoittamiseen
 - CE-bitti (Congestion Experienced)
 - » reititin asettaa, kun on havainnut ruuhkaa
 - » esim. kun RED-algoritmin jononpituus indikoi ruuhkaa
 - ECT-bitti (ECN Capable Transport)
 - » kertoo, että kuljetuskerros kykenee käsittelemään ruuhkailmoituksia (Explicit Congestion Notification)
- IPv4: TOS-kentän bitit 7 ja 6 ehdotettu
- IPv6: Traffic Class -kentän bitit 7 ja 6 ehdotettu

Muutoksia TCP:hen

- ◆ Sovittava vastapuolen kanssa ECN:n käytöstä
- ◆ osattava reagoida CE-bitin asetukseen
 - vastaanottajan ilmoitettava lähettäjälle
 - » ECN Echo -lippu
 - lähettäjän vähennettävä lähetystään samalla tavalla kuin jos paketti olisi todella kadonnut
 - lähettäjän ilmoitettava vastaanottajalle, että on vähentynyt jo lähettämistään
 - » Congestion Window Reduced (CWR) -lippu

Yhteydenmuodostus

- ◆ Yhteyden muodostusvaiheessa sovitaan ECN:n käytöstä käyttäen ECN Echo -lippua (bitti 9 TCP-otsakkeen varattukentässä)



- ◆ Kun käytöstä on sovittu, lähetettyjen IP-pakettien ECT-kenttä on asetettu
 - » jos ei ole asetettu, ei vastaanottajan tule reagoida
- ◆ kun vastaanottaja saa ruuhkasta ilmoittavan IP-paketin, sen tulee kuittauksessa asettaa ECN Echo -bitti
- ◆ CE-bittejä asetetaan kuittauksiin niin kauan, kunnes saadaan paketti, jossa CWR-bitti on asetettu

- ◆ Kun lähettäjä saa kuittauksen, jossa ECN Echo -bitti on asetettu, niin sen tulee
 - puolittaa ruuhkaikkuna
 - pienentää hitaan aloituksen lopettamisen kynnyksarvoa
- ◆ toiminta sama kuin paketin kadotessa
- ◆ vähennys korkeintaan kerran yhden kiertoviiveen aikana

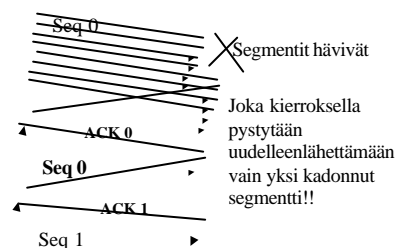
NewReno

- ◆ RFC 2581: **TCP Congestion Control**
M. Allman, V. Paxson, W. Stevens
April 1999 (Obsoletes RFC 2001)
(Status: PROPOSED STANDARD)
- ◆ RFC 2582: **The NewReno Modification to TCP's Fast Recovery Algorithm**
S. Floyd, T. Henderson, April 1999
(Status: EXPERIMENTAL)

Nopea toipuminen (Fast Recovery)

- ◆ Toteutettiin ensimmäisen kerran 1990 Reno-versioon
- ◆ Ei toimi hyvin, jos useita paketteja katoaa
 - tarvitaan kiertoaika jokaista kadonnutta pakettia kohden
- ◆ eräs ratkaisu on SACK-optio
 - kuittauksessa ilmoitetaan, mitkä saatu kunnolla ja mitkä vielä puuttuvat

- ◆ Kun useita segmenttejä katoaa 'samasta ikkunasta'



NewReno -ratkaisu

- ◆ Kun saadaan kolmas toistokuittaus, niin
 - asetetaan kynnyisarvo $ssthresh = \max(\text{FlightSize} / 2, 2 * MSS)$
 - ja talletetaan viimeksi lähetetty järjestysnumero muuttujaan "recover"
 - Lähetetään puuttuva segmentti ja asetetaan ruuhkaikkunan arvoksi $cwnd = ssthresh + 3 * MSS$
 - kaikki vielä tulevat toistokuittaukset kasvattavat ruuhkaikkunaa yhdellä MSS:lla

- ◆ Lähetetään segmentti
 - mikäli ruuhkaikkuna ja vastaanottajan ikkuna tämän sallii
- ◆ Kun segmenttiin saadaan kuittaus,
 - se joko kuittaa kaikki recover -muuttujan ilmoittamaan arvoon asti => toipuminen suoritettu loppuun
 - tai kuittaus on johonkin aikaisempaan järjestysnumeroon
 - » osittainen kuittaus (partial ACK)

Kun tulee osittainen kuittaus

- ◆ Lähetetään uudelleen ensimmäinen kuittaamaton segmentti,
- ◆ kasvatetaan ruuhkaikkunaa kuitatuilla ja vähennetään siitä juuri lähetetty
- ◆ lähetetään uusia segmentteja, jos ruuhkaikkuna ja vastaanottajan ikkuna tämän sallii
- ◆ ja jatketaan toipumisvaihetta

Eri versioita

- ◆ Pitäisikö kuitenkin uudelleenlähettää kerralla useampi kuin yksi?
- ◆ Miten ajastin tulisi parhaiten asettaa kun uudelleenlähetetään segmentti?
- ◆ Yms.
- ◆ Nyt ollaan jo melko kaukana itse standardista. Nämä ovat avoimia tutkimuskysymyksiä!