5 LIIKENTEEN HALLINNAN MITTAUKSIA

ATM-verkossa yhteyden muodostusvaiheessa luotava liikennesopimus sisältää määrittelyjä solunopeudesta, siirtoviiveestä ja mahdollisesta soluhukkatodennäköisyydestä. Nämä parametrit ovat käyttäjän antamia subjektiivisia kuvauksia halutusta yhteydestä. Näin ollen annetut parametrit eivät välttämättä vastaa tarjottua liikennettä. Verkon ja sen kytkentälaitteiden kannalta suurehko toleranssi arvoissa johtaa ylikuormitustilaan, jonka hallintaan on erilaisia toimintamalleja.

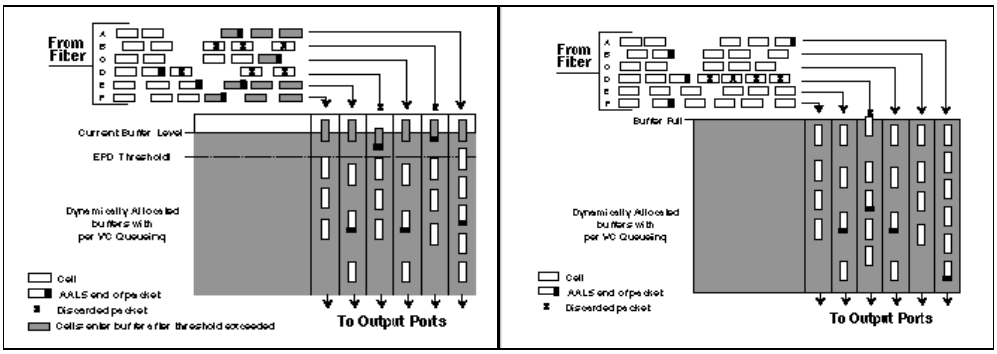
Koska päätelaitteiden kustannukset halutaan pitää mahdollisimman alhaisina, on ruuhkatilanteiden valvonta ja hallinta sijoitettu lähes täysin verkon kytkentälaitteisiin. Ainut poikkeus on ABR-päätelaitteet, jotka sisältävät kyvyn mukautua verkon estotilaan, tosin verkosta tulevilla ohjauskomennoilla.

5.1 YLIKUORMAN HALLINTA

Karkein tapa toimia on ylikuormittavan liikenteen karsiminen estoa kokevassa kytkentälaitteessa. Karsiminen perustuu tällöin solun otsikon CLP-bittiin ja on näin ollen hyvin karkeaa. Jotta karsiminen olisi olemassa oleville yhteyksille mahdollisimman näkymätöntä, tulisi solujen poisto tapahtua niiltä yhteyksiltä, joilla CLP on asetettu ‘ykköseksi’ ja/tai poistetut solut muodostavat korkeamman protokollakerroksen siirtokehyksiä (AAL-PDU). Ylempien protokollakerrosten huomioiminen on kannattavaa, sillä ATM-verkossa tullaan välittämään vielä pitkään perinteisiä kuljetusprotokollia, kuten TCP ja TP4. Perinteiset protokollat eivät kykene paketin osittaisiin uudelleen lähteyksiin ja siksi yhden solun hukkaaminen johtaakin koko protokollakehyksen (10-200 -solua) poistamiseen. [20]

Valikoivia liikenteen karsintamenetelmiä ovat EPD *(Early Packet Discard)* ja PPD *(Partial Packet Discard)*. Ne perustuvat AAL5 -kehyksen tunnistamiseen solun otsikon PTI-kentän avulla (AAL5 -kehyksen viimeinen solu merkitään muuttamalla PTI-kentän kolmas bitti ykköseksi). Näin pystytään toimimaan solutasolla vaikkakin toiminta perustuu ylempiin protokollakerroksiin. EPD on estoa ennakoiva menetelmä. Se hyödyntää tietoa jonon pituudesta. Mikäli jonon pituus ylittää aseteltavan rajan hylätään saapuva AAL5 -kehys. PPD on vastaavasti

reaktiivinen toimintamalli, jossa täyttynyt jono on aiheuttanut ylivuodon puskurissa ja näin ollen aikaansaanut AAL5 -kehyksen vikaantumisen. PPD poistaa jäljelle jääneet solut jonosta ja näin vapauttaa tilaa tuleville soluille.



Kuva 5-1 EPD- ja PPD-algoritmien toimintatapa [39]

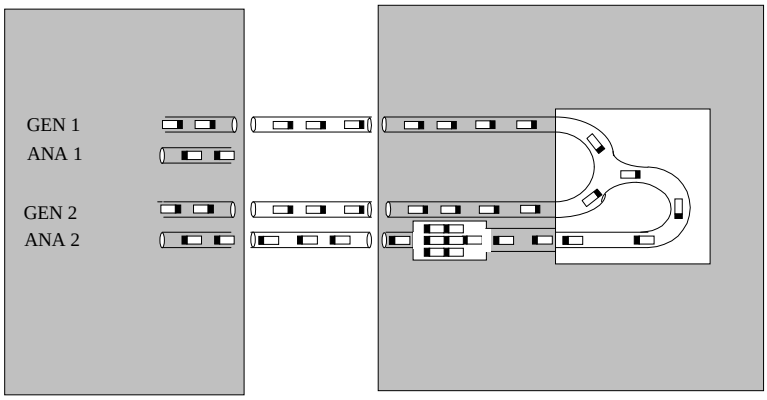
Äärimmäisenä liikenteen hallinnan keinona voidaan kokonaisia yhteyksiä purkaa ja näin helpottaa ylikuormitustilannetta.

5.1.1 Mittaus 1: Identtisten CBR-yhteyksien karsiminen

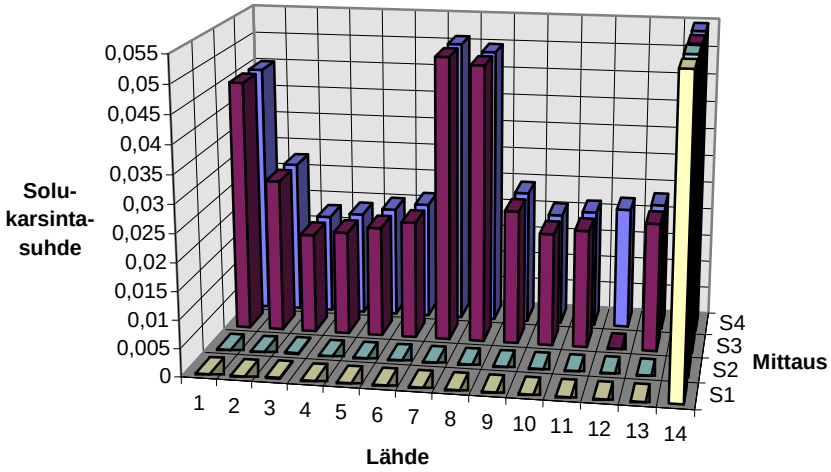
Yhteyksien vaatimaa liikennettä generoitiin kahdesta liitynnästä siten, että CBR-lähteiden ja yhden UBR-lähteen yhteiskuormitus ylitti 5%:lla linkin kapasiteetin. Generoinnissa käytettiin testisolua PRBS $1\text{-}2^{-9}$ ja prosessina periodista solugeneraatiota nopeudella 26000 solua/s

Taulukko 5-1 Yhteyksien liikenneparametrit

YHTEYS	PCR(0)	PCR(0+1)	SCR	BT	CDVT	CLP
1-7		26500			6	0
8-13		26500			6	0
14	UBR					1



Kuva 5-2 Mittauskytkentä solukarsinnan mittaukselle.



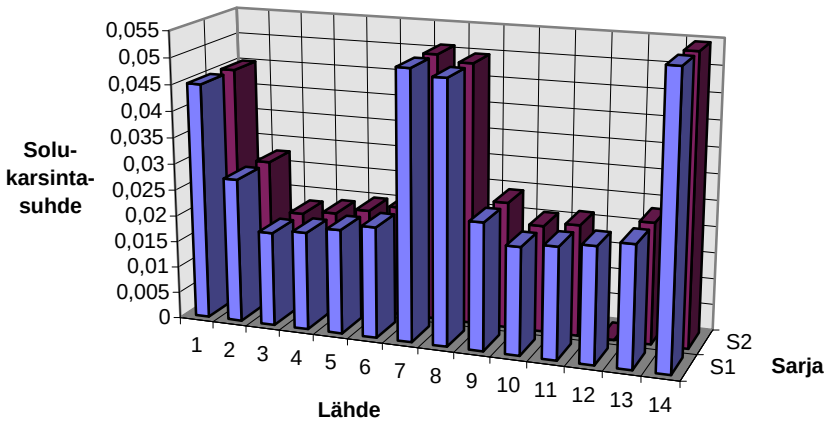
Kuva 5-3 CDR eri yhteyksillä

5.1.2 Mittaus 2: Identtisten VBR-yhteyksien karsiminen

Yhteyksien vaatimaa liikennettä generoitiin kahdesta liitynnästä siten, että lähteiden yhteiskuormitus ylitti 5%:lla linkin kapasiteetin. Generoinnissa käytettiin testisolua PRBS $1\text{-}2^{-9}$ ja prosessina Poisson-prosessin solugeneraatiota keskinopeudella nopeudella 26000 solua/s.

Taulukko 5-2 Yhteyksien liikenneparametrit

YHTEYS	PCR(0)	PCR(0+1)	SCR	BT	CDVT	CLP
1-13		353208	26000	100	5	0
14	UBR					1



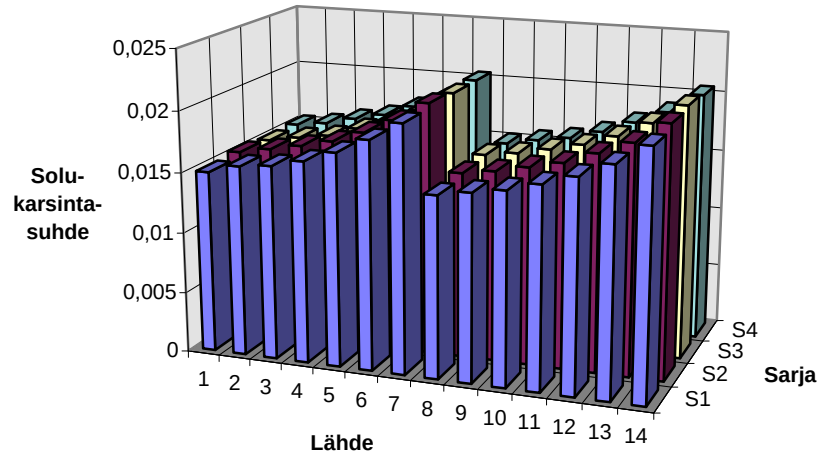
Kuva 5-4 CDR eri yhteyksillä

5.1.3 Mittaus 3: Identtisten UBR-yhteyksien karsiminen

Yhteyksien vaatimaa liikennettä generoitiin kahdesta liitynnästä siten, että lähteiden yhteiskuormitus ylitti 5%:lla linkin kapasiteetin. Generoinnissa käytettiin testisolua PRBS $1\text{-}2^{-9}$ ja prosessina periodista solugeneraatiota nopeudella 26000 solua/s.

Taulukko 5-3 Yhteyksien liikenneparametrit

YHTEYS	PCR(0)	PCR(0+1)	SCR	BT	CDVT	CLP
1-14	UBR					1



Kuva 5-5 CDR eri yhteyksillä

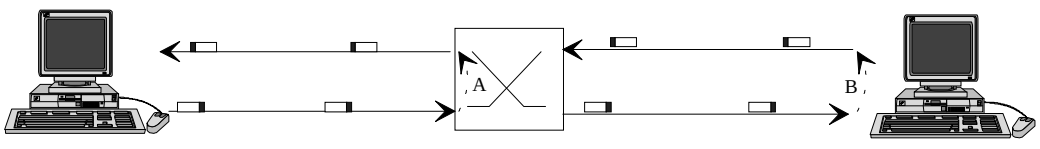
5.1.4 Yhteenveto tuloksista

Mittauksen 1 mittaussarjat S1 ja S2 sekä S3 ja S4 sisältävät huomattavia eroja tuloksissa. Mittaukset suoritettiin samana ajankohtana mutta mittauksessa vaihdettiin kytkimen liitäntäkorttia. Mittaussarjat 1 ja 2 suoritettiin FOREn liitäntäkorttiin NM-C-OC3c/STM1c-MM-ST-128KB-4PT, kun mittaukset 3 ja 4 perustuvat liitäntäkorttiin NM-A-OC3c/STM1c-MM-ST-4PT. Sarjan C liitäntäkortti hyödyntää puskurimuistin dynaamista jakamista sekä erilaisia painokertoimia (prioriteetteja) eri yhteystyypeille. Mittaustulosten perusteella voi päätellä, että sarja A:ssa ei ollut kyseisiä ominaisuuksia. C-sarjan korteissa on nähtävissä kytkimen deterministinen ratkaisu rajoittaa UBR-liikenteen palvelua ylikuormitustilanteessa. CBR-liikenteen soluhukkasuhde pysytteli koko mittauksen ajan alle 10^{-8}. Mittauksen kolme toteutus vastasi mittausta yksi. Siinä tulokset eivät juurikaan eroa toisistaan mutta tämä oli odotettavissa, sillä UBR-liikenne vastaa suoraa solukytkentää ilman likenteen hallinnan takuuta paluvelun laadulle. Mittauksessa kaksi käytettiin C-sarjan liitäntäkorttia, tulos sinällään on hieman yllättävä. Odotettavissa oli UBR-lähteen voimakas karsinta huippunopeus piikkien aikana mutta se, että kuvaaja muistuttaa CBR-liikenteen tapausta vanhoilla liitäntäkorteilla oli yllätys.

5.2 LIIKENTEEN OHJAUS

Liikenteen ohjausta voidaan suorittaa uuden ABR-lähdetyypin avulla. ABR hyödyntää verkon aktiivista ohjausta. Ohjaus perustuu verkon kytkentälaitteiden antamiin kuormitustietoihin ja niiden perusteella muodostettuihin hetkellisiin kapasiteettiarvoihin. Ohjaustieto välitetään resurssien hallintasoluilla läpiverkon, joko päästä-päähän (B) tai siirtoyhteyksittäin (A). Ohjaustiedon käsittelylle on olemassa kaksi vaihtoehtoa:

- Luottopohjainen
- Nopeuspohjainen



Kuva 5-6 Resurssien hallintasolun kulkeminen läpi verkon ABR-palvelussa

Luottopohjainen hallinta hyödyntää EFCI (Efficient Forward Congestion Identification) informaatiota, joka välitetään solun otsikon PTI-kentässä. EFCI-informaatio on CLP:n tavoin kaksi tasoinen: esto ja ei estoa. Koska näin voidaan välittää vain karkeita ohjauskomentoja, joudutaan nopeutta pienentämään resurssin hallintasolujen välillä portaittain. Tällaisesta liikenteen ohjauksesta on seurauksena solunopeuden jatkuva värähtely teoreettisen ihannearvon ympärillä.

Nopeuspohjainen ratkaisu edellyttää verkon laitteilta suurempaa älykkyyttä ja prosessoinnin tehokkuutta. Siinä resurssin hallintasoluihin lisätään solunopeus, joka sillä hetkellä kyetään välittämään verkon läpi. Näin ollen solun tullessa takaisin päätelaitteeseen, päätelaite tietää mihin nopeuteen, tai alle, sen on muutettava lähetysnopeutensa. [32]

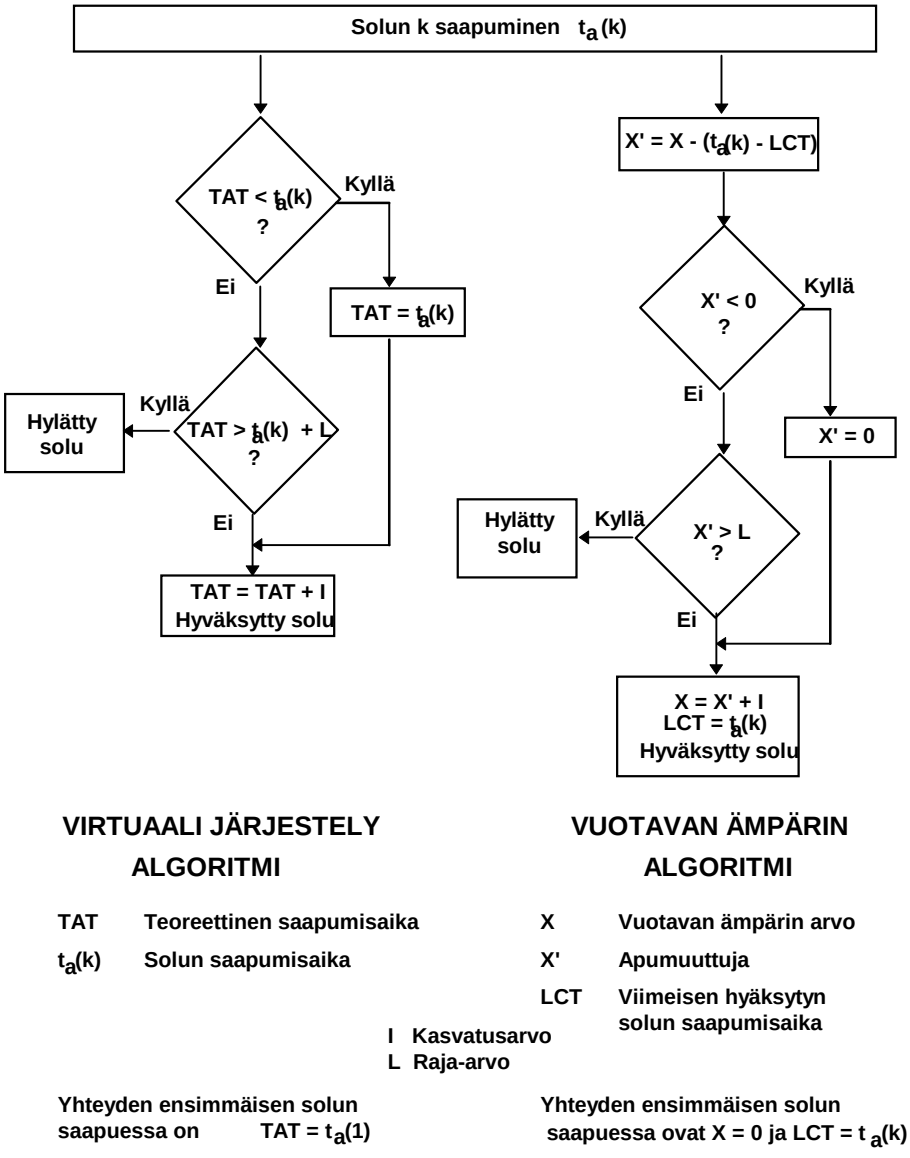
5.3 LIIKENNEPARAMETRIEN VALVONTA

Liikenneparametrien valvonnan (*Usage Parameter Control ja Network Parameter Control*) on tarkoitus estää yhteyksiä vaikuttamasta toisten yhteyksien kokemaan palvelun laatuun. Tämä tapahtuu valvomalla kutakin yhteyttä tai tilatun väylän tapauksessa kutakin väylää erikseen. Parametrivalvontaa suoritetaan tilaajan ja verkon liityntäpisteessä (UPC) sekä kahden verkon välissä (NPC). Parametrivalvonta suoritetaan liikennesopimuksen yhteysparametreille, joiden mukaisesti päätelaitteen tulisi lähettää soluja verkkoon. Tämä solujen lähetysprosessi ja sen liikennesopimuksen mukaisuus, jää kuitenkin täysin päätelaitteen ja viimekädessä käyttäjän vastuulle. Liikennesopimuksen rikkomisen tulisi aina johtaa korjaustoimenpiteeseen, joka voi olla solun hylkääminen tai sitten koko liikenteen muokkaaminen soluja viivästämällä. Parametrivalvontaa kannattaakin tarkastella vasteajan ja valvonnan oikeellisuuden suhteen. Suositusten mukaan valvonnan pitää olla äärettömän nopeaa, virheetöntä läpinäkyvää. [29]

Liikenneparametrien valvontaprosessi perustuu ITU-T:n suosituksissa I.356 ja I.371 esitettyyn GCRA-algoritmiin. Algoritmi määrittelee vastaanotettujen solujen tuloajan $t_a(k)$ perusteella onko saapunut solu liikennesopimuksen mukainen. Tuloaikaa verrataan teoreettiseen tuloaikaan (TAT), joka saapuvalla solulla tulisi olla. Koska yhteys voi olla purskeinen on tarkastelukin kaksi vaiheinen. Tuloaikaa tarkastellaan huippusolunopeuden (PCR) määrittelemän väliajan T_p,

keskimääräisen solunopeuden (SCR) määrittelemän väliajan T_s, pursketoleranssin (BT) määrittelemän ajan τ_s ja solun siirtoviiveen varianssin (CDVT) määrittelemän ajan τ avulla. CDVT on satunnaisviive, joka syntyy kytkentälaitteessa ja on siten yhteinen keski- ja huippunopeudelle.

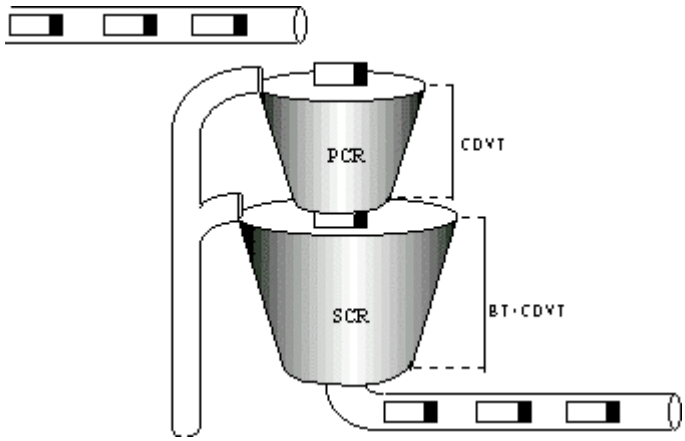
Ensimmäinen tarkastelu suoritetaan huippunopeudelle, jolle sallitaan tietty varianssi eli GCRA(T_p,τ). Toinen tarkastelu perustuu keskinopeuteen ja pursketoleranssiin eli GCRA(T_s,τ_s+τ).



Kuva 5-7 GCRA-algoritmin toteutustapoja [29]

GCRA-algoritmi on liukuvan ikkunan laskenta-algoritmi, jota voidaan ajatella vuotavana ämpärinä (oikean puoleinen toteutus). Vuotavia ämpäreitä on yleensä kaksi peräkkäistä. Ensimmäinen ämpäri valvoo huippunopeutta. Ämpärin koko on

soluviiveen varianssin määrittelemä ja sen tyhjenemisnopeus on huippunopeus. Toinen ämpäri valvoo keskinopeutta. Ämpärin koko määritellään pursketoleranssin ja soluviiveen varianssin summana. Ämpäri tyhjenee keskinopeuden määräämällä nopeudella.



Kuva 5-8 Dual Leaky Budget rakenne

Liikennesopimuksessa arvot määritellään soluina sekunnissa kun taas GCRA:n arvot on määritelty mikrosekunteina. Arvot täytyykin muuntaa oikeaan muotoon.

$$T_p = \frac{1}{PCR} \qquad (3)$$

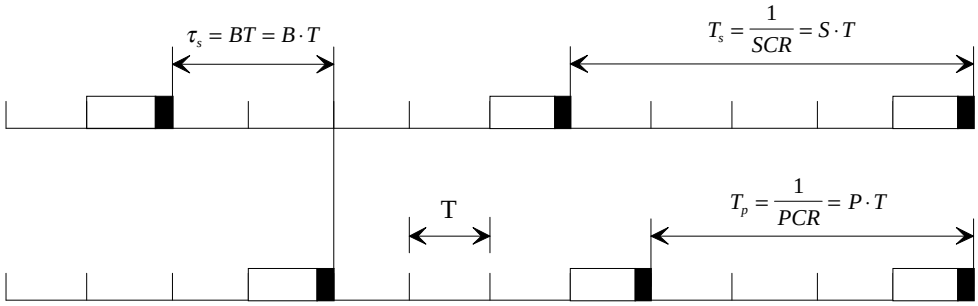
$$T_s = \frac{1}{SCR} \qquad (4)$$

$$\tau = CDVT \qquad (5)$$

$$\tau_s = BT \qquad (6)$$

$$MBS = \left\lfloor 1 + \frac{\tau_s}{T_s - T_p} \right\rfloor \qquad (7)$$

Pursketoleranssi BT esitetään usein huippusolunopeudella lähetettävän purskeen maksimikokona (MBS).



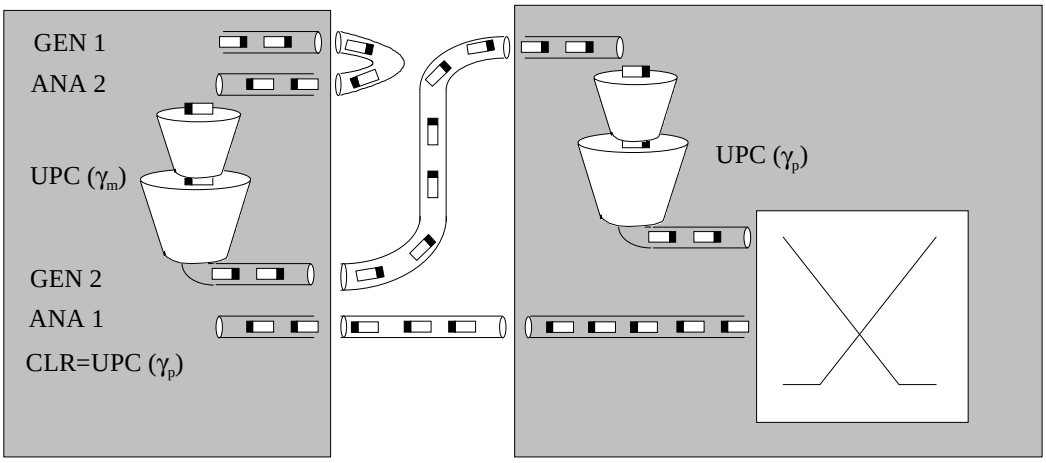
Kuva 5-9 Solunopeuksien merkitykset ja suhteet

Koska algoritmi perustuu solujen hylkäämiseen, mikäli ne rikkovat liikennesopimusta, eroaa algoritmin karsiman soluvirran parametrit sallituista arvoista. Tämä ero on aina tilaajalle epäedullinen, sillä algoritmi ei ota huomioon aikaisempien toimenpiteiden aiheuttamaa solunopeuden tippumista. Algoritmin perusajatus on tarkastella jokaista saapuvaa solua omana yksikkönä ja täysin riippumatta edellisistä toimenpiteistä. Mikäli halutaan varmistua, että UPC-prossessi ei aiheuta yhteyden vajaakäyttöä, tarvitaan joko solujen uudelleen järjestelyä tai tarpeeksi suurta arvoa CDVT:lle. CDVT, joka määritti ämpärin koon, kompensoi lievän nopeuden ylityksen ja solujen jaksollisen poistamisen siten, että poistettu solu ei aiheuta ämpärin tyhjenemistä ennen seuraavaa saapuvaa solua. Näin ajateltuna valvontaprosessin toimet eivät johda yhteyden vajaakäyttöön.

UPC-prosessin tehokkuutta ja tarkkuutta on tarkasteltu Euroopan Unionin rahoittamassa RACE *(Research and technology in Advanced Communications technologies in Europe)* -ohjelmassa. RACE-ohjelman hanke R2061 (EXPLOIT) rakensi laajamittaisen tutkimusympäristön, jossa suoritettiin myös liikenteen hallinnan (UPC ja CAC) mittauksia. [12]

5.3.1 Parametrivalvonta kanavatasolla

Kanavatasolla parametrivalvontaa suoritetaan erikseen jokaiselle virtuaalikanavalle. Virtuaalikanavan valvontaa tarkastellaan kytkennällä, jossa lähetettävä soluvirta muokataan mittalaitteessa täyttämään annettu yhteyssopimus. Tämän jälkeen soluvirta ohjataan kytkentälaitteeseen ja sieltä takaisin mittalaitteeseen. Mittalaitteessa vastaanotettua soluvirtaa tarkastellaan analysaattorin GCRA-algoritmilla.



Kuva 5-10 Parametrivalvonnan mittauskytkentä

Mittauksen avulla on etsitty riippuvuutta eri parametrien muutokselle ja sen aiheuttamalla parametrin valvontaprosessille. ATM Forumin suositukset määrittelevät UPC-prosessille hyvyysluvun, joka määritellään ideaalisen GCRA-algoritmin havaitsemien liikennesopimuksen rikkovien solujen lukumäärän ja mitattavan GCRA-algoritmin havaitsemien liikennesopimuksen rikkovien solujen lukumäärän erotuksen suhteena kaikkiin soluihin. [32]

$$F = \gamma_M - \gamma_p$$

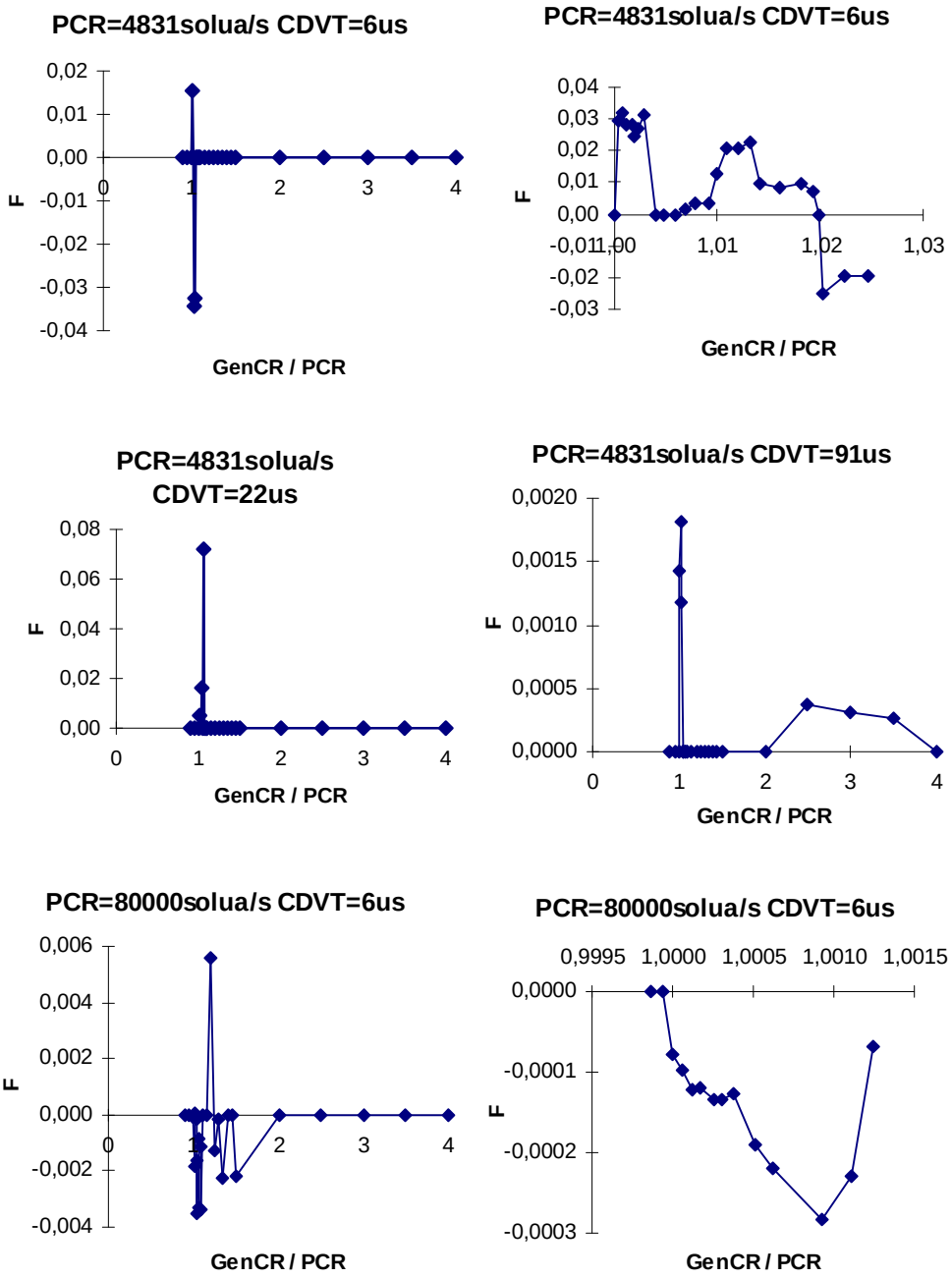
γ_M = Ideaalisen GCRA:n havaitsemien liikennesopimusta rikkovien solujen suhde lähetettyihin soluihin

γ_P = Mitattavan GCRA:n havaitsemien liikennesopimusta rikkovien solujen suhde lähetettyihin soluihin

Taulukko 5-4 Virtuaalikanavien liikenneparametrit

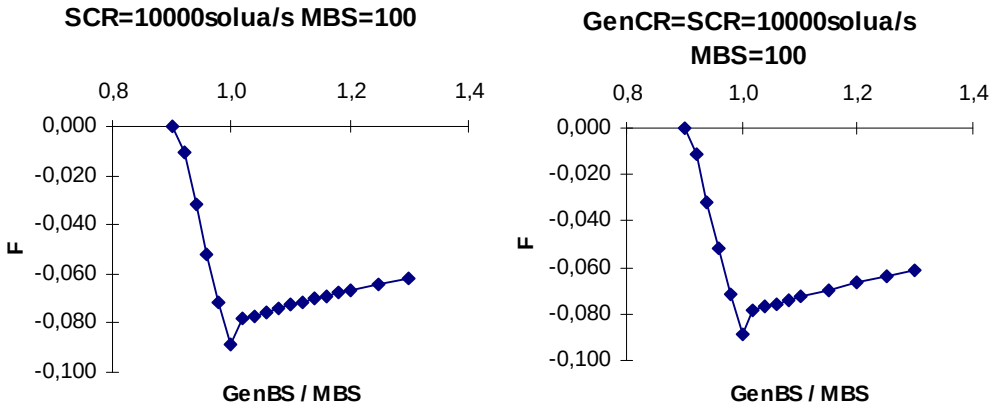
UPC	PCR(0)	PCR(0+1)	SCR	MBS	CDVT
1		4831			6
2		4831			12
3		4831			22
4		4831			91
5		80000			6

6	80000			12
7	353207	10000	100	12
8	353207	10000	100	12



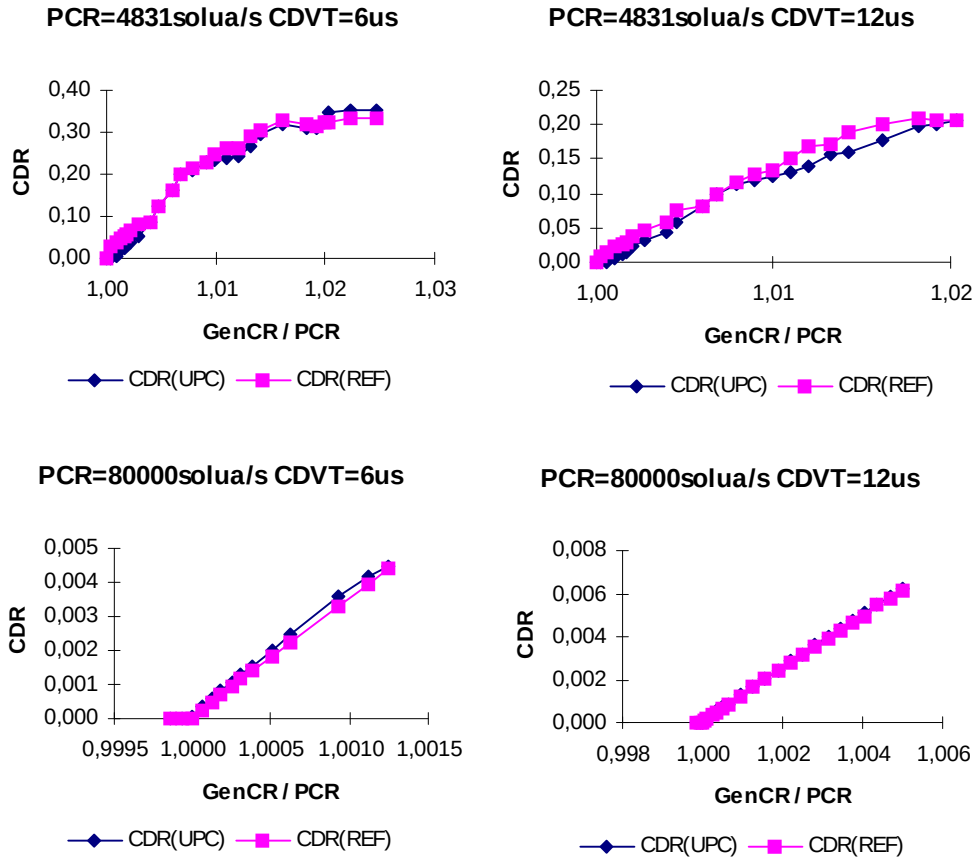
Kuva 5-11 UPC-prosessin hyvyysluku (F) erilaisilla parametrivariaatioilla.

Pienen suhteellisen nopeuseron omaavat oikean puoleiset kuvat riveillä yksi ja kolme on saatu absoluuttisella nopeuden ylityksellä. Algoritmin suhteellinen luonne näkyy selvästi absoluuttisten mittausten tulosten selvänä erona.

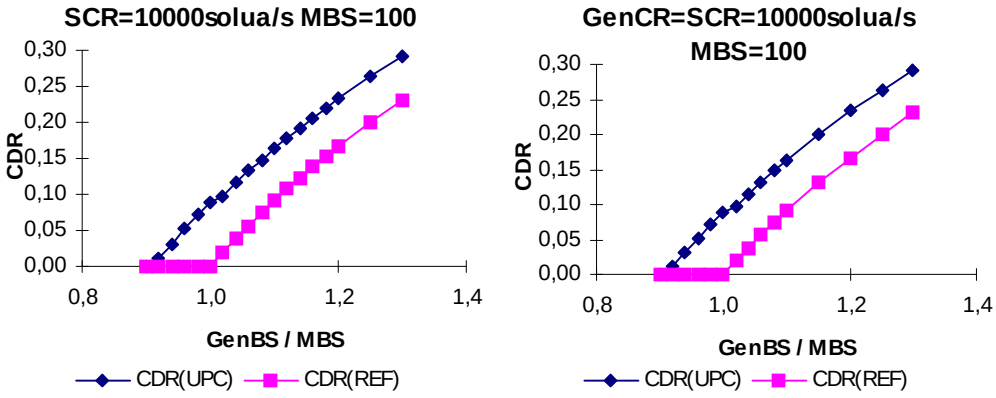


Kuva 5-12 UPC-prosessin hyvyysluku (F) on / off-purskeilla

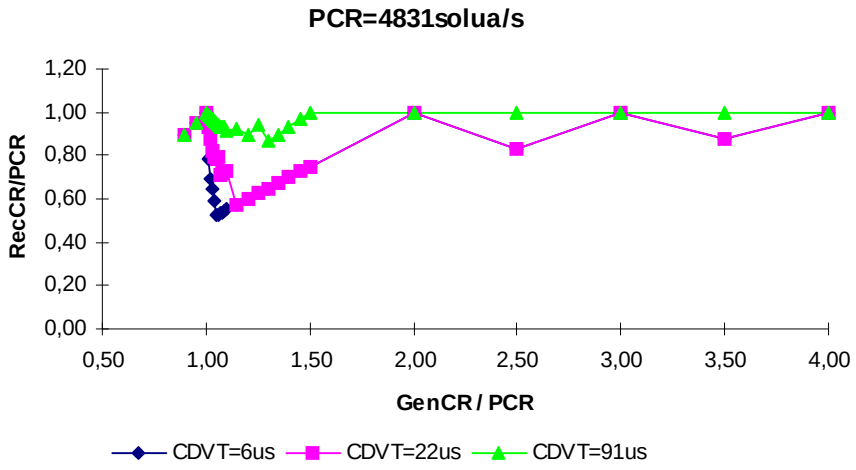
Oikean puoleinen kuvaaja on saatu vakiomalla generoitu solunopeus siten, että purskeen koon kasvaessa niiden suhteellinen osuus pienenee. Tällä ei kuitenkaan näytä olevan merkitystä vaan algoritmi reagoi puhtaasti purskeiden pituuteen.



Kuva 5-13 CDR: kytkentälaitteessa $\gamma_{P\,(UPC)}$ ja mittalaitteen referenssi $\gamma_{M\,(REF)}$

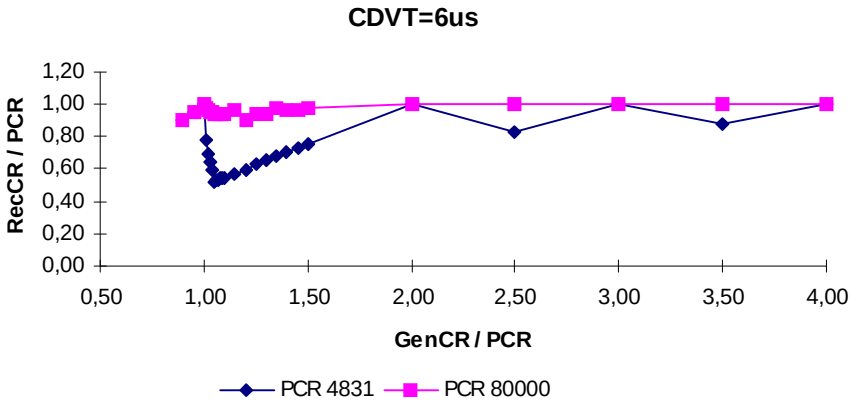


Kuva 5-14 CDR: kytkentälaitteessa $\gamma_{P\,(UPC)}$ ja mittalaitteen referenssi $\gamma_{M\,(REF)}$



Kuva 5-15 CDVT:n vaikutus läpi menevään solunopeuteen.

Kuvaajat on saatu generoimalla liikennettä sopimusta alittavalla ja ylittävällä nopeudella. CDVT:n vaikutus näkyy ensimmäisen karsintakuopan syvyytenä.



Kuva 5-16 Sopimuksen PCR-nopeuden vaikutus läpi menevään solunopeuteen

5.3.2 Parametrivalvonta väylätasolla

Väylätasolla parametrivalvontaa suoritetaan erikseen jokaiselle virtuaaliväylälle. Virtuaaliväylän sisällä yksittäiset liikennelähteet saavat käyttää siirtokaistaa täysin vapaasti aina väylän maksimikapasiteettiin asti. Kun maksimikapasiteetti ylitetään tulisi parametrivalvonnan rajoittaa liikennettä väylällä. Valvontaprosessi on muuten identtinen verrattuna virtuaalikanaviin.

Koska parametrivalvonta tapahtuu väylälle ei yksittäisillä yhteyksillä ole selvää mahdollisuutta hallita omaa palveluaan. Solujen karsinta perustuu kokonaisliikenteeseen, jolloin yhteyksien kokeman karsinta jakautuu tilastollisesti.

5.3.3 Yhteenveto tuloksista

Tulokset osoittavat, että parametrivalvonta on varsin vaikea toteuttaa suosituksen mukaisena eksaktina valintaprosessina. Mittaustuloksista on nähtävissä, että vakionopeuksinen liikenne on huomattavasti helpompaa hallita, kuten saattoi jo etukäteen olettaa. Vaihtelevan nopeuden mukana tuoma purskeisuus hallitaan kohtuullisella tarkkuudella, tosin toiminta ei ole läpinäkyvää vaan algoritmi reagoi ennen kuin sopimusta edes on rikottu. Algoritmin toiminnan epätarkkuus vaihtelevanopeuksisella liikenteellä on selitettävissä pursketoleranssin ja maksimipurskekoon väliseen suhteeseen. ITU:n GCRA-algoritmi toimii aikatasossa, kun taas laitteet vastaanottavat parametrit purskekokona. Näin ollen toinen arvoista on jatkuva ja toinen diskreetti. Laitteiden sisäinen ratkaisu edellyttää aina jonkinlaista porrastusta aika-akselille, tämä saattaa aiheuttaa poikkeamaa BT:n ja MBS:n välille. Mahdollinen porrastus ilmenisi suorittamalla sama mittaus hieman eri arvoilla.

Toisaalta ero mittalaitteen referenssi algoritmiin on kaikissa mittauksissa hyvin pieni, mikä antaa luottamusta prosessin toimintaan jo nykyiselläänkin. Tuloksista nähdään, että pienet nopeudet ovat selvästi hankalampia hallita. Tämä johtunee algoritmin tavasta toimia aikatasossa. Kun perusaikayksiköitä vaaditaan riittävän monta solujen saapumisaikojen välillä, nousee virheen mahdollisuus huomattavasti.

Tarkasteltaessa CDVT:n vaikutusta solunopeuteen havaitaan selvästi, että CDVT:n arvo kannattaa aina valita käytettävän lähteen puitteissa mahdollisimman isoksi. Tämä takaa sen, että parametrivalvontaprosessin suorittama solujen

karsinta, etenkin pieni nopeuksisilla CBR-yhteyksillä, ei aiheuta vastaanottavan päätelaitteen solunopeuden vaihtelun kompensointiin tarkoitetun puskurin alivuotoa.

6 PUSKUROINNIN MITTAUKSIA

ATM-verkossa hyödynnettävä tilastollinen kanavointi mahdollistaa hetkittäisen ylikuormituksen syntymisen kytkentälaitteen eri johdoille. Kuormituksen vaihtelu yksittäisellä johdolla on stokastinen prosessi, jolla on tietty odotusarvo ja varianssi. Koska informaation siirrolle on asetettu varsin tiukat kriiteerit solujen kadottamisen suhteen, tarvitaan kytkentälaitteisiin puskureita.

Puskurointitekniikoita on useita, kuten luvussa 2. esitettiin, ja niillä on selvästi erilaiset vaikutukset yhteyksiin ja niiden kokemaan palvelutasoon. Puskuroinnissa on oleellista missä vaiheessa kytkentää se suoritetaan, miten jonot on järjestetty ja miten tapahtuu palveluun valinta eri jonoista.

Mikäli yhteys on reaaliaikainen ei sitä voida liialti puskuroida, vaan ylikuormitetun portin täytyy antaa yhteydelle korkeampi prioriteetti kuin soluhukkakriittiselle yhteydelle. Tämä johtuu sovelluksen luonteesta. Reaaliaikainen sovellus ei hyödynnä uudelleen lähetystä päätelaitteessa. Koko yhteyden kokema palvelunlaatu perustuu ajoissa välitettyihin soluihin - myöhästynyt solu on yhtä arvoton kuin hukkunut solu.

Soluhukasta kärsivät yhteydet ovat sovelluksia, joille hetkittäinen viive ei vaikuta sovelluksen kokemaan palveluntasoon. Tärkeämpää on puskuroida soluja, jotta soluhukka ja siten uudelleenlähetykset eivät kasvaisi liian suuriksi.

Kyseiset ominaisuudet määrittelevät prioriteetit, joilla soluja välitetään ja karsitaan estoa kokevalla johdolla. Normaalisti aikapriorisoidut solut välitetään ennen hukkapriorisoituja. Mikäli aikapriorisoitu solu on ollut matkalla tietyn aikarajan yli voidaan se karsia ylikuormittuneesta puskurista, koska sen merkitys vastaanottavalle sovellutukselle on mitätön. Vasta tämän jälkeen hylätään hukkapriorisoituja soluja. [8]

6.1 PUSKUROINTITEKNIIKOIDEN VAIKUTUS SUORITUSKYKYYN

Esitettävät analyyttiset ratkaisumallit perustuvat kirjallisuudesta löytyviin ratkaisutapoihin [2, 4]. Ratkaisut olettavat puskureiden olevan johtokohtaisia FIFO *(First In First Out)*-palvelurutiinia noudattavia puskureita. Käytännössä puskurointi suoritetaan jollain kehittyneemmällä periaatteella.

Oletetaan, että tuleva liikenne jokaisella (NxN)-kytkimen tulojohdolla on riippumatonta, identtistä Bernoulli-prosessin mukaista. Silloin todennäköisyys, että solu saapuu tulojohdolla tietyssä aikayksikössä on *p*, mistä saadaan tulojohdon kuormitukseksi keskimäärin:

$$E[a] = p$$

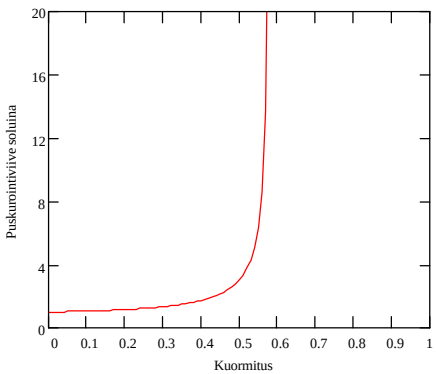
Lisäksi jokaisella solulla on yhtäläinen todennäköisyys (*1/N*) osoitukselle kullekin lähtöjohdolle. Tästä seuraa todennäköisyys sille, että johdolta saapuva solu, joka on osoitettu tiettylle lähtöjohdolle on (*p/N).*

6.1.1 Tulopuskuroinnin suorituskyky

Tulopuskuroinnin tarkastelussa on tarkasteluhetken alkutilanne valittava siten, että jokaisella tulojohdolla on liikennettä ja että tasapainotila on muodostunut. Tasapainotilassa muodostunut jonotusmalli on Geo/Geo/1 eli solujen saapumisajat ja järjestelmän palveluajat ovat geometrisesti jakautuneita.

Geometrisessa järjestelmässä jononpituudelle voidaan johtaa Z-muunnoskaava, jonka derivaatan arvo pisteessä z=1 antaa jonon pituuden odotusarvon. Tuloksen tarkastelu voidaan suorittaa, joko jonon pituutena tai keskimääräisenä odotusaikana jonojärjestelmässä. Littlen kaava suhteuttaa keskimääräisen odotusajan, keskimääräisen asiakkaiden lukumäärän systeemissä sekä keskimääräisen palveluajan. Lopputuloksena keskimääräiselle odotusajalle saadaan:

$$\overline{W} = \frac{1-p}{q-p} = \frac{p^2 - 3p + 2}{p^2 - 4p + 2}$$



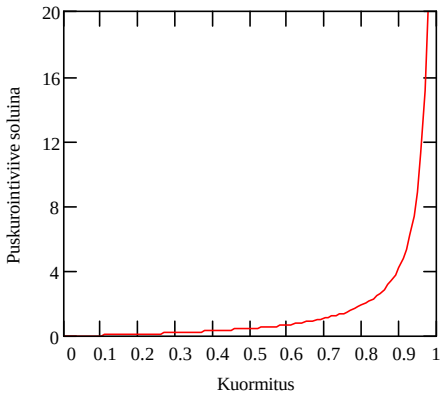
Kuva 6-1 Keskimääräinen odotusaika tulopuskuroidussa kytkimessä

6.1.2 Lähtöpuskuroinnin suorituskyky

Lähtöpuskurointi on puskurointi vaihtoehdoista yleisimmin käytetty. Toteutustapa voi olla joko jaettu tai kohdistettu. Suorituskyky yksittäiselle johdolle kohdistetulle puskurille on helppo ratkaista, sillä solujen saapumistodennäköisyydet perustuvat binomijakaumaan. Z-muunnos binomijakautuneille satunnaismuuttujille antaa keskimääräisen solujen saapumisintensiteetin, josta jonon pituuden johtaminen on suoraviivainen tehtävä.

Keskimääräisen jononpituuden kaavasta on huomattavissa, että lähtöpuskuroinnin keskimääräinen jononpituus Bernoulli-tuloprosessilla vastaa M/D/1-jonojärjestelmää. Eron muodostaa skaalaustermi (N-1)/N mutta, kun N lähestyy ääretöntä yhtyvät tulokset. Käyttämällä Littlen kaavaa saadaan keskimääräiseksi odotusajaksi järjestelmässä [2, 4]

$$\overline{W} = \frac{N-1}{N} \cdot \frac{p}{2 \cdot (1-p)}$$



Kuva 6-2 Keskimääräinen odotusaika lähtöpuskuroidussa kytkimessä

Jaettua muistia hyödyntävä puskurointi saavuttaa suorituskyvyn osalta samat arvot kuin kohdennettupuskurointi. Säästöä saadaan kuitenkin siinä, että muisti on dynaamisesti jaettua, jolloin muistin tarve jää huomattavasti pienemmäksi kuin yksittäisiä lähtöpuskureita hyödyntävässä ratkaisussa. Muistin säästöä saavutetaan jo noin kymmenen johdon laitteissa, mikä selittääkin ratkaisun yleisyyden. Jaettu puskurimuisti on kooltaan noin viidesosa verrattuna yksittäisiin puskureihin.

Analyyttisesti tarkastelua voisi suorittaa muodostamalla konvoluutio kaikkien lähtöpuskurien kuormitusjakaumasta ja asettamalla reunaehdot solun

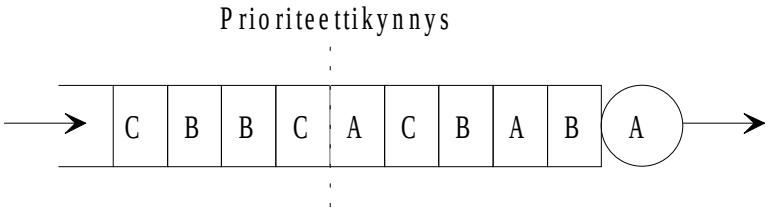
hukkaamistodennäköisyydelle (10^{-9}). Näin saataisiin tarvittava puskurinkoko eri liikenteillä. Mikäli sisääntuloprosessi ja liikenteen jakauma kaikille lähtöjohdoille on tasainen ei jaetulla puskuroinnilla saavuteta etua - mutta reaalimaailman tilanteissa, joissa erilaiset liikennevaihtelut ja jakaumat ovat vallitsevia saadaan suuria säästöjä puskuroinnin kustannuksissa.

6.2 PUSKURIEN KOKO JA JAKOPERUSTE

Yksittäisen liitynnän / johdon puskuri voi olla rakenteeltaan hyvinkin moninainen, toisaalta ratkaisu voi yksinkertaisimmillaan olla yksi FIFO-puskuri, kuten edellä suoritetussa analyysissä oletettiin.

6.2.1 Yksi FIFO-jono

FIFO-puskuroinnissa eri yhteyksien solut tulevat yhteen jonoon saapumisjärjestyksessä. Tätä jonoa palvellaan, riippumatta yhteyden tyypistä, vakiojärjestyksessä. Näin ollen FIFO-jonossa ei voida toteuttaa priorisointia. Kehittyneempi ratkaisu tästä on tietyn kynnyksen omaava FIFO-jono. [8] Siinä viiveherkkien yhteyksien sallitaan täyttää jonoa maksimissaan kynnysarvoon asti, jonka jälkeen saapuvat viivepriorisoidut solut hukataan. Tämä solujen hylkääminen perustuu solujen kokemaan jonotusaikaan FIFO-jonossa ja siten viivepriorisoitujen solujen vähentyneeseen arvoon. Mikäli palvelurutiinia tai jonokuria muutetaan, voidaan yhteen jonoon järjestetty puskuri toimia hyvinkin tehokkaana ratkaisuna. Sen etu on tehokas muistikapasiteetin käyttö mutta haitaksi muodostuu jonokurin hankaloituminen.

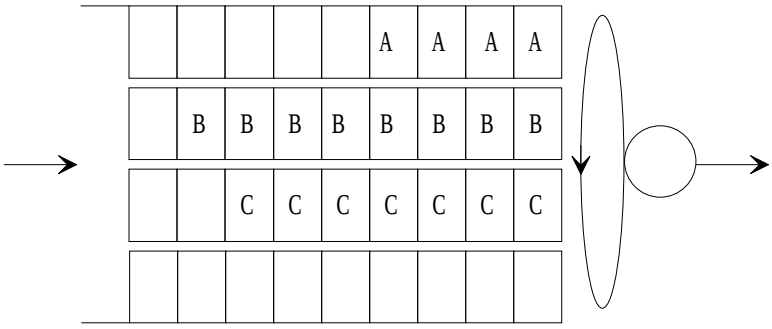


Kuva 6-3 FIFO-puskurointitekniikka

6.2.2 Rinnakkaiset jonot

Osittain rinnakkaiset jonot ovat yhteyksienhallinnan kannalta helpoin ratkaisu. Sen heikko puoli on kiinteästi toteutettuna huono hyötysuhde muistin käytölle. Etuna ovat kuitenkin priorisoinnin toteuttamisen helppous sekä mahdollisuus

yhteyksien yksilölliseen hallintaan. Mikäli fyysinen muisti jaetaan dynaamisesti yhteyksien hetkellisen tarpeen perusteella saavutetaan selvää muistin säästöä sekä mahdollisimman pieni soluhukkasuhde. [3, 39]



Kuva 6-4 Rinnakkaisten jonojen puskurointitekniikka

Rinnakkaiset jonot vaativat logiikan palvelun jakamiselle eri jonojen välillä. Palvelun jakaminen perustuu aina priorisointiin. Mikäli jako tapahtuu yksinkertaisena kiertävänä menettelynä (tasaprioriteetti), taataan jokaiselle yhteydelle reilu pääsy linkille. Tasaprioriteetti aikaansaa lineaarisen riippuvuuden jonotusviiveelle ja solunopeudelle. Nopeusriippuvuuden huono puoli on yhteyksien siirtoviiveen vaihtelun lisääntyminen yhteyksien nopeuserojen suhteessa. Yleisessä tapauksessa painokertoimet eri jonojen välillä eroavat. Painokertoimet määrätyvät yhteyden tyypin perusteella eli lähteen vaatimasta verkkopalvelusta. Karkeimmassa tapauksessa prioriteetteja on neljä: kaksi kumpaankin vapausasteen suuntaan.

Viive \ Hukka	Korkea	Matala
Korkea	I	II
Matala	III	IV

Kuva 6-5 Prioriteettityypit ja niiden määräytyminen [13]

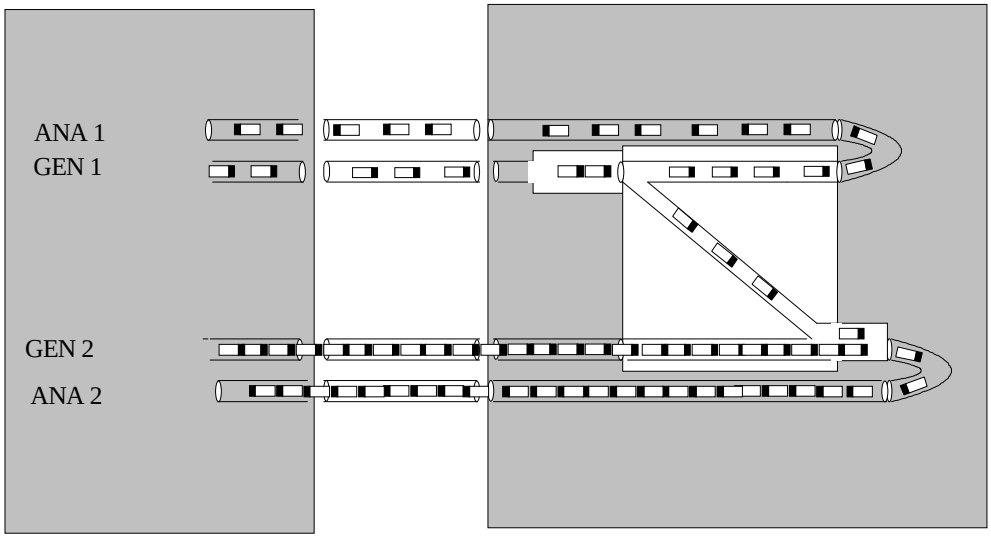
Käytännön painokertoimina voidaan käyttää esimerkiksi yhteyden pyytämiä CDVT ja CLR-arvoja. Tarkastelemalla CDVT:tä soluina/s ja esittämällä se kahden potenssina sekä CLR:stä eksponenttia, saadaan matriisi, jossa on vaaka-

akselilla CDVT:n potenssit ja pystyakselilla CLR:n eksponentit. Näin on saatu muodostettua yksinkertainen prioriteettimatriisi, jossa on noin 64 prioriteettitasoa.

6.3 PUSKURIEN MITTAAMINEN

6.3.1 Puskurin sijainnin mittaaminen

Puskurin sijainti voidaan määrittää, kun muistetaan tulopuskuroinnissa esiintyvä HOL-esto. Mittauskytkentä ja mahdolliset jonon muodostukset nähdään kuvasta 6-5. Mikäli toisesta portista generaattorista generoidaan soluja täydellä nopeudella, ei siirtotielle jää tyhjää tilaa generaattorista yksi tuleville soluille. Mikäli kytkin hyödyntää tulopuskurointia pitäisi yhteyden, joka kulkee generaattorista yksi vapaalle johdolle kokea estoa. Esto havaitaan joko puskurin ylivuotona tai siirtoviiveen selvänä kasvuna verrattuna kuormittamattomaan tilaan. Mikäli kyseessä on lähtöpuskurointi pitäisi kyseisen yhteyden olla estoton.



Kuva 6-6 Puskurin sijainnin ja koon mittauskytkentä

Mikäli puskurointitapa on edellisten yhdistelmä tullaankin alueelle, jota on huomattavasti hankalampi kartoittaa. Lisäksi, jos jonojärjestelmä on jokin muu kuin FIFO on tulopuskurin määritys hankalaa. Toisaalta sopivan kuormituksen valinnan pitäisi aina tuoda esille HOL-estoa.

Mittauksessa ilmeni, että FORE hyödyntää lähtöpuskurointia tai varsin kehittynyttä tulopuskurointia. Estotilanne ristiinkytketyllä yhteydellä ei aiheuttanut soluhukkaa kummallaan suoraan kytketyllä yhteydellä.

6.3.2 Puskurin koon mittaaminen

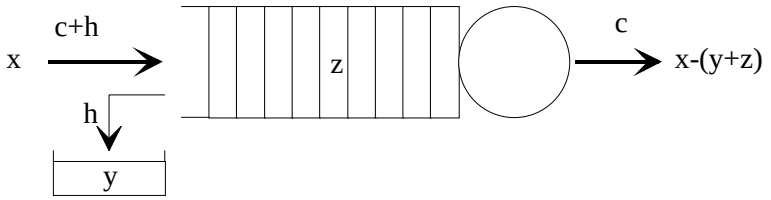
Puskurin koko vastaavasti on hankalampi mitata. Puskuroinnissa käytettävät jonojärjestelmät toiminnaltaan niin monimutkaisia, että mitään tarkkoja arvoja jonon pituudelle on lähesmahdotonta selvittää. Jonon pituudelle saadaan kyllä suuntaa antavia arvoja, joiden tarkkuus on varsin karkea.

Mikäli puskuri on yksittäinen FIFO-jono voidaan sen koko määrittää mittauksella, jossa johdolle ohjataan kaksi yhteyttä. Toinen yhteyksistä on hieman maksimi välityskapasiteetin alapuolella ja toinen yhteyksistä taas on on/off-tyyppinen. On/off-lähteen on-jakson pituutta vaihdellaan siten, että on-tilan lähetysnopeus on maksimi solunopeus. Näin ollen puskuri täyttyy ja tyhjenee määrävälein. Tämä periodinen siirtoviiveen vaihtelu antaa jonon pituuden siirtoviiveen minimin ja maksimin erotuksena. Mittaus on teknisesti ihanteellinen. Siirtoviiveen minimi huomioi kytkentälaitteen rakenteelliset ominaisuudet ja antaakin siksi oikean tuloksen jonon pituudelle. Lisäksi kytkentälaitteessa ei tapahdu solujen menetyksiä, jolloin mittalaitteen mittaustarkkuuden pitäsi olla paras mahdollinen. [17]

Toinen mahdollisuus on generoida liikennettä yli maksikuormituksen ja rekisteröidä maksimisiirtoviive, joka yhteydelle syntyy. Jonon täyttyminen aiheuttaa puskurin ylivuodon, jolloin välitettävät solut näkevät aina täyden jonon ja kokevat siten maksimisiirtoviiveen. Näin ollen siirtoviiveen tasapainojakauman tulisi keskittyä maksimisiirtoviiveen kohdalle, kunhan mittausperiodi on tarpeeksi pitkä. Jonon pituus saadaan selville siirtoviiveen minimin ja maksimin erotuksena.

Kolmas mahdolisuus on generoida liikennettä yli maksikuormituksen ja rekisteröidä satunnaisella ajanhetkellä lähetetyt-, hukatut- ja vastaanotetut solut. Näin saadaan selville jonon pituus systeemissä olevien solujen keskimääräisenä lumääränä. Tästä tuloksesta on kuitenkin vähennettävä tyhjän jonon tapauksen systeemissä olevien solujen lukumäärä. Mittaus on teknisesti heikko mutta antaa nopean tuloksen, mikäli tarkkoja aikoja ei voida mitata.

Kaikki edellä mainitut menetelmät soveltuvat suoraan ainoastaan FIFO-jonolle, jossa liikennelähteillä on yhtenevät prioriteetit. Toisaalta käytänön jonot ovat kaukana yhdestä FIFO-jonosta. Tämän takia tulokset antavat karkean likiarvon, josta voidaan päätellä vain tuloksen dekadi.



Kuva 6-7 Jonon pituuden mittaus ylivuotomenetelmällä

..... x Lähetetyt solut c Linkin nopeus

y ..... Hukatut solut h..... Ylikuorma

z ..... Jonossa olevat solut

Jonon pituus mitattuna menetelmällä kaksi:

$$Z = \frac{CTDmax - CTDmin}{2.8316us}$$

Yhden solun lähetysaika 155Mbit/s SDH-liitynnässä on 2,8316us.

Mittaus antaa kaksi erillaista tulosta riippuen käytetystä liitäntäkortista. Sarjan A liitäntäkortti antoi jonon pituudeksi 248 solua, kun taas sarjan C liitäntäkortin tulos riippui kuormituksesta ja valitusta palvelutyypistä. UBR-liikenne antoi pisimmän jonon, kun yksi CBR-lähde kuormitti lähes koko johdon kapasiteetin. UBR-liikennettä puskuroitiin tällöin 9126 solua.

6.4 YHTEENVETO PUSKUROINNIN MITTAUKSISTA

FOREn puskurointitapa suoritettujen mittausten perusteella näyttää olevan lähtöjohdon puskurointia, sillä mittauksissa ei ilmennyt ylimääräistä solujen menetystä tai viivästystä mittalaitteen portissa yksi. Toisaalta puskurin kokoa oli mahdotonta määrittää, koska kokonaismuistia jaettiin dynaamisesti eri yhteyksille.

Koska fyysinen toteutus on usein sidoksissa binäärilukuihin, voidaan mitatuista arvoista päätellä, että sarjan A liitäntäkortilla olisi ollut 256 (2^8) solun jono ja sarjan C jononpituus olisi 8192 (2^{13}) solua. Koska C sarjan moduleissa on rinnakkaisia jonoja täytyy kokonais puskurikapasiteetin olla vähintään 16384 (2^{14}) solua. Nämä yleistykset ovat varsin karkeat mutta antavat kuvan siitä miten hankalaa on arvoida jonon todellista pituutta mitatuista tuloksista.

Ylikuormittamalla johtoa suurella määrällä erillaisia yhteyksiä olisi teoriassa mahdollista selvittää puskurin maksimikoko, mutta mittauksen luotettavuus on kyseenalainen.