S-38.201 ATM JA MULTIMEDIA SEMINAARI, KEVÄT -97

ABR- ja TCP-vuonohjauksen yhteensopivuus

Timo Paajanen

S 39296E

timo.paajanen@ntc.nokia.com

TIIVISTELMÄ

TCP/IP-liikenne on ehdotettu välitettäväksi ATM-verkossa joko ABR- tai UBR-palvelun avulla. Käytettäessä ABR-palvelua voidaan vuonohjauksen avulla käytössä oleva kapasiteetti jakaa reilusti käyttäjien kesken ja saadaan solujen häviösuhde (CLR) pieneksi. Ongelmana on kuitenkin ATM-kerroksen ja kuljetuskerroksen vuonohjauksen päällekkäisyys. TCP:n sallima lähetysikkuna ja ABR:n eksplisiittinen nopeus eivät yleensä vastaa toisiaan. UBR.n avulla TCP liikenne siirretään ns. 'best effort' -liikenteenä, jolloin seurauksena on usein solujen katoamisia ja TCP-pakettien uudelleenlähettämisiä.

1. JOHDANTO

ATM-Forum on määritellyt ABR-palvelun, jonka avulla pyritään välttämään ruuhkatilanteet ja hyödyntämään korkeampi prioriteettiselta CBR- ja VBR-liikenteeltä yli jäänyttä verkon kapasiteettia. ABR-palvelussa on määritelty vuonohjaus, joka ohjaa lähteitä verkon kuormitustilanteen perusteella. Nopean vuonohjauksen avulla taata käyttäjille pieni solujen häviämistodennäköisyys. ABR-palvelun ongelmana ovat lähinnä monimutkainen toteutus ja palvelua tehokkaasti hyödyntävien sovellusten (native ATM-sovellusten) puute. Useimmat datasovellukset käyttävät nykyisin TCP/IP-protokollaa. Kuljetuskerroksen TCP takaa tiedon virheettömän välityksen ja huolehtii siitä, että tieto menee varmasti perille. Ruuhkien ehkäisemiseksi TCP:ssä käytetään vuonohjausta. Haluttaessa siirtää TCP/IP-liikennettä ATM-verkon yli käytetään ATM-kerroksella joko ABR- tai UBR-palveluluokkaa. UBR:ssä ongelmana on tiedonsiirron epävarmuus, joka aiheuttaa uudelleenlähetyksiä. ABR:n ja TCP:n yhteistoiminnassa on ongelmana kahden eri kerroksen vuonohjauksen yhteensovittaminen.

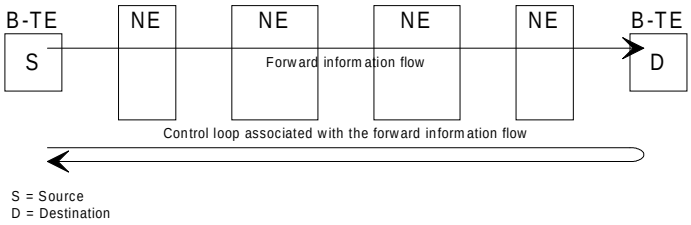
2. ABR-PALVELU

ABR-palvelun tavoitteena on hyödyntää korkeampi prioriteetiselta CBR- ja VBR-liikenteeltä yli jäänyttä verkon kapasiteettia. Palvelun avulla pyritään pitämään verkon käyttöaste korkeana ja solujen hävikkisuhde (CLR) pienenä sekä välttämään ruuhkatilanteet. Tavoitteet saavutetaan käyttämällä vuonohjausmenetelmää ATM-kerroksella.

ABR välttää verkon ruuhkatilanteet, jolloin voidaan minimoida puskuritarve ja soluhäviöt. ABR siirtää soluhäviöt ATM-verkon reunoille, jolloin suurten puskurien tarve siirtyy verkon keskeltä reunoille. Lähteitä ohjataan kytkentälaitteiden puskurin täyttöasteen tai kasvunopeuden perusteella ja pyritään siten välttämään solujen häviämiset ja ruuhkatilanteet.

2.1 ABR-PALVELUN VUONOHJAUS

ABR-palvelun vuonohjauksessa käytetään resurssin hallinta -soluja, RM-soluja. Lähde lähettää datasolujen välissä tietyin välein, esimerkiksi joka 32:s solu RM-solun. Perustapauksessa RM-solut päätyvät kohteelle, joka kääntää RM-solut takaisin lähteelle samaa kaksisuuntaista virtuaaliyhteyttä pitkin. Tätä on havainnollistettu kuvassa 2-1.



Kuva 2-1: ABR-yhteyden vuonohjaus [1]

Verkon kytkentälaitteet informoivat lähteitä RM-solujen avulla verkon kuormitustilanteesta. RM-solu sisältää tiedon eksplisiittisestä lähtelle sallitusta lähetysnopeudesta.

2.1.1 ABR-PARAMETRIT

Liitteessä 1 on esitetty ABR-yhteyttä muodostettaessa määriteltävät parametrit. Parametrejä on 15, joista vain osa on pakollisia parametreja. Tärkeimpiä vuonohjaukseen vaikuttavia parametreja ovat huippunopeus (PCR), miniminopeus (MCR), aloitusnopeus (ICR), nopeuden lisäyskerroin (RIF), nopeuden vähennyskerroin (RDF) ja kiinteä päästä päähän viive (FRTT).

2.1.2 KYTKENTÄLAITTEEN TOIMINTA

Eksplisiittistä nopeuden ilmaisua tukeva kytkentälaite asettaa RM-solun ER-kenttään sen hetkisen kyseiselle yhteydelle käytössä olevan kaistan. Kytkentälaite muuttaa ER-kentässä jo olevan nopeuden tilalle omansa vain mikäli se pysty antamaan kyseiselle yhteydelle niin paljon kaistaa kuin ER-kenttä ilmoittaa. Lisäksi kytkentälaite asettaa tarvittaessa RM-solun CI- tai NI-bitin.

Kytkentälaitteen tulee toteuttaa ainakin yksi seuraavista ruuhkanhallintatoiminteista jonotuspisteissä. [AF/TMS]

1) Kytkentälaite voi asettaa ruuhkatilanteessa EFCI-bitin datasolun otsikossa (engl. EFCI Marking).
2) Kytkentälaite voi asettaa CI=1 ja/tai NI=1 FRM- ja/tai BRM-soluihin ruuhkatilanteessa. (engl. Relative Rate Marking).
3) Kytkentälaite voi vähentää lähteen nopeutta pienentämällä ER-kentän arvoa FRM- ja/tai BRM-soluissa (engl. Explicit Rate Marking).

4) Kytkentälaite voi segmentoida ABR-liikenteen ohjaussilmukan käyttämällä virtuaalista lähdettä ja kohdetta (engl. VS/VD Marking).

2.1.3 LÄHTEEN TOIMINTA

Lähde ylläpitää sallittua lähetysnopeutta (ACR), vastaanottamiensa RM-solujen informaation perusteella. Yhteyden alussa lähde lähettää ensimmäisenä soluna RM-solun. Lähetysnopeutena käytetään aloitusnopeutta (ICR) siihen saakka kunnes lähde on vastaanottanut ensimmäisen RM-solun verkosta. Lähteen sallittu nopeus (ACR) määräytyy taulukon 2-1 mukaisesti. ER on lähteen vastaanottaman RM-solun ER-kentän arvo ja ACR edellisen RM-solun vastaanoton jälkeen laskettu ACR-parametrin arvo. Lähteen tulee lähettää FRM-soluja siten, että vähintään joka Nrm:s solu on RM-solu. Nrm-parametrin valinta vaikuttaa lähteen ohjauksen nopeuteen. Trm-parametri rajoittaa kahden peräkkäisen FRM-solun välistä maksimiaikaa, jolloin hitaidenkaan lähteiden ohjaukseen ei tule liikaa viivettä. TBE-parametri kertoo verkon tilapäisten puskureiden määrän eli solujen määrän, jotka lähde voi lähettää verkkoon ennen kuin se on ottanut vastaan ensimmäisen RM-solun verkosta.

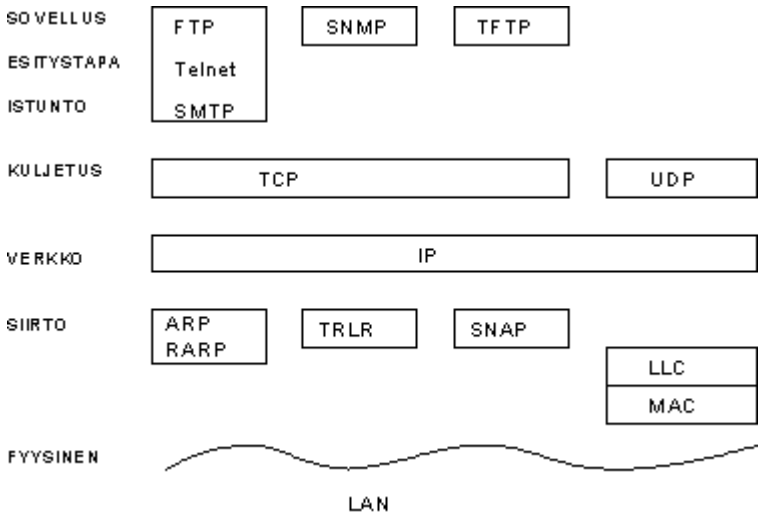
NI	CI	ACR
0	0	ACR= min(ER, (ACR+RIF*PCR), PCR)
0	1	ACR= min(ER, (ACR-ACR*RDF))
1	0	ACR= min(ER, ACR)
1	1	ACR= min(ER, (ACR-ACR*RDF))

Taulukko 2-1:Lähteen ACR-parametrin arvo [1]

3. TCP-PROTOKOLLA

Kuljetuskerroksen yhteyskäytännöt toimivat sovellusten välillä. TCP on yhteydellinen yhteyskäytäntö, joka takaa tiedon virheettömän välityksen ja huolehtii siitä, että tieto menee varmasti perille. TCP huolehtii myös datan järjestyksen säilymisestä ja yhteyden päättämisestä. Se paketoi sovellusten tuottaman datan segmentteihin ja antaa segmentit sitten IP-yhteyskäytännön siirrettäväksi. TCP tarjoaa eräälaisen kaksisuuntaisen virtuaalisen putken, jonka avulla tietokoneet siirtävät tietoa toisilleen.

Koska TCP on yhteydellinen yhteyskäytäntö, tiedonsiirtoa edeltää aina yhteyden muodostaminen. TCP-yhteyden (istunto) määrittelevät yleensä molempien koneiden IP-osoitteet ja porttiosoitteet yhdessä, sillä yhteys muodostetaan kahden sovelluksen välille. Kahden koneen välillä voi olla useita TCP-yhteyksiä, jotka käyttävät samaa IP-yhteyttä (kahden IP-osoitteen välillä). TCP-yhteys muodostuu, kun toinen sovellus lähettää TCP-kehyksen, jossa se ilmoittaa oman kehystahdistuksensa aloitusnumeron. Vastaaottavan osapuolen sovellus lähettää kehyksen, jossa se kuittaa vastapuolen kehystahdistuksen ja ilmoittaa omansa. Mikäli ensin lähettänyt sovellus hyväksyy kehystahdistuksen ja lähettää siitä kuittauksen, on yhteys muodostunut ja ollaan siirrytty datasiirtotilaan. /2/



Kuva 3-1 Yleisimmät internet-yhteyskäytännöt

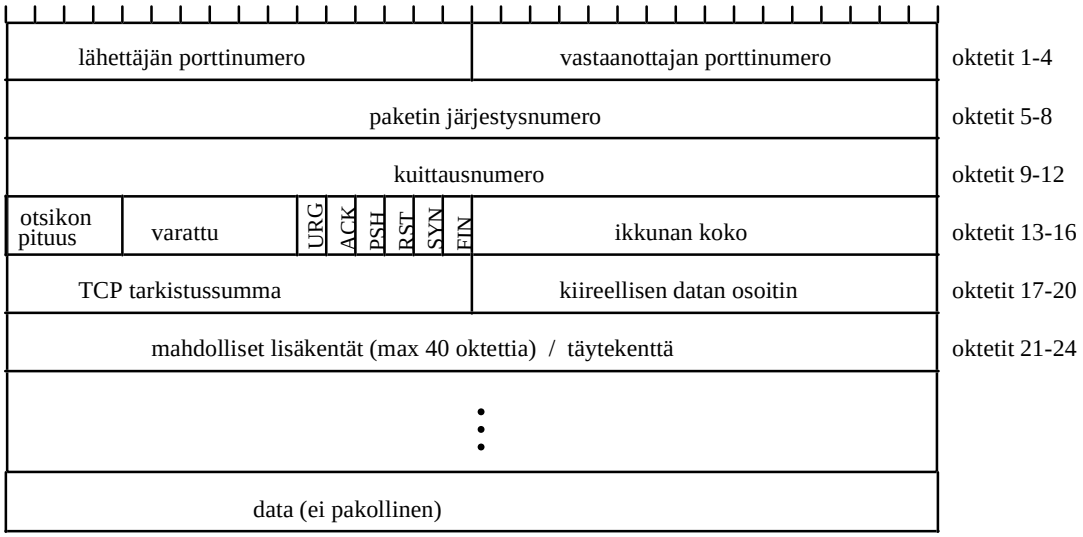
Tiedonsiirrossa vastaanottaja kuittaa saamansa paketit, ja ilman kuittausta jäänyt paketti lähetetään tietyn ajan kuluttua automaattiseti uudelleen. Virheiden ilmaisu perustuu tarkistussumman käyttöön, joka lasketaan sekä otsikolle että hyötykuormalle. Virheellisen paketin vastaanottanut sovellus voi pyytää vastapuolelta uudelleenlähetystä. Tiedonsiirrossa käytetty pakettikoko määräytyy TCP-kerroksella, jossa se pyritään optimoimaan IP-yhteyskäytäntöä varten. Tietovirran ohjausta varten voidaan määritellä haluttu paketin maksikoko, jotta voitaisiin optimoida puskureiden käyttöä.

IP ei ole varmennettu tiedonsiirtoprotokolla. IP vain hylkää paketit, jotka ovat olleet liian kauan verkossa (outdated) tai ovat ylittäneet sallitut siirto välit (hops) verkossa. Tämä näkyy TCP-kerroksella pakettien katoamisina. Koska IP tarjoaa yhteydettömän yhteyden, TCP:n tehtäväksi jää varmennus,

vuon ohjaus, järjestyksen säilyttäminen, yhteyden perustaminen ja purkaminen. Useita TCP:n toiminteita (vuon ohjaus, varmennnus, järjestyksen säilyttäminen) voidaan käsitellä sovellusohjelmien avulla.

3.1 TCP-KEHYS

Kuvassa 3-2 on esitetty TCP-protokollan kehys.



Kuva 3-2 TCP-kehys

Lähettäjän porttinumero (16 bits)

Identifioi lähettävän ylemmän kerroksen sovelluksen käyttämän TCP-kerroksen portinumeron.

Vastaanottajan porttinumero (16 bits)

Identifioi vastaanottavan ylemmän kerroksen sovelluksen, jolle paketti on menossa. Eli numero joka yksilöi sovellusprosessin joka ottaa vastaan TCP-yhteyden liikenteen.

Paketin järjestynumero (32 bits)

Kenttä sisältää ensimmäisen käyttäjän datassa olevan oktetin järjestysnumeron. Sen avulla voidaan varmistua myös pakettien järjestyksen säilymisestä.

Kuittausnumero (32 bits)

Vastaanottaja asettaa kuittausnumeroksi sen paketin ensimmäisen dataoktetin järjestysnumeron, jonka seuraavaksi odotetaan saapuvaksi lähettäjältä. Koska kuittausnumerokentän arvoksi asetetaan seuraavaksi odotun (ei vielä vastaanotetun) paketin järjestysnumeron, kuittaus tarkoittaa sitä että kaikki ennen tätä (kuittausnumerokentässä olevaa) järjestysnumeroa ennen olevat paketit kuitataan vastaanotetuiksi tällä sanomalla.

Otsikkokentän pituus (4 bits)

Otsikkikentän pituus -kenttä määrittää TCP-paketin otsikon pituuden (otsikon pituus ei ole vakio) 32-bitin kerrannaisena. Kentän avulla voidaan selvittää missä kohtaa otsikko loppuu ja käyttäjän data alkaa.

Varattu (6 bits)

Kenttä on varattu tulevaa käyttöä varten ja sen bitit asetetaan nolliksi.

Liput (6 bits)

TCP-istunnon aikana käytettävien palveluiden ja toiminteiden spesifiointiin tarkoitetut ohjausbitit. Jotkut liput määrittelevät kuinka muita TCP-kehyksen otsikkokenttiä tulee tulkita

UGR (1 bit)

Lippu osoittaa onko kiireellisen datan osoitin merkitsevä vai ei.

ACK (1 bit)

Lippu osoittaa onko kuittauskenttä merkitsevä vai ei.

PSH (1 bit)

Lippu osoittaa käytetäänkö PUSH toiminnetta vai ei.

RST (1 bit)

Lippu osoittaa että yhteys on resetoitu.

SYN (1 bit)

Lippu ilmaisee että järjestysnumerot on synkronoitu. Sitä käytetään yhteyden muodostusvaiheessa ilmaisemaan kättelyn tapahtumista.

FIN (1 bit)

Lippu ilmaisee että lähettäjällä ei ole enempää dataa lähetettävänä.

Ikkunan koko (16 bits)

Ikkunan koko -kenttää käytetään ilmaisemaan kuinka monta oktettia vastaanottaja on valmis ottamaan vastaan. Kentän arvo perustuu kuittausnumerokentän arvoon siten että ikkunan koko -kentän arvoksi asetetaan kuittausnumero + lähettäjälle tarjottu ikkunan koko.

TCP -tarkistussumma (16 bits)

Tarkistussumman avulla tarkastetaan ovatko TCP-paketin otsikko ja data kunnossa. Sen avulla määritetään onko lähettäjältä saapunut data tullut virheettömänä perille.

Kiireellisen datan osoitin (16 bits)

Kenttää käytetään ainoastaan silloin kun URG-lippu on asetettu. Kentän avulla osoitetaan dataosasta kohta jossa kiireellinen data sijaitsee mutta mitään erillisiä toimenpiteitä ei suoriteta.

Lisäkentät (max 40x8bits=320 bits)

- end to end option list
- no operation
- maximumm segment size

Täytekenttä (max 31 bit)

Täytekentän tehtävänä on varmistaa että TCP-paketin otsikko on 32 bitin kerrannainen

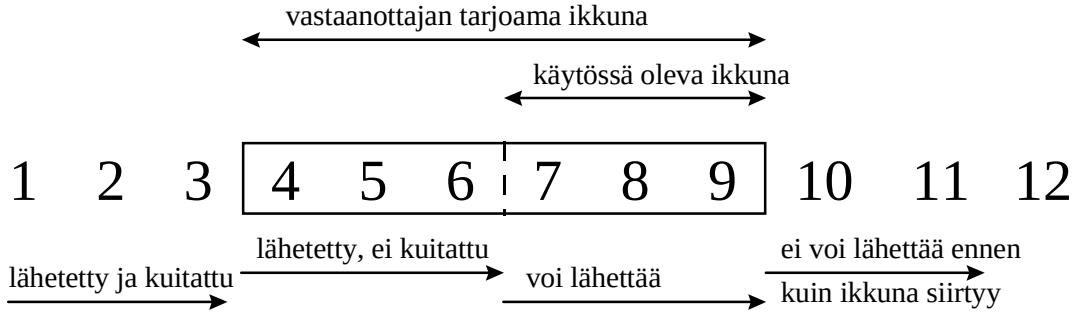
Käyttäjän data (ei pakollinen, vaihtuvan mittainen)

3.2 TCP-VUONOHJAUS

TCP:n vuonohjauksessa käytetään liukuvan ikkunoinnin menetelmää. Liukuvan ikkunoinnin käyttäminen mahdollistaa nopean datasiirron, koska lähettäjän ei tarvitse odottaa kuittausta jokaisen paketin jälkeen vaan se voi lähettää useita paketteja, ja vastanotettuaan kuitattauksen ensimmäisistä paketeista se voi lähettää jälleen lisää.

3.2.1 LIUKUVA LÄHETYSIKKUNA

Kuvassa 3-3 on numeroitu oktetit 1…12. Vastaanottajan tarjoama ikkuna rwnd (receiver window) pitää sisällään oktetit 4:stä 9:ään, tarkoitttaen että vastaanottaja on kuitannut kaikki oktetit oktettiin 3 saakka (oktetti 3 mukaan lukien) ja vastaanottajan tarjoaman ikkunan koko on 6 oktettia. Ikkunan koko on suhteessa kuitattuun järjestysnumeroon. Lähettäjä laskee käytössä olevan ikkunan, joka kertoo kuinka monta paljon dataa voidaan vielä lähettää. Ajan kuluessa vastaanottajan tarjoama ikkuna siirtyy oikealle sitä mukaa kun vastaanottaja kuittaa dataa. Ikkunan reunojen liikkuessa suhteessa toisiinsa ikkunan koko joko kasvaa tai pienenee. Mikäli lähettävä TCP vastaanottaa paketin joka siirtäisi ikkunan vasenta reunaa takaisin vasemmalle on kyseessä kahdentunut kuittaus ja se hylätään.



Kuva 3-3:TCP:n liukuva lähetysikkuna [3]

Yleensä vastaanottava prosessi kontrolloi tarjotun lähetysikkunan kokoa. Jokainen kuittauspaketti sisältää viimeisen järjestyksessä kuitatun oktetin numeron lisäksi tarjotun ikkunan koon, joka kuvaa sitä kuinka monta oktettia dataa vastaanottaja pystyy ottamaan vastaan. Tarjotun ikkunan koko riippuu vastaanottajan puskurikapasiteettista.

3.2.2 LÄHETTÄVÄN TCP:N VUONOHJAUS

Lähetysikkunan koko, *win* ilmoittaa kuinka paljon lähettävä TCP voi lähettää kuittaamatonta dataa, oktetteina. Sovellus voi lähettää TCP:lle dataa esimerkiksi oktetti kerrallaan. TCP kerää datan puskuriin segmenteiksi, joiden maksimipituus on MSS (yleensä 512 oktettia). Segmentin maksimipituus voidaan välittää TCP-kehyksen otsikon lisäkenttien avulla. Lähettävä TCP voi siis lähettää korkeintaan *win*/MSS kappaletta paketteja verkkoon. Lähettävän TCP:n lähetysikkunan koko lasketaan seuraavasti *win* = min(rwnd,cwnd*MSS). Rwnd on vastaanottavan TCP:n kuittauspaketeissa tarjoama ikkuna (oktetteina), jonka suuruus riippuu siitä, kuinka täynnä vastaanottajan sisääntulopuskuri on. Cwnd on arvio siitä kuinka monta pakettia verkkoon voidaan enintää lähettää ilman verkon ruuhkautumista.

Lähettävä TCP asettaa yhteyden alussa cwnd=1. Ensimmäisen paketin lähettämisen jälkeen lähettävä TCP jää odottamaan kuittausta, jossa se saisi tietää vastaanottajan tarjoaman ikkunan, rwnd. Yhden kuittauksen vastaanottaminen lisää lähettävän TCP:n cwnd-muuttujan arvoa yhdellä (cwnd = cwnd+1). Vastaanottajan kuittauspaketeissa tarjoaman ikkunan, rwnd koko on yleensä suuri, joten lähettävän TCP:n ikkuna kasvaa yleensä yhdellä paketilla kuittausta kohti. Tätä sanotaan slow start -vaiheeksi, jolloin ikkunan koko win kasvaa eksponentiaalisesti.

Usein saavutetaan tilanne, jossa verkko ruuhkautuu ja paketteja aletaan hylkäämään tai pakettien kuittaaminen kestää liian kauan. Lähettävällä TCP:llä on kaksi mahdollisuutta pakettien katoamisen havaitsemiseen ajastin laukeaa (timeout) tai vastaanotetaan useita (väh. kolme) kuittauksia samasta paketista. Tämä kertoo lähettävälle TCP:lle että cwnd on kasvanut liian suureksi. TCP:n on varmentavana protokollana lähetettävä kadonneet paketit uudelleen.

Lähettävä TCP ylläpitää kahta muuttujaa jokaista yhteyttä kohti cwnd ja slow start -vaiheen kynnysarvoa (slow start threshold), ssthresh.
1. Aluksi asetetaan cwnd=1 ja ssthresh=65535 (ikkunan maksimikoko = 2exp16)
2. TCP ei koskaan lähetä enempää kuin win=min(rwnd,cwnd*MSS).
3. Ruuhkan sattuessa (eli ajastin laukeaa tai vähintään kolme kuittausta samasta paketista) puolet sen hetkisestä ikkunan koosta (kuitenkin vähintään 2 pakettia) talletetaan ssthresh-muutujaan eli ssthresh=min(2,win/2).

Timeout

Paketin kuittaus pitää saapua RTO-ajan (Retransmission TimeOut) kuluessa paketin lähettämisestä. RTO-aika riippuu kiertoajasta (RTT). Liitteessä 2 on kerrottu lisää RTT:n ja RTO:n laskemisesta. Mikäli ruuhkan havaitseminen perustui ajastimen laukeamiseen (timeout) asetetaan cwnd=1 ja palataan takaisin slow start -vaiheeseen. Uudelleenlähetys aloitetaan järjestysnumeroltaan pienimmästä kuittaamattomasta paketista ja cwnd-muuttujan arvoa kasvatetaan yhdellä jokaista kuittausta kohti. Kun

cwnd:n arvo on kasvanut suuremmaksi kuin ssthresh (eli puolet alkuperäisestä ikkunan koosta), siirrytään congestion avoidance -vaiheeseen. Tällöin cwnd:n arvoa kasvatetaan siten että koko ikkunallisen kuittaaminen lisää ikkunan kokoa yhdellä eli cwnd = cwnd + 1/cwnd. Kun on lähetetty cwnd kappaletta paketteja ja saatu niihin kaikkiin kuittaus kasvatetaan cwnd:n arvoa yhdellä. Congestion avoidance lisää cwnd:n arvoa 1/cwnd:llä aina kun kuittaus on vastaanotettu, cwnd kasvaa siis yhdellä paketilla kiertoajassa riippumatta siitä kuinka monta kuittausta on vastaaotettu RTT:n kuluessa. Tällöin ikkunan koko kasvaa lineaarisesti.

Fast retransmit ja fast recovery

Jos ruuhkan havaitseminen perustui lähettävän TCP:n vastaanottamaan vähintään kolmeen samaa pakettia uudelleen pyytävään kuittaukseen, asetetaan cwnd = (ssthresh/MSS)+3 ja lähetetään kadonnut paketti uudelleen ilman että odotetaan uudelleenlähetysajastimen laukeamista. Tätä sanotaan fast retransmit-algoritmiksi. Uudelleenlähetyksen jälkeen siirrytään congestion avoidance-vaiheeseen eikä slow start-vaiheeseen niin kuin ajastimen laukeamisen jälkeen. Tätä kutsutaan fast recovery -algoritmiksi. Aina kun lähettävä TCP saa uuden samaa pakettia pyytävän kuittauksen lisätään cwnd:n arvoa yhdellä (paketilla) ja lähetetään uusi paketti (mikäli uusi cwnd sallii sen). Kun ensimmäiseen uudelleenlähetyksen jälkeen lähetettyyn pakettiin saadaan kuittaus asetetaan cwnd=sstresh/MSS ja siirrytään congestion avoidance -vaiheeseen. Tällöin ikkunan koko on puolet siitä mitä se oli paketin häviämishetkellä.

Kun TCP:n RTO-ajastin laukeaa ja suoritetaan uudelleenlähetys TCP:n ei tarvitse lähettää uudelleen samanlaista pakettia. Sen sijaan TCP voi paketoida uudelleen esimerkiksi suuremman paketin, suorituskyvyn parantamiseksi. Tämä on mahdollista koska TCP tunnistaa lähetetyn ja kuitatun datan tavunumeroista eikä paketin numerosta.

Push -toiminnetta käytetään kun sovellus haluaa varmistua siitä että kaikki sen alemmalle kerroksella eli TCP:lle antama data on lähetetty. Se liittyy TCP:n puskurien hallintaan. Sovellus lähettää komennon TCP:lle jossa PUSH-parametri lippu on asetettuna ykköseksi. Se vaatii TCP:tä lähettämään kaiken puskureissa olevan datan vastaanottavalle osapuolelle.

3.2.3 VASTAANOTTAVAN TCP:N TOIMINTA

Vastaanottavan TCP:n tehokkain vuonohjauskeino on vastaanottopuskurin koon rwnd asettaminen kuittauspaketteihin. Sen avulla vastaanottava TCP voi pienentää lähetysikkunan kokoa, jos se ei pysty ottamaan vastaan kaikkea lähetyttyä dataa.

Vastaanottava TCP käyttää tarkistussummaa tarkastaakseen datan. Mikäli data ja otsikko ovat kunnossa se lähettää positiivisen kuittauksen lähettäjälle. Mikäli data on vahingoittunut vastaanottava TCP hylkää datan ja ilmoittaa lähettäneelle TCP:lle ongelmista ilmoittamalla vahingoittuneen paketin järjestysnumeron.

TCP tarkastaa myös datan kahdentumisen ja vastaanottava TCP hylkää kahdentuneet paketit. Samoja paketteja voi esiintyä useampia mikäli vastaanottaja ei ehdi kuitata pakettia vastaanotetuksi riittävän ajoissa ennen kuin lähettäjä ehtii lähettää paketin jo uudelleen.

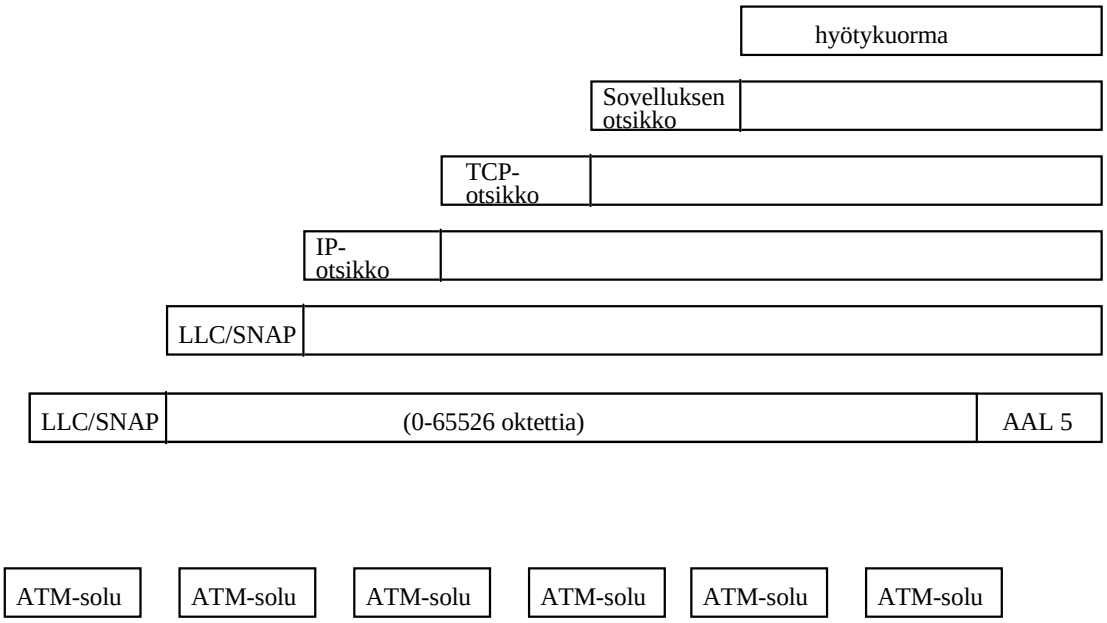
Sen lisäksi että järjestynumerointia käytetään kuittauksissa, järjestysnumerointia käytetään myös segmenttien uudelleen järjestämiseksi, mikäli ne saapuvat vastaanottajalle väärässä järjestyksessä. TCP käyttää mukaanlukevaa (inclusive) kuittaus menetelmää. Kuittausnumero kuittaa kaikki oktetit kuittausnumeroon saakka mukaanlukien kuittausnumeron vähennettynä yhdellä. Tämän avulla saavutetaan yksinkertainen ja tehokas kuittausmenetelmä. TCP voi kuitata vain suurimman järjestyksessä vastaan otetun paketin mikäli suurimman järjestysnumeron ja jatkuvan järjestysnumero jonon väliltä puuttuu paketteja ei niitä voida kuitata. Tämä voi aiheuttaa sen että vastaanotettuja mutta kuittaamattomia paketteja lähetetään uudelleen aivan turhaan.

4. TCP- JA ABR-VUONOHJAUS YHDESSÄ

4.1 DATAN PAKETOINTI

Sovelluksen data pakataan TCP-pakettiin, joka siirretään IP:lle kuljetettavaksi. IP-paketiin lisätään LLC/SNAP-otsikko. AAL5-kerroksella LLC/SNAP-otsikon lisäksi tulee AAL5-traileri. AAL5-kerroksen paketit pilkotaan 48 oktettia hyötykuormaa sisältäviksi ATM-soluksi.

TCP paketin kapsulointi ATM:n päälle: 20 tavun TCP otsikko , 20 tavun IP otsikko, 8 tavua RFC1577 LLC/SNAP enkapsuloinninsta ja 8 tavua AAL5 informaatiota yhteensä 56 tavua. Koska TCP:n paketin maksimikoko MSS on 512 tavua tulee yhden TCP-paketin kuorman siirrossa ATM.n yli siirrettäväksi 568 tavua hyötykuormaa. 568 tavun kuorman siirtäminen ATM:n yli vaatii 12 ATM-solua (48 tavua dataa kussakin).



Kuva 4-1 Otsikkokenttien lisäys

4.2 TCP- JA ABR-VUONOHJAUKSEN YHTEENSOPIVUUS

TCP:n datavuota kontrolloidaan TCP-vuonohjauksen avulla ennen sen saapumista ATM-kerrokselle. TCP:n lähetysikkunan koko cwnd kaksinkertaistuu alussa (kun dataa aletaan lähettämään) aina kiertoajan välein (slow start-vaihe). Tällöin liikenne näyttää ATM-kerroksella purskeiselta. Vasta kun ikkunan koko on suurempi kuin RTT, datavirta näyttää jatkuvalta. Kun TCP:n lähetysikkuna kasvaa niin suureksi että se vastaa suurempaa nopeutta kuin verkon lähteelle sallima nopeus ACR niin lähde muuttuu ikkunarajoitetusta nopeusrajoitetuksi. Tästä huolimatta TCP:n lähetysikkunan koko voi kasvaa edelleen, kunnes se saavuttaa ikkunan maksimikoon. Tällöin TCP ylikuormittaa ABR lähdettä ja päätelaitteeseen muodostuu ruuhkia. Mikäli päätelaitteessa ei ole riittävästi puskurikapasiteettia voidaan menettää soluja jolloin suorituskyky kärsii.

Binäärisillä ABR-vuonohjausmenetelmillä suorituskyky riippuu paljon RIF- ja RDF-parametrien arvoista. Ongelmana epäreiluus sillä lyhyiden yhteyksien RTO-parametri on pienempi kuin pitkillä yhteyksillä, jolloin ajastin laukeaa helpommin. Yleensä binääriset menetelmät ovat liian hitaita jotta voitaisiin ehkäistä ruuhka ATM-kerroksella ja vaikka ruuhka saataisiin ehkäistyä ATM-kerroksella päätelaitteiden puskureiden ylivuodot ovat melko todennäköisiä.

Tärkeimpiä tekijöitä jotka vaikuttavat yleisesti TCP:n toimintaan ABR:n päällä ovat kytkinalgoritmit ja nopeusohjaamattomien linkkiyhteyksien olemassaolo ja TCP:n toteutuksen yksityiskohdat: TCP:n purskeisuuden puskurointi, protokollakerrosten istuntojen välinen eristys ja kerrosten välinen vuonohjaus. Lisäksi verkkokonfiguraatio vaikutaa TCP:n toimintaan ABR:n päällä.

Fast Retransmit- ja Fast Recovery- toiminnot parantavat TCP:n suorituskykyä mikäli yksi TCP paketti kadotetaan. Suurinopeuksisilla linkeillä (kuten yleensä ATM-verkossa) verkon ruuhkat aiheuttavat useiden pakettien hukkumisia , jolloin joudutaan aloitamaan slow-start vaiheesta. Maksimi läpäisy saavutetaan kun TCP-lähteet ovat ABR:n nopeusrajoittamia.Läpäisyn huonontuminen johtuu usein liian suurista aika-askelista. TCP:n kellon asettaminen ja kuittausten niputtaminen eli ei kuitata jokaista pakettia erikseen vaan useampia kerralla, voivat aiheuttaa TCP:n ajastimien laukeamisen ja huonon suorituskyvyn.

4.2.1 ABR VAI UBR TCP:N KANSSA

Perustavoitteena TCP:lle sekä UBR:n että ABR:n päällä on se että soluja katoaa mahdollisimman vähän ja että puskurikapasiteettia on riittävästi jotta voitaisiin saavuttaa mahdollisimman hyvä TCP:n suorituskyky. Pyrittäessä solujen häviöttömyyteen ABR on skaalautuva virtuaaliyhteyksien lukumäärän suhteen eli tarvittava puskurikapasiteetti ei riipu yhteyksien lukumäärästä. Käytettäessä ABR:ää kytkentälaitteen pienin tarvittava puskurikoko riippuu yhteyksien kiertoajasta mutta ei yhteyksien määrästä. UBR:ssä tarvittava puskurikapasiteetti kasvaa yhteyksien määrän lisääntyessä, koska tarvittava puskurikapasiteetti riippuu TCP:n ikkunoiden maksimikokojen summasta.

UBR:ssä puskurikokovaatimus verkon kytkentälaitteille on yhteyksien lukumäärän ja kiertoajan funktio. UBR:ssä soluhäviöt johtavat usein kaistan epäreiluun jakamiseen, reiluutta voidaan parantaa kunnollisella puskurin jakamismenetelmällä, hylkäys politiikalla (esim. EPD) ja käyttämällä yhteyskohtaista jonotusta. /4 /

ABR:ssä tarvittava puskurin koko on suhteessa maksimi kiertoaikaan. Tarvittavan puskurin koko ja yleensäkin ABR:n suorituskyky riippuvat käytetystä ABR-liikenteen vuonohjausalgoritmista (esim. konvergointinopeudesta). Esimerkiksi ERICA-algoritmissa 4*RTT:n kokoisen puskurin luvataan takaavan häviöttämän toiminnan. /5/

Verkossa jossa ei ole päästä päähän ATM-yhteys (esimerkiksi kaksi LAN ethernetiä liitetty toisiinsa ATM-verkon avulla) ABR vuonohjaus saattaa siirtää ruuhkan verkon reunoille. Tällöin ATM-verkko voi olla ruuhkaton ABR-vuonohjauksen ansiosta mutta käyttäjän kokema palvelun laatu (esim. tiedoston siirtoaika) voi olla huonokin LAN-yhteyden vuoksi. Välitettäessä TCP-liikennettä UBR:n avulla ruuhka ATM-verkossa aiheuttaa solujen hylkäämistä, joka puolestaa aiheuttaa TCP:n lähetysikkunan (cwnd) pienetymisen. ABR-vuonohjaus pakottaa ATM-verkkoon lähettävän reitittimen (tai päätelaitteen) pienentämään lähetysnopeuttaan. Vasta reitittimen puskurin alettua täyttyä, alkaa TCP:n lähetysnopeus pienentyä. Tästä voi aiheutua pidempi ohjaussilmukan säätöviive kuin UBR:n tapauksessa. UBR edellyttää enemmän puskurikapasiteettia itse ATM-verkossa ja ABR puolestaan verkon reunoilla. /6/

ABR ohjaa lähteitä kytkentälaitteiden puskurin täyttöasteen tai kasvunopeuden perusteella ja pyrkii siten välttämään solujen häviämiset ja ruuhkatilanteet. Mikäli käytetään TCP:tä kuljetuskerroksella sovelluksen lähettämät TCP-kehykset täytyy puskuroida päätelaitteessa, mikä voi johtaa päätelaitteiden puskureiden ylivuotoon TCP:n lähetysikkunan kasvaessa liian suureksi. Yleisesti ABR:n ja UBR:n paremmuus riippuu konfiguraatiosta, ja pääteleitteiden ja kytkimien puskurikapasiteetista. Mikäli TCP-prosesseja on samassa koneessa, ruuhka kohdistuu päätelaiteeseen. Tällöin joissakin simuloinneissa UBR antaa paremman läpäisyn kuin ABR. Varsinkin isoissa verkoissa (WAN) UBR voi toimia paremmin kuin ABR. Tämä johtuu siitä että ABR:n rajoittamien lähteiden RTO-parametrit laukeavat

helpommin koska lähetysnopeus on ACR-parametrin eikä ikkunan rajoittama. Todellinen lähetysnopeus on siis pienempi kuin TCP olettaa jolloin myös uudelleenlähetysajastin laukeaa oletettua helpommin.

4.2.2 TAUSTALIIKENNE, ABR JA TCP

Käytettäessä ABR:ää verkossa, jossa ei ole taustaliikennettä eli VBR-liikennettä, TCP lähteet voivat kadottaa muutamia paketteja alussa, mutta tasapainotila saavutetaan nopeasti. Tässä tilassa verkkoon saapuvaa kuormitusta rajoittavat maksimi lähetysikkunan koko ja verkon sallima lähetysnopeus (ACR). Verkon puskurijonojen pituudet pysyvät pieninä, mutta lähteiden puskurijonot ovat pitkiä ja riippuvat lähetysikkunan maksimikoosta ja polun pituudesta. Lähetysikkunan koon suurentaminen kasvattaa lähteen verkkokortin puskurijonon pituutta.

Taustaliikenteellä pieni TBE-parametrin arvo toimii paremmin kuin suuri TBE-parametrin arvo. TBE:n arvo tulee asettaa myös suhteessa kiertoaikaan. TCP-paketin katoamisesta aiheutuvan vahingon suuruus riippuu RTO-ajan pituudesta, joka taas riippuu kääntäen kiertoajasta. TCP-implementaatiot mittaavat kiertoajan vain joko 100ms tai 500ms välein. (n*100 tai n*500). Pienemmät kuin yhden askeleen viiveet lasketaan yhdeksi askeleeksi eli n*100ms tai n*500ms. Yleensä mitä suurempi aika-askel sen huonompi suorituskyky. /7/

5. YHTEENVETO

TCP:n toteutuksen yksityiskohdat vaikuttavat suorituskykyyn kun käytetään TCP:tä ABR:n päällä. TCP:n kellojakson pienentäminen parantaa toimintaa. TCP:n kuittausten niputtaminen ja pitkät päätelaitteiden puskurijonot hidastavat vuonohjausta, jolloin suorituskyky huonontuu. Lisäksi lähteen puskurikapasiteetti on ratkaisevassa asemassa suorituskyvyn kannalta.

ABR-vuonohjausalgoritmien nopeus on ratkaisevassa roolissa käyttettäessä TCP:tä ABR:n päällä. Binääriset ABR-vuonohjausmenetelmät eivät toimi kovin luotettavasti TCP:n kanssa, koska toiminta riippuu liiaksi parametrien valinnasta. ER-algoritmit ovat parempia, koska ne konvergoivat yleensä nopeasti.

UBR:n ja ABR:n paremmuus TCP:n kanssa riippuu paljolti siitä millaisessa ympäristössä liikennettä välitetään. UBR on yksinkertainen ja sen avulla voidaan saavuttaa melko hyvä suorituskyky mikäli puskurit on mitoitettu oikein, käytetään sopivia solujen pudottamismenetelmiä ja yhteyskohtaista jonotusta. ABR:n toiminnan tehokkuus TCP-liikenteen välityksessä riippuu valitusta algoritmista, puskurikapasiteetista ja verkon konfiguraatiosta. Esimerkiksi ABR:ää tukemattomat kytkentälaitteet voivat huonontaa suorituskykyä ratkaisevasti

LÄHTEET

/1/ ATM Forum Traffic Management Specification 4.0. 1996. 107s

/2/ U. Black. 1992. TCP/IP and Related Protocols, McGraw-Hill.372 s.

/3/ W. R. Stevens. 1996. TCP/IP Illustrated, Volume 1, The Protocols. Addison-Wesley Publishing Company.

/4/ ATM Forum Contribution 96-518. 1996. Performance of TCP over UBR and buffer requirements. 7 s.

/5/ ATM Forum Contribution 96-517. 1996. Buffer requirements for TCP over ABR. 6 s.

/6/ ATM Forum Contribution 96-1406. 1996. End-to-End Traffic Management in IP/ATM Internetworks. 4 s.

/7/ ATM Forum Contribution 96-177. 1996. TBE and TCP/IP traffic. 5 s.

LIITE 1

Lyhenne	Parametrin kuvaus	Yksiköt ja vaihteluväli	Oletusarvo	Neuvottelu
PCR	Huippusolunopeus (Peak Cell Rate) on nopeus, jota lähde ei saa koskaan ylittää.	Solua/sekunti	-	pakollinen
MCR	Minimisolunopeus (Minimum Cell Rate) on nopeus, jolla lähde saa vähintään lähettää kaikissa tilanteissa.	solua/sekunti	0	pakollinen
ICR	Alkulähetysnopeus (Initial Cell Rate) on nopeus, jolla aloitetaan liikenteen lähetys ja jatketaan lähetystä mikäli välillä on ollut jaksoja jolloin ei ole lähetty mitään.	Solua/sekunti	PCR	pakollinen
RIF	Nopeuden lisäys kerroin (Rate Increase Factor) kontrolloi nopeuden lisäystä, joka voidaan tehdä yhtä vastaanotettua BRM-solua kohti.	kahden potenssi, vaihtelee: 1/32768 - 1, yksikkö: solua/sekunti	1	pakollinen
Nrm	Solujen maksimimäärä, jonka lähde voi lähettää yhtä lähettämäänsä FRM-solua kohti.	kahden potenssi, vaihtelee 2 ja 256 välillä	32	valinnainen
Mrm	Mrm kontrolloi kaistan allokointia FRM-solujen, BRM-solujen ja datasolujen välillä.	vakio Mrm=2	2	-
RDF	Nopeuden vähennyskerroin (Rate Decrease Factor) kontolloi solunopeuden vähentämistä.	kahden potenssi, vaihtelee 1/32768 ja 1 välillä	1/32768	pakollinen
ACR	Lähteen sallittu kaista (Allowed Cell Rate) on nopeus, jolla lähde saa kyseisellä hetkellä lähettää. ACR on lähteen ylläpitämä muuttuja.	solua/sekunti	-	-
CRM	Puuttuvien RM-solujen lukumäärä. CRM ilmaisee FRM-solujen lukumäärän, jotka lähde voi lähettää ilman että se on ottanut vastaan yhtään BRM-soluja	CRM on kokonaisluku, jonka suuruus on toteutusriippuvainen	-	-
ADTF	Lähteen sallitun kaistan (ACR) vähennysaika-kerroin on maksimi aika joka sallitaan kahden lähteen lähettämän RM-solun lähettämisen välillä. Mikäli RM-soluja lähetetään harvemmin kuin ADTF asetetaan ACR=ICR.	yksikkö: sekunteja vaihtelee: 0.01 -10.23 s 10ms portaat	0,5	valinnainen
Trm	Suurin aika joka sallitaan aktiivisen lähteen lähettämien FRM-solujen välillä. Trm ilmoittaa kuinka pitkä on korkeintaan lähteen lähettämien RM-solujen välinen aika.	100 kertaa kahden potenssi, vaihtelee $100*2^{-7}$ ja $100*2^{0}$ välillä, yks. millisekunti	100	valinnainen
FRTT	Edestakainen päästä päähän-viive, joka muodostuu kiinteiden ja etäisyyden aiheuttamien viiveiden summasta, joka kertyy lähteestä kohteeseen ja takaisin (Fixed Round-Trip Time)	yksikkö: mikrosekunti vaihtelee: 0 ja 16.7 s välillä	lasketaan yhteyden muodostuksessa	-
TBE	Tilapäisten puskureiden määrä eli solujen määrä, jonka lähde voi lähettää lähetystä aloitettaessa tai hiljaisen jakson jälkeen ennen kuin se on saanut takaisin ensimmäisen BRM-solun. (Transient Buffer Exposure)	yksikkö:soluja, vaihtelee 0 ja 16,777,215 välillä	16,777,215	pakollinen
CDF	Kontrolloi ACR vähentämistä joka liittyy puuttuvien RM-solujen määrään (CRM). (Cutoff Decrease Factor)	0 tai kahden potenssi, vaihtelee 1/64 ja 1 välillä	1/16	valinnainen
TCR	Rajoittaa nopeuden jolla lähde voi lähettää 'out-of-rate' FRM-soluja. (Tagged Cell Rate)	vakio TRC=10 solua/sekunti	10	-

ABR -parametrit [1]

LIITE2

Kiertoajan mittaaminen

TCP:n ajastimet ja uudelleenlähetykset mitataan suhteessa kiertoaikaan (RTT), joka muuttuu ajan mittaan. Se voi muuttua koska pakettien reitti voi muuttua ja koska verkon liikennekuormitus vaihtelee. Uudelleenlähetysajastimet ovat kompleksisia koska:

a) viive, joka kuluu vastaanottavalta asemalta tulevan kuittauksen saapumiseen lähettäjälle riippuu verkon kuormitustilanteesta

b) lähettäjän lähettämät paketit voivat hukkua verkkoon

c) vastaanottajan lähettämät kuittaukset voivat hävitä verkkoon

Ajastimia ei voi asettaa kiinteiksi, koska viiveet vaihtelevat joten on käytettävä arviointiin perustuvia menetelmiä./2/

Aluksi TCP:n on mitattava RTT tietyn lähetetyn pakettin ja sen kuittaamisen välillä. Tässä voi tulla epätarkkuutta sillä vastaanottaja ei välttämättä kuittaa kutakin vastaanottamaansa pakettia kerrallaan vaan yleensä useampia paketteja kuitataan yhdessä kuittauspaketissa.

Alkuperainen TCP:n määrittely käyttää tasoitettua RTT estimaattoria (R). M on mitattu RTT:n arvo.

$R=a*R+(1-a)M$

jossa a suositellaan asetettavaksi arvoon 0,9. Tätä eksponentiaalista liukuvaa keskiarvoa päivitetään aina kun uusi mittaus (M) tehdään. 90 % uudesta arvosta muodostuu vanhasta arvosta ja vain 10 % mitatusta arvosta. IETF suosittaa asetettavaksi RTO=Rb, jossa b on viiveen varianssitekijä (b:n suositusarvo on 2).

Tämän menetelmän ongelmana on se että se ei pysy mukana suurissa RTT-arvon heilahteluissa, jolloin seurauksena voi olla tarpeettomia uudelleenlähetyksiä jo valmiiksi ruuhkaiseen verkkoon./3/

Ongelman ehkäisemiseksi voidaan pitää yllä tietoa RTT-mittausten varianssista tasoitetun RTT estimaattorin lisäksi. Laskemalla RTO keskiarvon ja varianssin perusteella saadaan huomattavasti parempi vaste suuriin RTT-ajan vaihteluihin. Keskimääräinen poikkeama on hyvä aproksimaatio stadardista poikkeamasta (standard deviation).

$Err= M - A$ (Err on mitatun ja tasoitetun RTT-ajan keskiarvon erotus)

$A = A + g*Err$

$D = D + h*(|Err| - D)$

$RTO = A +4*D$

jossa A on tasoitettu RTT (keskiarvon estimaattori) ja D on tasoitettu keskimääräinen poikkeama. Err on ero mitatun RTT:n arvon ja sen hetkisen RTT estimaattorin välillä. Sekä A:tä että D:tä käytetään lasketttaessa seuraavalle RTO:lle arvoa. Kerroin g asetetaan 1/8 eli 0,125. Kerroin h asetetaan 0,25:ksi. Annettaessa suurempi arvo h:lle aiheuttaa RTO:n kasvamisen nopeammin kun RTT muuttuu.

Ongelmia syntyy kun paketteja lähetetään uudelleen, jos paketin uudelleenlähetys aiheutuu uudelleenlähetysajastimen laukeamisesta RTO:n arvoa kasvatetaan RTO = RTO*y (yleensä y=2). /3/

Karnin algoritmissa määritellään että kun uudelleenlähetys ja uudelleenlähetysajastimen laukeaminen tapahtuu, ei RTT estimaattoria päivitetä. Lisäksi Karnin algoritmissa RTO:n arvoa kasvatetaan aina kun ajastin on lauennut. Karnin algoritmi toimii hyvin ellei RTT-ajan vaihtelut ole suuria. /2/