Lasse Laaksonen

Verkkoresurssien dynaaminen jako

Tietotekniikan
pro gradu -tutkielma
9. toukokuuta 2007

Jyväskylän yliopisto

Tietotekniikan laitos

Jyväskylä

Tekijä: Lasse Laaksonen
Yhteystiedot: lplaakso@cc.jyu.fi
Työn nimi: Verkkoresurssien dynaaminen jako
Title in English: Dynamic resource allocation
Työ: Tietotekniikan pro gradu -tutkielma
Sivumäärä: 80
Tiivistelmä: Tietoliikenneverkkojen kuormitus ja palvelunlaadun vaatimukset ovat kasvaneet viime vuosina reaaliaikaisten multimediasovellusten yleistyttyä. Sen vuoksi verkko-operaattoreiden tulee pystyä jakamaan verkkoresurssinsa siten, että tuotto kyetään maksimoimaan ja asiakkaille voidaan taata palvelutaso, josta he ovat maksaneet. Tämä pro gradu -työ käsittelee palvelunlaatuun ja hinnoitteluun liittyviä menetelmiä, joilla verkon resursseja voidaan jakaa dynaamisesti.
English abstract: Load of the telecommunication networks and requirements of Quality of Service have been growing increasingly in past few years as the multimedia applications have generalized. Therefore network operators should be able to share their network resources in a way that a revenue can be maximized and customers receive a fair Quality of Service. This master thesis considers different resource sharing mechanisms, which enables ensuring of QoS and profit maximization.
Avainsanat: tietotekniikka, pro gradu -tutkielma, palvelunlaatu, Integrated Services, Differentiated Services, hinnoittelu, adaptiivinen skedulointi, dynaaminen resurssien jako
Keywords: information technology, master's thesis, Quality of Service, Integrated Services, Differentiated Services, Pricing, adaptive scheduling, dynamic resource sharing

Sanasto

AF	Assured Forwarding
BE	Best Effort
CAR	Committed Access Rate
CBWFQ	Class-Based Weighted Fair Queuing
CL	Controlled Load
CoS	Class of Service
DiffServ	Differentiated Services
DSCP	Differentiated Services Code Point
EF	Expedited Forwarding
FIFO	First-In-First-Out
FRTS	Frame-Relay Traffic Shaping
FTP	File Transfer Protocol
GS	Guaranteed Services
GTS	Generalized Traffic Shaping
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IntServ	Integrated Services
IPv4	Internet Protocol version 4
LAN	Local Area Network
LE	Lower Effort
LFI	Link Fragmentation and Interleaving
NBAR	Network-based application recognition
P2P	Peer-to-peer
PBR	Policy-Based Routing
PHB	Per Hop Behaviour
PQ	Priority Queuing
RED	Random Early Detect
RSVP	Resource ReSerVation Protocol

RTP	Real Time Protocol
SLA	Service Level Agreement
SLO	Service Level Objectives
SLS	Service Level Specification
TCA	Traffic Conditioning Agreement
TCP	Transmission Control Protocol
ToS	Type of Service
UDP	User Datagram Protocol
QoS	Palvelunlaatu
URL	Uniform Resource Locator
VoIP	Voice over Internet Protocol
WFQ	Weighted Fair Queuing
WRR	Weighted Round Robin

Sisältö

1 Johdanto

Nykypäivänä tietoliikenneverkoista on muodostunut korvaamaton väline erilaisille organisaatioille, sekä yksityishenkilöille. Andrew M. Odlyzko toteaa artikkelissaan *Internet traffic growth: Sources and implications* [9], että Internetin liikenteen määrän vuosittainen kasvu tulee pysymään tulevaisuudessa 70 ja 150 prosentin välissä (kasvaa keskimäärin tuplaten vuodessa) kuten se on tehnyt vuodesta 1997 lähtien. Internetin liikenteen määrän kasvaessa on syntynyt myös tietokoneverkkoja hyödyntäviä erilaisia sovelluksia organisaatioiden ja tavallisten kotikäyttäjien tarpeisiin. Yritysten kansainvälistyminen ja sitä kautta toimipisteiden leviäminen eripuolille maapalloa ovat asettaneet tietoliikenneverkoille aivan uudenlaisia tarpeita, joihin verkoista vastaavien operaattoreiden täytyy pystyä vastaamaan. Usein kansainvälisiin projekteihin liittyy monia osapuolia, jotka saattavat toimia maantieteellisesti kaukana toisistaan. Koska matkustaminen vie paljon rahaa ja aikaa, tarvitaan riittävästi verkkokapasiteettia esimerkiksi videoneuvotteluiden pitämiseen. Lisäksi voi olla tarve siirtää suurikokoisia tiedostoja projektin osapuolien välillä. Yritysmaailmassa tarvitaan siis paljon verkkokapasiteettia, jotta kommunikaatio osapuolien välillä toimii sujuvasti. Sairaaloissa saatetaan hyödyntää verkkoa esimerkiksi röntgenkuvien siirtämiseen paikkakunnalta toiselle ja lääkärit voivat avustaa leikkauksissa videokuvan välityksellä. Myös yksityishenkilöiden Internetin käyttö aiheuttaa verkon kapasiteetin kuormittumista: koteihin hankittavien Internet-liittymien määrä on kasvussa sekä yhä useammat älypuhelimet tarjoavat Internet-liitynnän ja tätä kautta mahdollisuuden käyttää liki samoja palveluita kuin tavallisella tietokoneella. Vertaisverkkoliikenne (**P2P**) – jossa jaetaan musiikkia, sovelluksia ja videoita – vie suuren osan verkkokapasiteetista. *Garrett Hardin* esittää artikkelissaan *The Tragedy of the Commons* [2] vertauksen, jolla hän demonstroi sitä kuinka vapaa pääsy ja rajoittamaton tarve rajalliseen resurssiin saa aikaan kohteena olevan resurssin väärinkäyttöä ja lopulta sen loppumisen. Yksilö yrittää maksimoida resurssinkäyttönsä, koska saavuttaa siitä tiettyä hyötyä, kun taas väärinkäytön kustannukset jaetaan kaikkien resurssia käyttävien yksilöiden kesken. Tämä sama huomio pätee myös nykyiseen Internet-arkkitehtuuriin, jossa kaikkia tietoliikennepaketteja palvellaan pääpiirteittäin samanarvoisina: tarjolla oleva Internet-kaista toimii yhteisenä resurssina, josta yksittäiset asiakkaat pyrkivät ottamaan käyttöönsä mahdollisimman suuren osuuden. Siten asiakas saa käyttöönsä muita paremman suorituskyvyn.

Tietoliikenneverkkojen nykyinen tapa käsitellä paketteja samanarvoisesti ei aiheuta ongelmia web-sovelluksille, joita ovat esimerkiksi verkkosivut, sähköposti ja uutisryhmät. Esimerkiksi multimediasovellukset sen sijaan kärsivät muiden käyttäjien huonosta verkkokäyttäytymisestä. Edellä mainittu ongelma herättää tarpeen luokitella verkkoliikennettä erityyppistä suorituskykyä tarjoaviin palveluluokkiin. Tietty palvelu saattaa aiheuttaa vain pienen määrän verkkoliikennettä, mutta tarvitsee pitkäkestoisen yhteyden reaaliaikaisella vasteella. Esimerkiksi tiedonsiirtoa varten tarvitaan hetkellisesti jopa megatavujen purskeita. Verkon ylitse siirrettävä ääni ja liikkuva kuva tarvitsevat vähintään tietyn suuruisen tiedonsiirtonopeuden, jotta ei tapahtuisi nopeuden hidastumista. Mikäli verkko käsittelee kaiken liikenteen samalla tavalla, se saattaa johtaa huonoon palvelunlaatuun. Esimerkiksi samassa verkkoalueessa tapahtuva puhe (**VoIP**) ja tiedostojen siirto (**FTP**) häiritsevät toisiaan: tietyllä kaistanleveydellä tapahtuva puheensiirto saattaa aiheuttaa FTP-liikenteeseen ruuhkia ja pakettien häviämistä. Purskeisen FTP-liikenteen vaikutukset saattaa huomata VoIP-puhelussa äänen pätkimisenä ja sen laadun huonontumisena. Tämän tyyppisten ongelmien vuoksi Internet-infrastruktuuria tulisi parantaa siten, että verkossa liikkuvat paketit jaettaisiin liikenneluokkiin. Luokittelemalla paketteja voitaisiin palvella yksilöllisesti, jakaa verkkoresursseja luokkien kesken ja pystyttäisiin kontrolloimaan reaali-aikaliikennettä häiritsevää päästä-päähän viivettä. [1]

Sen lisäksi, että liikenteen luokittelun avulla voidaan ratkaista sovellusten suorituskykyyn liittyviä ongelmatilanteita, on pakettien luokittelulle myös ekonomiset syyt. Koska Internetistä on tullut yrityksille liiketoiminnan kannalta välttämätön väline, voi operaattori myydä yrityksille erityistä suorituskykyä, jolloin kustannukset riippuvat saadusta palvelunlaadusta. Kun liikenteen luokittelu yleistyy Internetissä, useammat yrityksistä ja loppukäyttäjistä saavat reilumpaa palvelua kasvattaen samalla operaattorien kokonaistuottoa. Myös verkon ylläpitokustannukset laskevat kun olemassa olevia verkon resursseja pystytään jakamaan tehokkaammin.

Tässä luvussa esitellään varsinainen tutkimusongelma, sekä aiheeseen liittyviä tekniikoita, menetelmiä ja viitekehyksiä. Keskeisiä asioita ovat reitittimen toiminta resurssien jaon näkökulmasta, palvelunlaatu (**QoS**), integroitu palvelinarkkitehtuuri (**IntServ**) ja palveluiden erottelukäytäntö (**DiffServ**). Toisessa luvussa esitellään adaptiiviseen skedulointiin liittyviä tekniikoita: reititin on keskeinen fyysinen laite verkossa, joka sisältää paketteja käsitteleviä mekanismeja kuten skedulereita, joilla verkon suorituskykyä on mahdollista parantaa. Kolmannessa kappaleessa tutustutaan erilaisiin viimevuosina kehitettyihin hinnoittelumenetelmiin. Neljännessä kappaleessa esitellään työn käytännön osuus, jossa analysoidaan eräällä adaptiivisella kaistanjako mallilla ajettuja simulaatioita. Viidennessä kappaleessa tehdään yhteen-

veto, jossa summataan aiemmissa kappaleissa käsitellyt aiheet.

1.1 Tutkimusongelma

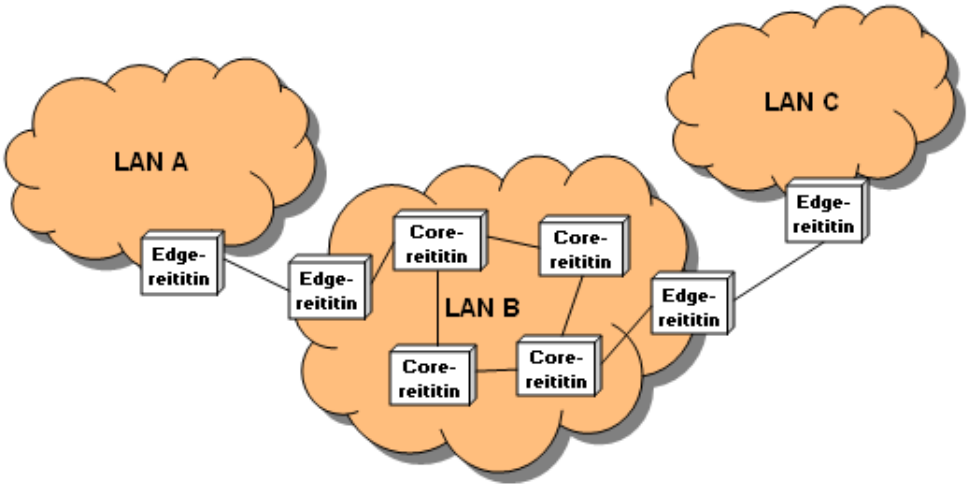
Ajankohtainen kysymys tietoliikenneverkoista puhuttaessa on se kuinka tietoliikenneverkkoja hallinnoivat palveluntarjoajat pystyisivät jakamaan verkkoresurssinsa tehokkaasti, tuottoa maksimoiden sekä asiakkaiden käyttämien sovellusten erilaiset vaatimustasot tyydyttäen. Aiheesta on kirjoitettu tieteellisiä tutkimuksia, joista ensimmäiset ovat peräisin 1990-luvun lopusta. Aihealueeseen liittyvät tekniikat ovat ehtineet kehittyä tähän päivään mennessä ja ongelmaan kehitetyt ratkaisut koostuvat monenlaisista eri tekniikoista. Erilaisia verkkoresurssien hinnoittelumenetelmiä on kehitetty useita ja tunnetuimmat niistä perustuvat joko kiinteään laskutukseen tai käyttöön perustuvaan laskutukseen. Verkon resursseja voidaan jakaa lukuisilla menetelmillä, joita esitellään kappaleessa 3. Tarkoituksena on esitellä myös viimeisimpiä alan tutkimuksia, joissa on yhdistetty verkkoresurssien dynaaminen jako ja palveluiden hinnoittelu. Tämän tutkimuksen käytännön osuutta varten on ajettu simulaatioita, joiden avulla tutkitaan artikkelissa [22] esiteltyä adaptiivista skedulointialgoritmia. Simulaatiosta saatuja tuloksia analysoidaan tarkemmin ja katsotaan kuinka skedulointialgoritmin käyttämien parametrien muuttaminen vaikuttaa palveluntarjoajan saamaan kokonaistuottoon, sekä luokkien painokertoimiin ja kaistankäyttöön.

1.2 Reitittimet

Reititin toimii jaettuna laitteena tietoliikenneverkossa ja se yhdistää useampia verkkoja toisiinsa lähettäen datapaketteja näiden välillä. Kytkin yhdistää yksittäisiä koneita lähiverkkoon (**LAN**), kun taas reititin yhdistää lähiverkkoja toisiinsa. Pakettien perille toimittamiseen eli reititykseen, reititin käyttää apunaan reititysprotokollia. Reititin ylläpitää niin kutsuttua reititystaulua reititysprotokollan tarjoaman informaation avulla. Reititystaulussa säilytetään reitittimeltä tiettyyn kohdeosoitteeseen kulkevia yhteyksien reittejä, reitittimiin liittyvää mittatietoa, sekä seuraavan reitittimen osoitetta.

Reitittimet jakautuvat niin kutsuttuihin reunareitittimiin (engl. *edge-router*) ja sisäreitittimiin (engl. *core-router*) [kuva 1.1]. Isompia verkkoalueita yhdistävät reunareitittimet sijaitsevat verkkoalueiden laidoilla, jonka vuoksi niiden läpi kulkee huomattavasti suurempi määrä liikennettä kuin verkon sisällä sijaitsevien reitittimien

kautta. Reunareititin pystyy näin ollen keräämään vuokohtaista statistiikkaa ja sen vuoksi niissä on järkevintä tehdä liikenneluokittelu ja pakettien käsittely. Sisäreitittimet ohjaavat verkkojen sisällä kulkevaa liikennettä ja tarjoavat paketeille luokittelun mukaista palvelua parhaan kykynsä mukaisesti.



Kuva 1.1: core- ja edge-reitittimet.

1.3 Ethernet-verkon palvelunlaatua tukevat mekanismit

Ethernet-verkossa datapaketteja voi kadota ja uudelleenlähetykset voivat aiheuttaa ruuhkatilanteita. Pakettien katoamisen voi aiheuttaa, joko verkon reitittimen alimitoitettu sisääntulopuskuri tai liian pieni puskuritila ulosmenevälle portille. *Best Effort*-tyyppisessä liikenteessä pudotetaan paketteja myös sovelluksilta, joille luotettavuus on välttämätöntä. Ethernetin **CoS**-mekanismit perustuvat pakettien luokitteluun, jossa reitittimet voivat luokkatietojen perusteella pudottaa alempiarvoisiksi luokiteltuja paketteja reitittimen liitännän jonosta. Tällä tavoin kriittisemmät eli parempaan luokkaan kuuluvat paketit voidaan säilyttää reitittimen jonossa. Pakettien luokittelussa suurin ongelma on oikeiden mekanismien löytäminen, jotta kaikkia palveluluokkia kohdellaan reilusti ja toisaalta parhaat luokat saavat sopimuksen mukaista palvelua. Varsinaisesta palvelunlaadusta puhutaan vasta kun jokaiselle luokalle voidaan taata tietynlainen palvelutaso. **IETF**-yhteisön kehittämät IntServ- ja DiffServ-arkkitehtuurit tarjoavat työkaluja, joilla voidaan toteuttaa erilaisia palvelutasoja. Ne mahdollistavat myös uusien hinnoittelumallien kehittämisen ja toteuttamisen, jonka vuoksi luvut 1.6 ja 1.5 esittelevät kummankin arkkitehtuurin pääpe-

riaatteet ja ominaisuudet. [3]

1.4 Palvelunlaatu

QoS-arkkitehtuuri tarjoaa tavan määritellä liikenneluokille suorituskykyparametreja. QoS-arkkitehtuurin päämääränä on jakaa verkkoresursseja siten, että kaikki suorituskykyyn liittyvät vaatimukset täyttyvät. Mukaan on tullut myös taloudellinen näkökulma: palveluntarjoaja voi myydä asiakkaalle tietynlaatuista suorituskykyä, jossa palvelunlaatu on riippuvainen hinnasta. Palvelun hinta ja siitä riippuva suorituskyky sovitaan asiakkaan ja palveluntarjoajan välisessä **SLA**-sopimuksessa, josta kerrotaan lisää luvussa 1.4.1. QoS-arkkitehtuuriin liittyvät kolme peruskomponenttia ovat:

- QoS identifiointi- ja merkitsemistekniikat, joilla saadaan taattua palvelunlaatu päästä päähän verkkoelementtien välillä.
- Verkkoelementin sisäinen QoS, kuten jonotusmekanismit, vuoronjako ja liikenteenmuokkaus työkalut.
- QoS kuri-, hallinta- ja monitorointitoiminnallisuudet, joilla saadaan kontrolloitua verkon läpi kulkevaa liikennettä.

[4]

1.4.1 Service Level Agreement

Asiakkaan halutessa tietoliikenneyhteydelleen tiettyä palvelunlaatua, ensimmäinen askel on tehdä palveluntarjoajan kanssa sopimus palvelun tasosta. Tämä sopimus kulkee QoS-konseptissa nimellä *Service Level Agreement*. Sopimuksessa määritellään niin kutsuttuja palveluparametreja kuten oletettu kuorma tietyllä aikavälillä, toimenpiteet kuorman ylittäessä sovitun raja-arvon (esim. hinta). SLO-sopimus (engl. *Service Level Objectives*) määrittelee mittarit, joiden avulla voidaan valvoa SLA-sopimuksen toteutumista. SLS (engl. *Service Level Specification*) kuvaa tavan, jolla palveluntarjoajan tulee käsitellä asiakkaan verkkoliikennettä. Asiakas ja palveluntarjoaja määrittelevät sopimukseen tarkat laatuvaatimukset, jotka ilmoitetaan käyttäen tunnettuja suorituskykyparametreja, jotta palveluntarjoaja voi kytkeä ne osaksi reitittimen kokoonpanoa (engl. *configuration*) ja valvoa niiden toteutumista. Suorituskyvyn mittaamisessa käytettäviä perusparametreja ovat:

- reitittimen läpäisevä liikenne (engl. *throughput*),
- viive (engl. *delay*),
- viiveen vaihtelu (engl. *jitter*) ja
- pakettihäviö (engl. *packet loss*).

Throughput-parametri määrittelee liikennemäärän tavuissa/biteissä, jonka sovellus pystyy lähettämään häviöittä tietyn aikayksikön sisällä. Tämä parametri on käytössä useimmissa palveluluokissa. Läpimenevä liikenne lasketaan yleensä keskiarvona pidemmältä aikaväliltä, koska lyhyellä aikavälillä tehdyssä mittauksessa saattaa näkyä suuret hetkelliset liikennemäärän muutokset. *Delay*-parametri on kriittinen parametri reaaliaikasovelluksille. Paketin kokonaisviiveen muodostumiseen vaikuttavat siirtotien ominaisuudet, reitittimen kyky välittää paketteja eteenpäin, sekä paketin viettämä aika reitittimen sisäisessä jonossa. *Jitter*-parametri määrittelee peräkkäisten pakettien välisen viiveen vaihtelun. Parametri on tärkeä esimerkiksi määriteltäessä liikenneluokkaa VoIP-puheluille tai videoneuvotteluille. *Packet loss*-parametri ilmoittaa yhteyden aikana hylättyjen pakettien määrän. Tätä käytetään mittarina sovelluksille, jotka tarvitsevat taatun kaistanleveyden. Sovellus joutuu lähettämään paketin uudelleen aina kun reititin hylkää paketin. Pakettihäviöt aiheuttavat myös puhe- ja videoliikenteen laadun heikkenemistä. [1]

Tehtäessä palvelusopimusta asiakkaan kanssa tulee ottaa huomioon sovellukset, jotka saattavat lähettää dataa sopimuksessa mainittua rajaa suuremmalla nopeudella. Tämän vuoksi operaattorin on pakko käyttää liikennettä rajoittavia malleja, jotta sovellukset eivät häiritse muiden sovellusten toimintaa. Useimmat mallit perustuvat stokastisiin eli satunnaisiin prosesseihin, jotka ovat liian yksinkertaisia kuvaamaan lähdeosoitteen parametreja tai liian monimutkaisia mukautuvaan analyysiin. Muut käytössä olevat mallit rajoittavat liikennettä sen sijaan, että mallintaisivat prosessia tarkasti.

1.4.2 Vuo eli datavirta

Jotta haluttu palvelunlaatu voidaan taata, täytyy verkon yksilöidä erityistä kohtelua vaativille paketeille oma vuo (engl. *flow*). Yleinen tapa määritellä vuo on yhdistää kohde- ja lähdeosoitteet, kohde- ja lähdeportit, sekä sessiotunnus. Toisin sanoen vuo on sarja peräkkäisiä ja toisiinsa kuuluvia paketteja joilla on sama lähde- ja kohdeosoite. Vuoksi voidaan myös käsittää mikä tahansa paketti, joka tulee reitittimeen tietyltä sovellukselta tai sisääntulevalta liitynnältä. [4]

1.4.3 Pakettien identifiointi ja merkitseminen

Pakettien luokittelu koostuu voiden identifioimisesta ja merkitsemisestä. Ennen kuin tietynlaiselle liikenteelle voidaan tarjota sopivaa palvelua, täytyy se identifioida. Kun paketti identifioidaan, mutta jätetään merkitsemättä, sanotaan luokittelua hyppykohtaiseksi. Luokittelu pätee vain sen hetkisellä laitteella eikä kyseistä luokittelua käytetä enää seuraavalla laitteella. Kun paketteja merkitään verkonlaajuiseen käyttöön, voidaan luokittelu tehdä asettamalla niin kutsuttuja **IP precedence**-bittejä päälle tiettyyn järjestykseen. **IPv4**-protokollaa käytettäessä precedence-bitit sijaitsevat paketin otsakkeen **ToS**-kentässä.

QoS-arkkitehtuuria tukevan reitittimen on osattava tulkita vuokohtaisia parametreja, jotta kutakin reitittimen läpäisevää pakettivuota voidaan palvella SLA-sopimuksen mukaisesti. Koska parametrit täytyy etsiä jokaiselle reitittimelle saapuvalle paketille, vie operaatio aikaa. Jokaiselle reitittimelle saapuvalle paketille tulee etsiä oikeat parametrit luokittelun määrittävästä tietorakenteesta. Aikaa saadaan kutistettua merkitsemällä palveluluokka IP-otsakkeeseen.

CAR-ominaisuudella on mahdollista asettaa edellä mainittuja precedence-bittejä perustuen laajennettuun pääsylistaan (engl. *extented access list*). Tällä menetelmällä voidaan määrittää liikenneluokka muun muassa käyttäjän, sovelluksen, lähde- tai kohdeosoitteen perusteella. Menetelmää käytetään mahdollisimman kaukana verkon reunoilla, jotta seuraavat verkkoelementit voivat tarjota määritellyn säännön mukaista palvelua. **NBAR**-menetelmä identifioi liikennettä huomattavasti karkeammin: **HTTP**-paketin **URL**-osoite voidaan identifioida, jonka jälkeen paketille voidaan asettaa precedence-bitit.

Myös **PBR**-menetelmä mahdollistaa liikenteen luokittelun, joka perustuu laajennettuun pääsylistaan. PBR:n avulla on myös mahdollista asettaa precedence-bittejä. Määrittämällä saapuvalle liikenteelle tärkeystaso ja käyttämällä sitä tietoa yhdessä jononhallintamekanismien kanssa voidaan luoda eriytetty palvelu (engl. *Differentiated Services*). [4]

1.4.4 Verkkoelementin sisäinen palvelunlaatu

Verkkoelementin sisäisestä palvelunlaadusta huolehtivat skedulerit, linkin käytön optimointi, liikenteen viivästyttämistyökalut ja pakettien pudotusmekanismit.

Joskus purskeiset reaaliaikasovellukset aiheuttavat reitittimien jonojen tukkeutumisen ja näin ollen ruuhkia verkkoon. Reitittimen skedulerit on suunniteltu purkamaan ruuhkatilanteita reitittimien liitynnöillä, siten että paketteja pudotetaan mahdollisimman vähän ja pakettien edelleenlähetyksessä tarjotaan kaikille liikenneluo-

kille mahdollisimman reilua palvelua. Luvussa 2.2 käydään läpi tärkeimmät skedulerit vahvuuksineen ja rajoitteineen.

Reitittimen hitaissa linkeissä saattaa ajoittain ilmetä ongelmia suurien pakettien kanssa: suuren paketin viive saattaa olla niin pitkä, että takana tulevan viivekriittisen paketin viiveraja ylittyy ennenkuin paketti on edes lähtenyt reitittimeltä. Näissä tapauksissa linkin käyttöä voidaan optimoida pilkkomalla suurta pakettia pienempiin osiin (engl. *fragmentation*). Pilkkominen ei pelkästään riitä mikäli viivekriittinen paketti jää pilkottujen pakettien taakse. Pakettien lähetystä täytyy siis lomittaa (engl. *interleaving*). **RTP**-liikenteen otsakkeiden pakkaamisella voidaan pienentää verkon kuormaa. **LFI**-menetelmällä pystytään pienentämään viivettä ja viiveen vaihtelua hitaimmissa liitynnöissä pilkkomalla suuret paketit pienempiin osiin ja lomittamalla matalaviiveisiä paketteja pilkottelujen pakettien sekaan.

Liikenteen muotoilua (engl. *shaping*) käytetään rajoittamaan täyden kaistan käyttöä voilta. Näin voidaan välttää reitittimen lähetyspuskurin ylivuotaminen. Kaistamaksimin yläpuolella oleva liikenne bufferoidaan ja lähetetään myöhemmin, jolloin konfiguroitu lähetysnopeus ei ylity.

Liikenteen pudotus (engl. *policing*) rajoittaa myös kaistan käyttöä. Erona liikenteen muotoiluun on se, että kaistamaksimin yli meneviä paketteja ei puskuroida.

GTS ja **FRTS** ovat liikenteen muokkaus- ja pudotusmenetelmiä, jotka ovat mahdollista asettaa mukautumaan dynaamisesti vapaana olevaan kaistaan tai säätää kaistaa esiasetetun arvon suuruiseksi. [4]

1.4.5 Palvelunlaadun tasot

Palvelun laadun tasot määräytyvät sen mukaan minkälaista palvelunlaatua voidaan tarjota päästä-päähän eli toisin sanottuna minkälaisen palvelunlaadun verkko pystyy tarjoamaan eri liikennetyypeille koko yhteyden matkalta. Palvelut erottaa toisistaan QoS tason tiukkuus (engl. *QoS strictness*), joka kuvaa sen kuinka tarkasti palvelu voidaan sitoa tiettyyn parametriin (kaistanleveys, viive, viiveen vaihtelu tai pakettien pudotus). Palvelunlaadulle on määritelty kolme tasoa, jotka voidaan tarjota päästä-päähän heterogeenisen verkon läpi:

- *Best Effort* palvelulla tarkoitetaan perusyhteyttä, jolle ei ole suorituskykytakeita. Tällaista palvelua voidaan tarjota esimerkiksi **FIFO**-jonoilla, jotka käsittelevät kaikkia voita samalla tavalla.

- *Differentiated Services* (tai *soft QoS*) tarkoittaa sitä, että jotakin liikennetyyppiä palvellaan toisia liikennetyyppejä paremmin. Tietylle liikennetyypille voidaan

tarjota esimerkiksi enemmän kaistaa, pienempää keskimääräistä viivettä tai keskimääräistä pienempää pudotettavien pakettien määrää. DiffServ ei lupaa kuitenkaan tarkkoja rajoja palvelunlaadun parametreille. DiffServ voidaan toteuttaa jakamalla verkkoliikenne luokkiin ja käyttämällä esimerkiksi skedulereita ohjaamaan liikenteen käyttäytymistä. DiffServ konseptia ja skedulereita käsitellään lisää myöhemmin tässä luvussa.

- *Integrated Services* (eli *hard QoS*) tasossa tehdään absoluuttinen verkkoresurssien varaus tietynlaiselle liikenteelle. Palvelunlaadun parametreille asetetaan tiukat rajat, joiden täsmällinen pitävyys taataan. IntServ:ssä käytetään asianmukaista protokollaa resurssien varaamiseen. IntServ konseptia käsitellään lisää myöhemmin tässä luvussa.

Tehtäessä päätöstä verkossa käytettävästä palvelutyypistä, täytyy ottaa huomioon seuraavat tekijät:

- Kukin palvelutasoista on tarkoitettu tietynlaiselle liikenteelle, joten on selvitettävä mitä sovelluksia asiakas käyttää ja minkälaisia ongelmia yritetään ratkaista. Joissain tapauksissa palveluntarjoa joutuu toteuttaa sekä DiffServ- että IntServ-tasot.
- Täytyy ottaa huomioon kuinka nopeasti palveluntarjoajat voivat realistisesti päivittää järjestelmänsä uuteen tekniikkaan sopivaksi.
- IntServ-palvelutason toteutus tulee mitä luultavimmin maksamaan enemmän kuin DiffServ-tason toteutus.

[4]

1.5 Differentiated Services

DiffServ-arkkitehtuurissa liikennevuot, jotka ovat piirteiltään samanlaisia, luokitellaan luokkiin joita kutsutaan aggregaateiksi. Paketin saapuessa DiffServ-verkkoon, sen IP-otsakkeeseen merkitään niin kutsuttu **DSCP**-arvo (engl. *DiffServ codepoint*), joka määrittää mihin palveluluokkaan paketti kuuluu. Reitittimet luokittelevat ja ajastavat pakettien lähetyksen DSCP-arvon perusteella. Resurssit allokoidaan liikenneluokille, toisin kuin IntServ-arkkitehtuurissa, jossa resurssit jaetaan voiden kesken. Mikäli jokin vuo liikenneluokan sisällä alkaa kuluttaa enemmän resursseja, aiheuttaa se toisille luokan voille pakettien läpimenon hidastumista. Tämän vuoksi

DiffServ-verkossa mitataan ja säännöstellään voita. Reitittimet jaetaan sisä- ja reunareitittimiin. Reunareitittimet suorittavat raskaimpia operaatioita kuten pakettien luokittelua, mittausta ja merkitsemistoimintoja, kun taas sisäreitittimet suorittavat yksinkertaisempia operaatioita kuten pakettien edelleenlähetyksiä. DiffServ sisältää tietyt liikenneluokat, joiden paketit saavat ennaltamääriteltyä kohtelua reitittimiltä. Hyppykohtaiset säännöt (engl *Per-Hop Behaviour*) määrittelevät sen kuinka kutakin luokkaa tulisi kohdella. Hyppykohtainen sääntö liitetään tiettyyn pakettiin DSCP-tunnisteen avulla. Reititin päättää mekanismit, joilla haluttu kohtelu pystytään tarjoamaan. DiffServ-arkkitehtuuri määrittelee kolme tärkeintä PHB-tyyppiä: **EF** (engl. *Expedited Forwarding*) , **AF** (engl. *Assured Forwarding*) ja **BE** (engl. *Best Effort*). [1]

1.5.1 Differentiated Services arkkitehtuurimalli

DiffServ-toimialueeseen (engl. *DiffServ domain*) kuuluu DiffServ-solmuja (reitittimiä), joihin on implementoitu palveluiden provisiontikäytönnöt ja PHB luokat. Toimialueella on selvästi määritellyt reunat, joilla toimivat rajasolmut (engl. *DS Boundary Node*). Rajasolmut toimivat yhdyskäytävinä toisiin DiffServ-toimialueisiin sekä muihin toimialueisiin, joissa ei käytetä DiffServ-arkkitehtuuria. Sisäsolmut (engl. *DS Interior Node*) ovat yhteydessä vain saman toimialueen solmuihin. Sekä raja- että sisäsolmujen täytyy pystyä liittämään paketti sopivaan PHB-tyyppiin. Usein rajasolmujen täytyy kyetä luokittelemaan paketteja ja suorittamaan raskaita liikenteen muokkaus-operaatioita.

DiffServ-rajasolmut voidaan jakaa vielä kahteen tyyppiin: vastaanottaviin solmuihin (engl. *Ingress Node*) ja lähettäviin solmuihin (engl. *Egress Node*). Liikenne saapuu DiffServ-toimialueelle vastaanottavan solmun kautta ja lähtee toimialueelta lähettävän solmun kautta. Vastaanottava solmu huolehtii, että toimialueiden välinen sopimus liikenteen muotoilusta (engl. *Traffic Conditioning Agreement*) toteutuu. Lähettävän solmu hoitaa liikenteen muotoilutoimintoja.

DiffServ-toimialue koostuu tavallisesti yhdestä tai useammasta saman ylläpitäjän hallinnoimasta verkosta. Toimialueen ylläpitäjä on vastuussa siitä, että resurssit jaetaan alueen SLA-sopimusten mukaisesti. DiffServ-alueet voidaan yhdistää suuremmiksi alueiksi (engl. *DiffServ region*). Kun tietyn palvelun reitti menee useampien DiffServ-toimialuiden läpi, täytyy jokaisen toimialueen tehdä SLA-sopimus (sisältää **TCA**:n), jossa määrillään se kuinka liikennettä käsitellään toimialueiden rajoilla. DiffServ-alueelle (region) pystytään toteuttamaan yhteinen luokittelukäytäntö, jolloin liikennettä ei tarvitse erikseen käsitellä toimialueitten rajoilla. [5]

1.5.2 DiffServ palveluluokat

EF-luokka on suunniteltu tarjoamaan päästä päähän palvelua alhaisella pakettihäviöllä, latenssilla, jitterillä sekä taatun kaistan DiffServ-verkkoalueen läpi. EF on suunniteltu sovelluksille, joka käyttävät **UDP**-protokollaa tiedonsiirtoon, koska sen avulla voidaan siirtää reaaliaikaista ääntä ja videota halutulla nopeudella, vaikka joitain paketteja tiputettaisiinkin. Koska UDP ei reagoi pakettien hylkäämiseen, tälle luokalle voidaan käyttää yksinkertaista FIFO-puskuria.

AF-luokka on suunniteltu tarjoamaan erilaisia lähetyksen takaavia tasoja paketeille, jotka tarvitsevat BE-liikennettä parempaa palvelua. Luokka ei sisällä viiveeseen tai viiveen vaihteluun liittyviä vaatimuksia, vaan määrittelee neljä AF luokkaa: **AF1**, **AF2**, **AF3** ja **AF4**. Kullekin luokalle varataan oma minimikaista. Luokan sisällä paketit voidaan merkitä tietyllä arvonumerolla (engl. *drop precedence*) **AF21**, **AF22**, **AF23**). Kun DiffServ-reititin ruuhkaantuu, se alkaa tiputtamaan korkeamman arvonumeron omaavia paketteja. AF sopii TCP-sovelluksille, jotka voivat kasvattaa lähetysnopeutta ja reagoida pakettien hylkäämisiin pienentämällä lähetysikkunan kokoa. AF-luokalle käytetään jononhallintamekanismeja, jotka perustuvat **RED**-mekanismiin.

Taulukko 1.1: Assured Forwarding luokkien arvojärjestykset

	Luokka 1	Luokka 2	Luokka 3	Luokka 4
Low Drop	AF11	AF21	AF31	AF41
Med Drop	AF12	AF22	AF32	AF42
High Drop	AF13	AF23	AF33	AF43

BE-luokkaa käytetään lähettämään ei-kriittistä dataa, jolla ei ole QoS-vaatimuksia. Palveluntarjoaja voi varata kiinteän kaistan tälle luokalle, joka jaetaan aktiivisten datavoiden kesken. BE-luokkaa käyttävistä sovelluksista jotkut voivat olla tärkeämpiä kuin toiset, esimerkkinä sähköpostiliikenne. Näitä varten on kehitetty **LE**-luokka, jolle voidaan myöntää muilta toimettomassa tilassa olevilta luokilta jääneitä resursseja.

1.5.3 Huomioita

DiffServ-arkkitehtuurilla ei kyetä takaamaan yhteydelle päästä-päähän palvelunlaatua. Sen sijaan resursseja on mahdollista saada dynaamisesti juuri sen verran kuin verkolla on mahdollista tarjota, mutta kuitenkin palveluluokan määreiden ra-

joissa.

1.6 Integrated Services

Palveluntarjoajat haluavat päästä hallitsemaan kaistan jakamista yksittäisen linkin palveluluokkien kesken. Liikenne halutaan jakaa muutamaan hallittavaan luokkaan ja ruuhkatilanteessa jokaiselle luokalle pitää pystyä antamaan tietty minimiprosenttiosuus linkin kaistasta. Muina aikoina palveluluokkien käyttämätön kaista täytyy olla saatavilla. Luokat voivat koostua eri käyttäjäryhmistä tai protokollista. Kaistanhallintaominaisuudesta käytetään myös nimeä kontrolloitu linkin jako (engl. *Controlled link-sharing*). Integrated Services on Internetin palvelumalli, johon sisältyy BE-palvelu, reaaliaika-palvelu ja kontrolloitu linkin jako. IntServ-konseptissa kunkin käyttäjän resurssientarve tai saatavilla olevat resurssit ovat tarkasti määriteltyjä. Konseptissa luvataan pyydetty palvelunlaatu vain siinä tapauksessa, että se pystytään toteuttamaan koko yhteyden matkalta. Resurssien selvittämiseen käytetään **RSVP**-protokollaa, jolla reitittimet välittävät tietoa siitä kuinka paljon resursseja tarvitaan. Verkko vastaa pyyntöön joko posiivisesti tai negatiivisesti: se joko kertoo voivansa toteuttaa vaaditun palvelun sovituilla parametreilla tai ilmoittaa ettei voi toteuttaa pyyntöä ollenkaan. Pyyntö on voimassa niin kauan kunnes sovellus tai verkko ilmoittaa kyseisen liikenteen lopettamisesta. Protokollan avulla reittimet pystyvät konfiguroimaan itselleen ja sovellukselle sopivan QoS-profiilin. [1]

1.6.1 IntServ palveluluokat

IntServ-kehys määrittelee BE-luokan (engl. *Best Effort/NULL*) lisäksi kaksi muuta palveluluokkaa. **GS**-luokka (engl. *Guaranteed Service*) on tarkoitettu palveluille, jotka tarvitsevat takuun siitä, että paketit saapuvat perille täsmälleen luvattuun aikaan eikä paketteja hylätä jonojen ylivuodon takia. IntServ-reitittimet laskevat kokoajan arviota huonoimmalle mahdolliselle viiveelle ja varaa sen mukaan kaistaa, jotta luvattua enimmäisviiverajaa ei ylitetä. **CL**-luokka (engl. *Controlled Load*) on kehitetty palveluille, jotka ovat herkkiä ruuhkatilanteille. Luokan tavoitteena on tarjota riittävä kaista ja puskuritila, jotta voidaan taata kaistavaatimukset ja minimoida pakettihäviö. BE-luokka ei tarjoa sovelluksille minkäänlaisia QoS-vaatimuksia eikä se näin ollen käytä RSVP-protokollaa. Jokainen RSVP-protokollaa tukeva reititin ylläpitää listaa GS- ja CL-luokkien voista, ja kaikki vuot jotka eivät ole listalla ohjataan suoraan BE-luokkaan. [1]

1.6.2 Liikenteen kontrollointi IntServ-arkkitehtuurissa

Reitittimen toimintoa, joka luo uusia erityyppisiä palvelunlaatuja kutsutaan liikennekontrolliksi (engl. *Traffic control*). Liikenteen kontrollointiin tarvitaan kolmea eri komponenttia: skeduleria (engl. *Packet scheduler*), luokittelijaa (engl. *Classifier*), sekä pääsynvalvontaa (engl. *Admission control*). Jotta liikennettä voitaisiin ohjata, täytyy saapuvat paketit liittää jonkun palveluluokan jäseneksi. Kaikki samaan luokkaan kuuluvat paketit saavat samanlaista kohtelua skedulerilta. Luokittelijan tehtävä on liittää reitittimelle saapuvat paketit oikeisiin luokkiin. Luokittelu voidaan tehdä esimerkiksi IP-paketin otsakkeessa olevan luokittelunumeron perusteella. Luokka voi käsittää esimerkiksi kaikki reaaliaika-vuot tai kaikki tietylle organisaatiolle kuuluvat vuot. Toisaalta luokka voi sisältää vain yhden vuon, jolloin palveluntaso saadaan erittäin korkeaksi. Luokka on abstraktio, joka voi olla paikallinen, määritelty vain tietylle reitittimelle. Sama paketti voi olla luokiteltu eri tavalla reitin varrella olevilla reitittimillä. Esimerkiksi runkoreitittimet voivat ohjata monia voita muutamaan koottuun luokkaan, kun taas verkon reunoilla sijaitsevat reitittimet, joihin ei kasaannu niin paljon liikennettä voivat käyttää erillistä luokkaa jokaiselle jonolle. Skedulerin tehtävänä on hoitaa eri pakettivirtojen ohjaus reitittimen ulosmenoportille käyttämällä jonoja sekä ajastimia. Erilaisia pakettiskedulereita on esitelty luvussa 2.2. Pääsynvalvonta toteuttaa algoritmin, jota reititin käyttää selvittämään voidaanko uudelle vuolle myöntää pyydetty palvelunlaatu ilman että se häiritsee muille voille sovittujen palvelusopimusten toteuttamista luvatulla tavalla. Pääsynvalvontaa suoritetaan reitin jokaisella solmulla jossa tehdään erikseen päätös pystytäänkö palvelunlaatua tarjoamaan. [7]

1.6.3 Huomioita

Yhdistettäessä IntServ-arkkitehtuuria ja adaptiivisia skedulereita tulee huomioida, että IntServ-reititin varaa kaistaa yksilöllisille voille, kun taas adaptiiviset skedulerit varaavat resursseja palveluluokille. Ongelma voidaan ratkaista kun laitetaan reitin ryhmittelemään samaan luokkaan kuuluvat vuot ja jakamaan resursseja näin muodostuneiden luokkien kesken. Adaptiivinen skeduleri tulisi sijoittaa jokaiselle RSVP-reitittimelle, jossa on pullonkaulaksi muodostuneita linkkejä. RSVP-protokolla ei kuljeta mukanaan hinnoittelutietoa. Kuitenkin palveluntarjoaja voi liittää staattisen hinnan jokaiseen palveluluokkaan, jolloin hinnoittelutietoa ei tarvitse kuljettaa RSVP-viestissä. Toisaalta RSVP-protokolla on helposti laajennettavissa, joten uuden kentän luominen pakettiin ei muodostuisi ongelmaksi. [1]

IntServ takaa vuokohtaisen palvelunlaadun kommunikoivien osapuolien välille.

RSVP-protokollan avulla varatulle reitille voi sattua myös reitittimiä, jotka eivät tue protokollaa. Tällöin kyseinen osio yhteydestä suoritetaan BE-liikennöintinä. Paketin kulkeman reitin varrella oleville laitteille tehdyt resurssien varaukset ovat *kevyitä* joka tarkoittaa sitä, että varausten voimassaolo tai tila pitää päivittää aika ajoin. Tämä lisää liikennettä verkossa ja lisää mahdollisuutta, että varaus vanhenee mikäli päivitystietoja kuljettavat paketit hukataan. Resurssivarauksista joudutaan ylläpitämään tilatietoja jokaisella reitittimellä ja pääsynvalvonta täytyy suorittaa jokaisella hypyllä. Näiden lisäksi muistinkulutus reitittimellä kasvaa, koska joudutaan tukemaan suurta määrää resurssien varauksia. Edellä mainitut seikat lisäävät jokaisen reitin varrella olevan laitteen kompleksisuutta. Ja koska jokaista varausta varten joudutaan ylläpitämään tilatietoja jokaisella reitin varrelle sattuvalla reitittimellä, tulee kaikkien verkossa liikkuvien voiden hallinnasta monimutkaista. Koska RSVP-protokolla on helposti laajennettavissa, näihinkin ongelmiin on kehitetty jo ratkaisuja, joista voi etsiä tietoa muun muassa otsikoilla *RSVP Refresh Reduction and Reliable Messaging*, *RSVP scalability enhancements* ja *Proxy RSVP*. [8]

1.7 Yhteenveto

Yksinään sovellusten monimuotoisuus ja verkon käyttäjien erityyppiset suorituskykyvaatimukset pakottavat miettimään ratkaisuja, joilla verkkojen rajalliset resurssit saadaan riittämään. Nykyisen BE-luokan sijaan liikennettä tulisi luokitella jollain järkevällä tavalla. IntServ- ja DiffServ-arkkitehtuurit tarjoavat työkalut, joilla liikenteen luokittelu voidaan toteuttaa erilaisten palvelunlaadun tasojen toteuttamiseksi. Täsmällistä QoS-vaatimusten mukaista palvelunlaatua voidaan tarjota IntServ-arkkitehtuurin avulla. IntServ-arkkitehtuurissa tehdään resurssiin varaus tietyillä vaatimusparametreilla, jotka tulee täyttyä yhteyden koko matkalta. DiffServ-arkkitehtuurissa taas palveluluokille määritellään suurpiirteisempää palvelunlaatua: reitittimen varrella olevat laitteet yrittävät teteuttaa QoS-vaatimukset parhaan kykynsä mukaan. Edellä mainituista IntServ on raskaampi toteuttaa, koska varauksien käsittelyjä joudutaan implementoimaan jokaiselle IntServ-arkkitehtuuria tukevalle solmulle. Lisäksi RSVP-protokollan paketit kuormittavat verkkoa varsinkin ruuhkatilanteissa, jolloin voidaan joutua tekemään uudelleenlähetyksiä.

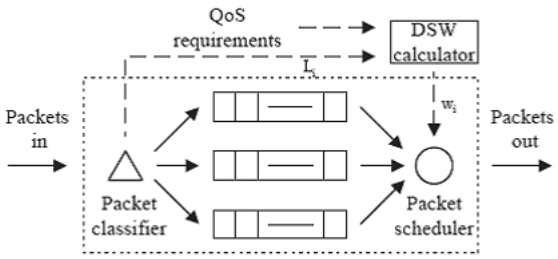
Seuraavassa luvussa esitellään verkonlaitteen sisäisiä jononhallintamekanismeja, jotka ovat tärkeässä asemassa jaettaessa verkon resursseja eri liikennetyypeille.

2 Adaptiivinen skedulointi

Reititin on verkon resursseja jakava laite. Resurssien määrä on rajoitettu ja niistä kilpailee suuri määrä erilaisia sovelluksia. Palveluntarjoajan tehtävänä on jakaa reitittimien ulostuloporttien kaistaa kaikille voille siten, että kukin vuo saa tasapuolisesti oman osuutensa kaistasta. Koska verkon sovelluksilla on keskenään erilaisia suorituskykyvaatimuksia, saattaa vuo (sovellus) vaatia käyttöönsä tietyn minimimäärän ulosmenokaistaa. Tästä syystä kaistaa ei ole mahdollista jakaa samansuuruisina osina voiden kesken. Lisäksi verkossa tapahtuu ruuhkatilanteita, jolloin reitittimessä tapahtuva resurssien jako on entistä tärkeämmässä roolissa: verkon ruuhkatilanteet johtuvat useimmiten reitittimissä ilmenevistä ylikuormitusongelmista. Ruuhkatilanne syntyy kun paketteja saapuu reitittimen ulosmenoportille nopeammin kuin paketteja voidaan lähettää sieltä eteenpäin verkkoon. Ruuhkan aiheuttama viive voi vaihdella edellisen paketin lähetyshetkestä äärettömyyteen, jolloin paketti on jo jouduttu pudottamaan puskurin ylikuormituksen vuoksi. Pakettien läpimenomäärän pienentyminen (engl . *throughput*), päästä-päähän mitatun viiveen kasvaminen (engl. *end-to-end delay*), viiveen vaihtelu (engl. *jitter*), sekä pakettien häviäminen (engl. *packet loss*) ovat seurausta siitä, että reitittimen puskurimuisti on riittämätön varastoimaan kaikkia jonoon saapuvia paketteja. Reitittimien resursseja voidaan jakaa käyttämällä puskurienhallintamekanismeja (**RED**), mutta tehokkaampaa on käyttää adaptiivisia skedulereita tai niiden yhdistelmiä. [1] [6]

2.1 Adaptiivinen reititin

Adaptiivisen reitittimen peruskomponentteja ovat pakettiluokittelija, tietty määrä jonoja sekä skeduleri. Näiden lisäksi on uusi moduuli **DSW** (engl. *Dynamic Service Weight calculator*), joka on vastuussa käytössä olevan skedulerin parametrien laskemisesta. Se laskee parametrien avulla paketeille painokertoimia, jotka lähetetään skedulerille.



Kuva 2.1: Adaptiivisen reitittimen komponentit

2.2 Skedulerit

QoS-ympäristössä tärkeimpiä komponentteja ovat reitittimillä toimivat skedulerit eli jonotusmekanismit. Kehittyneet skedulerit pystyvät parantamaan verkon suorituskykyä tarjoamalla takeita esimerkiksi kaistasta, viiveestä, viiveen vaihtelusta ja pakettihävikistä. Skedulereiden avulla voidaan välttää ruuhkia ja tarjota reilua palvelua, jolla puolestaan saavutetaan vakaampi ja ennustettavampi verkon käyttäytyminen. Ruuhkatilanteita voidaan siis ennalta ehkäistä erottelemalla paketit palveluluokkiin skedulereiden (engl. *queue scheduling discipline* tai *queuing mechanism*) avulla. Skedulerin tehtävänä on valita seuraavaksi reitittimen ulosmenoporttiin ohjattava paketti. Näin voidaan dynaamisesti säätää portin käyttämän ulosmenokaistan suuruutta.

Skedulerille asetetaan useita vaatimuksia, joista sen tulisi suoriutua:

- Jakaa reitittimen ulosmenoportin kaistaa oikeudenmukaisesti eri vaatimukset omaaville, keskenään kilpaileville palveluluokille. Tämä onnistuu asettamalla luokille painokertoimet, joiden avulla ulostuloportin hetkellinen kaista määritellään.

- Suojata samaa ulostuloporttia käyttäviä palveluluokkia siten, että sopimattomasti kaistaa käyttävät luokat eivät pysty vaikuttamaan muiden luokkien saamaan kaistaan tai viiveeseen.

- Sallii palveluluokkien käyttää kaistaa, joka on jäänyt käyttämättä kaistan omaavalta luokalta.

- Tarjoaa algoritmin joka on mahdollista toteuttaa reitittimen rautaan. Kun algoritmi toteutetaan kiinteästi laitteen rautaan, voidaan sillä jakaa kaistaa reitittimen nopeimmissa liitynnöissä vaikuttamatta heikentävästi reitittimen edel-

leenlähetysnopeuteen. Jos algoritmia ei voida toteuttaa raudalle, se täytyy toteuttaa hitaammille liitynnöille ohjelmistona joka taas laskee järjestelmän suorituskykyä.

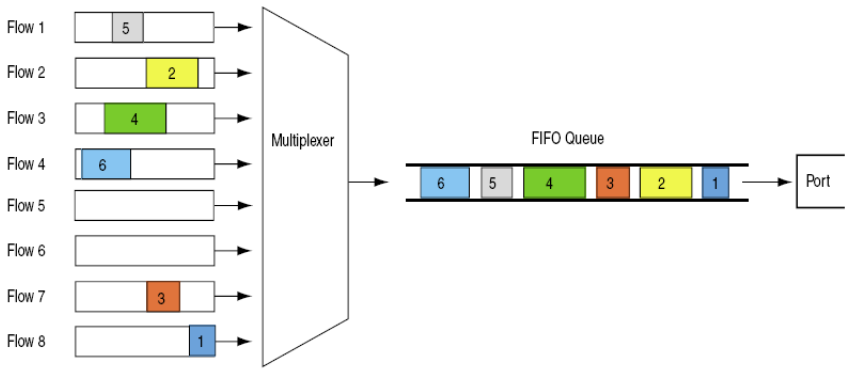
Skedulerit voidaan luokitella työnsäilyttäviin (engl. *work-conserving*) ja ei-työnsäilyttäviin (engl. *non-work-conserving*). Työnsäilyttävästä skedulerista on kyse silloin kun skeduleri ei ole toimettomana ja mikäli reitittimellä on paketteja lähetettävänä. Tämä luokka soveltuu käytettäväksi ympäristössä, jossa sovellus pystyy säätämään lähetettävän datan määrää (engl. *bit-rate*). Mikäli vapaata kaistaa on jäljellä, voi sovellus lähettää dataa suuremmalla nopeudella. Mikäli sovellus taas huomaa, että ruuhkatilanteessa paketteja on alettu pudottamaan, voi se pienentää lähetysnopeuttaan. Ei-työnsäilyttävä skeduleri voi olla lepotilassa, vaikka paketteja olisi lähetettävänä. Tämän tyyppiset skedulerit eivät muuta liikenteen profiilia vaan eliminoivat liikennevääristymiä viivyttämällä paketteja. Ei-työnsäilyttävä skeduleri ei osaa hyödyntää vapaana olevaa kaistaa vaan pyrkii täyttämään asetetut suorituskykyvaatimukset täsmällisesti. Ei-työnsäilyttäviä skedulereita ovat esimerkiksi **HRR** ja **RCSPQ** [11]. Edellä mainittuja skedulerityyppejä on vaikea käyttää yhtäaikaisesti, koska ne vaimentavat toistensa vaikutukset. Useimmat tietoliikennelaitteistojen valmistajat luottavat työnsäilyttäviin vuoronjakajiin ja myös DiffServ- ja IntServ-arkkitehtuurit suosivat niitä. Työnsäilyttävät skedulerit voidaan luokitella muutamiin pääluokkiin, jotka on nimetty tunnetuimpien skedulerien mukaan: First-Come-First-Served, Priority Queuing, Round-Robin ja Fair Queuing. [6] [1]

Seuraavissa kappaleissa käydään läpi tunnetuimpien työnsäilyttävien skedulereiden toimintaa, hyviä puolia, rajoituksia ja soveltuvuuksia. Läpikäytävät skedulerit ovat **FIFO**, **PQ**, **FQ**, **WFQ**, **WRR (CBWFQ)** ja **DWRR**. Lisäksi käydään läpi monimutkaisempia skedulereita, joita räätälöidään yhdistelemällä yksinkertaisimpien skedulereiden parhaat ominaisuudet.

2.2.1 First-In-First-Out

FIFO-skeduleri on kaikista jonohallintamekanismeista yksinkertaisin. Kaikki saapuvat paketit ohjataan yhteen jonoon ja niitä palvellaan saapumisjärjestyksessään. Näin ollen kaikkia paketteja kohdellaan samanarvoisina.

FIFO-skedulerin yksinkertaisuudesta johtuen sen etuna on erittäin pieni laskennallinen monimutkaisuus. Se on käyttäytymiseltään ennustettava, koska jonossa olevia paketteja ei järjestellä uudelleen ja jonon maksimiviive saadaan laskettua jonossa olevien pakettien määrän perusteella. Niin kauan kuin jonossa olevien pakettien määrä pysyy pienenä, FIFO tarjoaa tehokkaan ratkaisun resurssien jakamiseen.



Kuva 2.2: First-In-First-Out vuorottelija

Koska yksittäinen FIFO-jono ei tarjoa mahdollisuutta järjestellä puskurissa olevia paketteja, ei niitä voida myöskään palvella luokittaisesti. Ruuhkan lisääntyessä keskinmääräinen viive kasvaa kaikille liikennevoilla tasaisesti, joka voi johtaa kasvavaan viiveeseen, viiveen vaihteluun ja reaaliaika liikenteen pakettien häviöön. Ruuhkatilanteissa FIFO-mekanismi suosii UDP-voita TCP-voita enemmän, koska ruuhkatilanteissa TCP-pohjaiset sovellukset hidastavat lähetysnopeuttaan UDP-sovellusten jatkaessa lähettämistä tasaisella nopeudella. TCP-protokollaa käyttävät sovellukset yrittävät sopeutua muuttuneisiin verkko-olosuhteisiin hidastamalla lähetysnopeutta. Purskeinen vuo voi kuluttaa koko FIFO-jonon puskuritilan ja estää muiden voiden liikennöinnin, kunnes purskeinen vuo on palveltu loppuun.

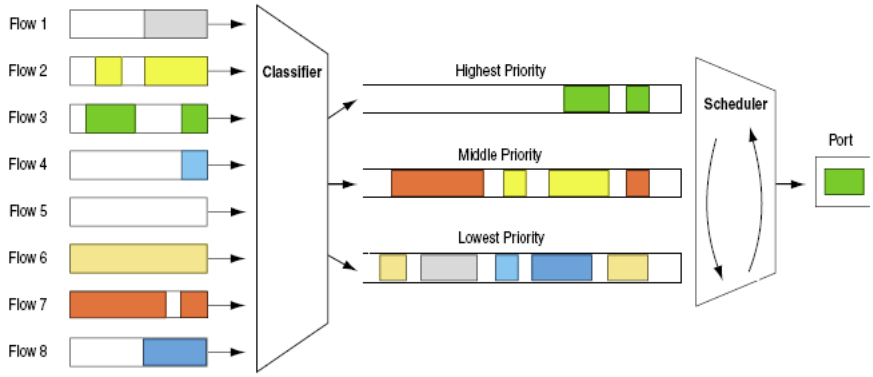
Käytännössä FIFO on tuettuna reitittimen ulosmenoportissa silloin kun muita vuorottelijoita ei ole konfiguroitu. [6]

2.2.2 Priority Queuing

Useimmat yksinkertaisimmista paketteja luokittelevista vuorottelualgoritmeista perustuvat *Priority Queuing* -skeduleriin. PQ-skedulerissa paketit luokitellaan ja sijoitetaan eri prioriteettijonoihin. Vuorossa olevan jonon ensimmäinen paketti käsitellään vain mikäli korkeamman prioriteetin jonot ovat tyhjiä. Kunkin prioriteettijonon sisällä paketit skeduloidaan FIFO-menetelmän mukaisesti.

Käytettäessä ohjelmistopohjaisia reitittimiä PQ-skeduleri aiheuttaa kohtuullisen pienen laskennallisen kuorman verrattuina muihin monimutkaisempiin skedulereihin. PQ-skeduleria käyttävän reitittimen on mahdollista järjestää puskurissa olevia paketteja ja näin kullekin luokalle voidaan tarjota erityyppistä palvelua.

PQ-skedulerin käytössä saattaa ilmetä myös ongelmia. Mikäli korkean priori-



Kuva 2.3: Priority Queuing skeduleri

teetin liikennettä ei rajoiteta verkon reunalaitteilla, matalamman prioriteetin liikenne saattaa kärsiä kohtuuttomasta viiveestä. Tämä siitä syystä, että matalamman luokan paketit joutuvat odottamaan kunnes jonoissa oleva korkean prioriteetin liikenne on palveltu kokonaisuudessaan. Jos taas korkean prioriteetin liikennemäärä kasvaa riittävän suureksi, matalamman prioriteetin liikenteelle varatut puskurit alkavat täyttyä ja lopulta paketteja joudutaan pudottamaan. Saman jonon sisällä sijaitsevat korkean prioriteetin vuot voivat häiritä toisiaan: mikäli jokin vuo käyttäytyy sopimattomasti varaten kohtuuttomasti suoritusaikaa, saattavat muut vuot kärsiä huomattavasta viiveestä ja viiveen vaihtelusta. PQ-skeduleria ei kannata käyttää jakamaan TCP-liikenteelle korkeampaa palvelunlaatua kuin UDP-liikenteelle. TCP:n vuon- ja ikkunanhallintamekanismit yrittävät käyttää kaiken ulostuloportissa vapaana olevan kaistan, jolloin alemman prioriteetin UDP-voiden palveluaika kärsii.

PQ-skedulerille on kaksi pääkäyttökohdetta:

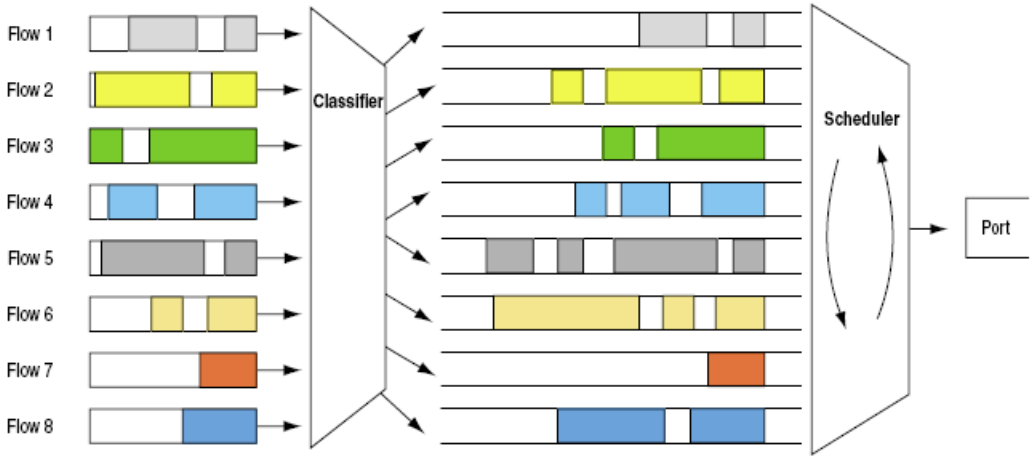
- Parantaa verkon luotettavuutta ruuhkatilanteissa kun reititinprotokolla- ja verkonhallintaliikenne ohjataan korkean prioriteetin jonoon.
- Tarjoaa paketeille suuren läpimenon, alhaisen viiveen ja viiveen vaihtelun, sekä pienen pakettihävikin. Kyseiset ominaisuudet ovat tarpeellisia käytettäessä reaaliaika-sovelluksia.

[6]

2.2.3 Fair Queuing

FQ-skedulerin avulla voidaan taata, että jokainen vuo pystyy saamaan käyttöönsä verkon resursseja reilulla tavalla. Lisäksi tällä menetelmällä voidaan estää tapauk-

sia, joissa purskeinen vuo kuluttaa kaistaa sallittua enemmän. FQ:ssa järjestelmä luokittelee paketit voihin, jonka jälkeen kullekin vuolle myönnetään yksilöllinen jono. Jonoja palvellaan paketti kerrallaan kiertävässä järjestyksessä ja tyhjät jonot hypätään yli.



Kuva 2.4: Fair Queue skeduleri

Suurin etu käytettäessä FQ-skeduleria on se, että purskeiset ja huonosti käyttäytyvät vuot eivät vaikuta muiden voiden saamaan palvelunlaatuun: kullekin vuolle myönnetään oma jono.

Mikäli vuo yrittää käyttää enemmän kaistaa kuin mitä sille on jaettu, näkyy vaikutus ainoastaan kyseisen jonon sisällä eikä sillä ole vaikutusta muiden samaan ulostuloon kytkettyjen jonojen suorituskykyyn.

Valmistajien FQ-toteutukset ovat ohjelmistopohjaisia, joten FQ:ta voidaan käyttää vain hitaammissa liitynnöissä verkon reunalaitteilla. FQ on suunniteltu siten, että jokainen vuo saa täsmälleen saman verran kaistaa. Se ei ole tarkoitettu tukemaan useita voita joilla on erilaiset kaistavaatimukset. FQ tarjoaa saman määrän kaistaa kullekin vuolle vain siinä tapauksessa, että kaikki jonoissa olevat paketit ovat samankokoisia. Vuot jotka sisältävät enemmän suuria paketteja saavat suuremman määrän kaistaa käyttöönsä kuin jonot joissa on pienempiä paketteja. Pakettien saapumisjärjestyksellä on merkitystä FQ:ta käytettäessä: mikäli paketti saapuu tyhjään jonoon juuri kun skeduleri on käsitellyt jonon, joutuu saapunut paketti odottamaan kunnes skeduleri on käynyt läpi muut jonot. FQ ei tarjoa reaaliaikapalveluja tukevaa mekanismia.

Luokkiin perustuvassa FQ-skedulerissa ulostuloportti jaetaan muutamaan palveluluokkaan. Kullekin palveluluokalle varataan tietty prosenttimäärä ulostuloportin kaistasta. Palveluluokkien sisällä kukin vuo saa saman määrän kaistaa käyttöön-

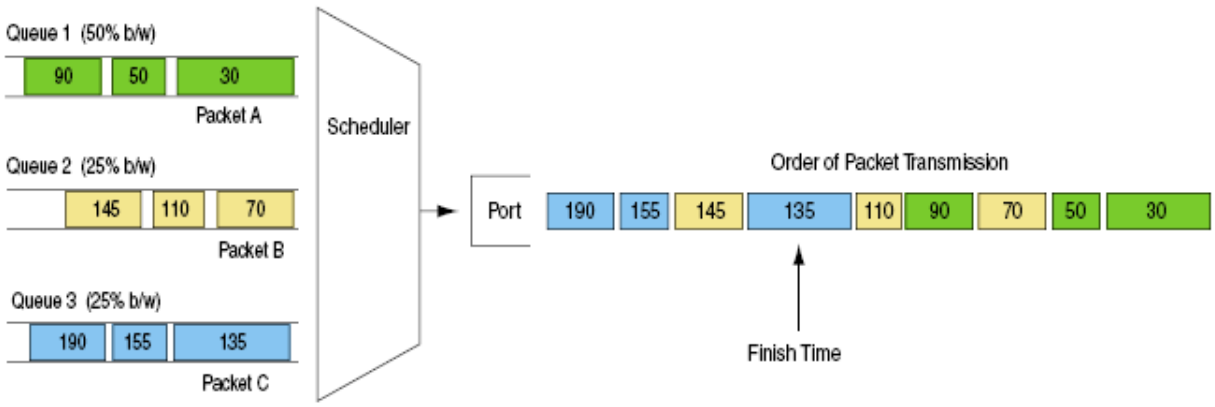
sä vuorollaan. [6]

2.2.4 Weighted Fair Queuing

WFQ-skeduleri on paranneltu versio Round Robin-skedulerista. WFQ tukee voita joilla on erilaiset kaistavaatimukset antamalla jonoille painokertoimet. Painokertoimen avulla jonolle voidaan antaa käyttöön tietyn verran ulosmenoportin kaistaa. Tämän lisäksi WFQ tukee muuttuvan mittaisia paketteja siten, että suurempia paketteja sisältävät vuot eivät varaa suurempaa kaistaa käyttöönsä kuin pienempiä paketteja sisältävät vuot. Toisaalta muuttuvanmittaisten pakettien reilu kohtelu lisää skedulointialgoritmin laskennallista monimutkaisuutta.

WFQ tukee kaistan reilua jakoa muuttuvan mittaisille paketeille arvioimalla ja jäljittelemällä **GPS**-järjestelmää. GPS on teoreettinen skeduleri jota ei ole mahdollista toteuttaa käytännössä.

WFQ luokittelee paketit yksi kerrallaan ja laittaa ne jonoon. Skeduleri laskee jokaiselle paketille virtuaalisen lopetusajan. Kun WFQ-skeduleri käsittelee jonoja, se valitsee ulosmenoportille lähetettäväksi paketiksi sen jolla on aikaisin (pienin) lopetusaika.



Kuva 2.5: Weighted Fair Queuing skeduleri

WFQ-skeduleria käytettäessä voidaan saavuttaa kaksi pääasiallista hyötyä:

- WFQ suojaa jokaista palveluluokkaa takaamalla minimitason ulosmenoportin kaistanleveydelle riippumatta muiden palveluluokkien käyttäytymisestä.
- Lisäksi WFQ takaa, että verkon reunalaitteilla ulostuloportin kaistaa jaetaan reilulla tavalla jokaiselle palveluluokalle rajoitetulla viiveellä. Näin ollen voidaan määrittää rajat yhteyden päästä-päähän mitatulle viiveelle. WFQ-skedu-

leri täytyy olla käytössä yhteyden kaikissa kytkimissä, jotta viiverajat voidaan taata.

WFQ tuo mukanaan ainakin seuraavia rajoituksia:

- Valmistajan WFQ-toteutukset ovat ohjelmistopohjaisia, joten menetelmää voidaan soveltaa vain verkon reunalaitteiden hitaammissa liitynnöissä.
- Huonosti käyttäytyvä vuo saattaa vaikuttaa suorituskykyä alentavasti muihin saman palveluluokan sisällä oleviin voihin.
- WFQ toteuttaa monimutkaisen algoritmin, jota varten joudutaan pitämään yllä tietoa palveluluokkien määrästä, sekä pakettien saapumis- ja lähetysaikoja.
- Laskennallinen monimutkaisuus vaikuttaa negatiivisesti WFQ:n skaalautuvuuteen kun tuetaan suurta määrää palveluluokkia verkonlaitteen nopeimmissa liitynnöissä.
- Vaikkakin WFQ:n takaamat ylärajat viiveelle saattavat olla paremmat kuin monessa muussa vuorottelumenetelmässä, voivat viiverajat olla silti verrattain suuret.

Vuosien mittaan on kehitetty useita WFQ-variaatioita, joilla on yritetty hallita laskennallista monimutkaisuutta, tehokkuutta ja suorituskykyä. Näistä neljä tunnetuimpaa ovat:

- **Class-based WFQ** jakaa paketteja jonoihin. Jako perustuu käyttäjien määrittelemiin paketin luokittelukriteereihin.
- **Self-clocking Fair Queuing** on WFQ-parannus, joka yksinkertaistaa virtuaalisen lopetusajan laskemisen kompleksisuutta.
- **Worst-case Fair Weighted Queuing** on myös WFQ-parannus, joka käyttää paketeille sekä aloitus että lopetusaikoja saavuttaakseen tarkemman GPS systeemin simulaation.
- **Worst-case Fair Weighted Queuing+** on parannus WF2Q-skeduleriin. Se toteuttaa uuden virtuaaliaikafunktion, jolla saavutetaan pienempi monimutkaisuus ja suurempi tehokkuus.

WFQ-menetelmää käytetään verkon reunoilla tarjoamaan reilun kaistanjaon kaikille palveluluokille. WFQ-skeduleria voidaan käyttää luokittelemaan paketteja suhteellisen suureen määrään jonoja käyttämällä hajautusfunktiota. Hajautusfunktio lasketaan käyttämällä kohde-/lähdeosoiteparia, kohde/lähde UDP/TCP portteja tai IP-paketin **ToS**-tavua. WFQ-skeduleri voidaan myös asentaa reitittimelle siten, että järjestelmä pystyy skeduloimaan rajoitetun määrän jonoja jotka liikuttavat ryhmiteltyjä voita. Tässä asennustavassa järjestelmä käyttää QoS-vaatimuksia tai kolmea vähitenmerkitsevää IP precedence bittiä ohjaamaan paketteja jonoihin. Kolmannessa menettelyssä voidaan käyttää käyttäjän määrittelemiä pakettien luokittelusääntöjä, joilla paketteja ohjataan jonoihin. Tämä lähestymistapa sallii käyttäjän määritellä tarkasti mitä paketteja ryhmitellään mihinkin palveluluokkaan ja minkä verran kaistaa myönnetään kunkin palveluluokan käyttöön. [6]

2.2.5 Weighted Round Robin

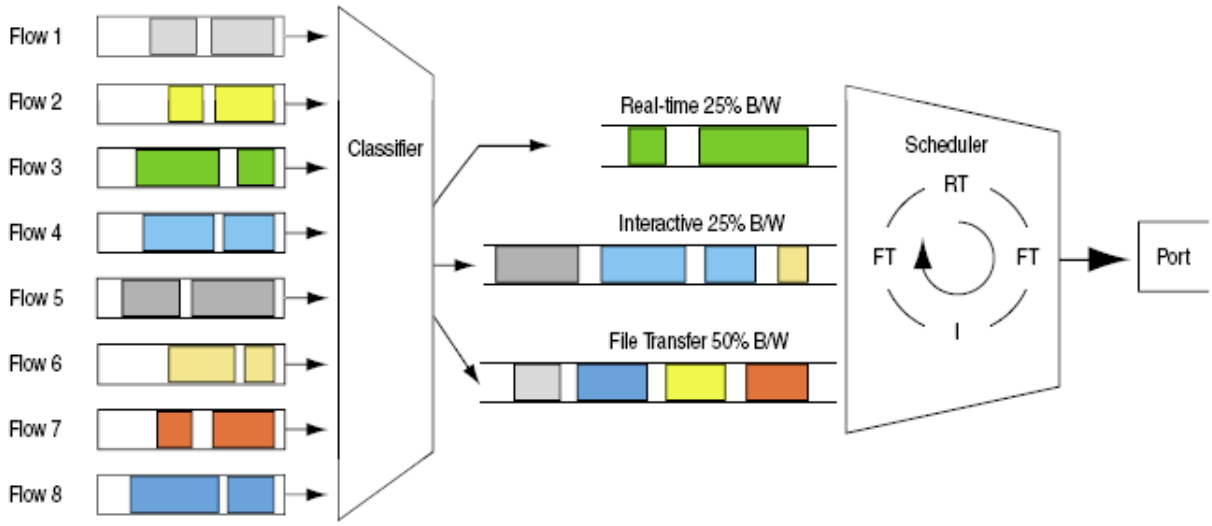
WRR-skeduleri on suunniteltu paikkaamaan FQ- ja PQ-skedulerien rajoituksia. Toisin kuin FQ-skedulerissa, WRR-skedulerissa jokaiselle jonolle voidaan antaa käyttöön erisuuruiset prosenttiosuudet ulosmenoportin kaistasta. PQ-skedulerin puutteita on korjattu varmistamalla, että matalamman prioriteetin jonojen paketit pääsevät puskuriin ja saavat käyttöönsä ulosmenoportin kaistaa. WRR-skedulerissa pudotetaan vähintään yksi paketti kustakin jonosta yhden palvelukierroksen aikana.

WRR-skedulerin toimintaperiaate on seuraava: ensin paketit luokitellaan erilaisiin palveluluokkiin (kuten reaaliaika-, interaktiivinen ja tiedonsiirtoliikenne), jonka jälkeen paketti ohjataan palveluluokalle määriteltyyn jonoon. Jokaista jonoa palvellaan kiertävässä järjestyksessä, sekä tyhjät jonot hypätään yli kuten PQ- ja FQ-skedulereissa.

WRR mahdollistaa erisuuruisten kaistaosuuksien jakamisen palveluluokille seuraavilla tavoilla:

- Suurikaistaiset jonot voivat lähettää enemmän kuin yhden paketin joka kerta kun kyseinen jono on suoritusvuorossa tai
- Jokainen jono voi lähettää vain yhden paketin suorituskierroksella, mutta suurikaistaisille jonoille voidaan antaa useampia suoritusvuoroja yhdella kierroksella.

WRR-skedulerilla jokaisen jonon käyttäytymistä voidaan kontrolloida halutun mukaiseksi säätämällä viivettä, viiveen vaihtelua, sekä pakettihäviötä. Näin voidaan säännöstellä kunkin palveluluokan käytössä olevia resursseja.



Kuva 2.6: Weighted Round Robin skeduleri

WRR-skeduleri tarjoaa seuraavia hyötyjä:

- WRR voidaan toteuttaa reitittimelle laitteistopohjaisesti, joten sitä voidaan käyttää nopeissa liitynnöissä verkon sisä- ja reunalaitteilla.
- WRR mahdollistaa ulostuloportin kaistan prosentuaalisen hallinnan kullekin palveluluokalle.
- WRR takaa, että jokainen palveluluokka saa käyttöönsä tietyn asetetun määrän kaistaa käyttöönsä.
- WRR tarjoaa tehokkaan mekanismin tukea DiffServ-luokkien käyttöä kohtuulliselle määrälle liikennevoita.
- WRR suosii ruuhkatilanteissa resurssien vähentämistä sen sijaan, että resurssien käyttö kiellettäisiin kokonaan ruuhkatilanteiden ajaksi. Resurssien käytön kieltäminen johtaa kaiken liikenteen estämiseen alemman prioriteetin palveluluokilta ja se estää samalla myös signalointiliikenteen.

WRR-skedulerin suurin rajoitus on se, että se tarjoaa tarkan prosentuaalisen kaistanjaon palveluluokille vain mikäli kaikkien jonojen kaikki paketit ovat joko yhdenmittaisia tai pakettien keskimääräinen koko on tiedossa etukäteen. Toisin sanoen jos yhden palveluluokan keskimääräinen pakettikoko on suurempi kuin yhdenkään muun luokan, saa suuremman pakettien keskikoon omaava luokka enemmän kaistaa käyttöönsä kuin mitä sen konfiguroitu ulostuloportin kaistaosuus on.

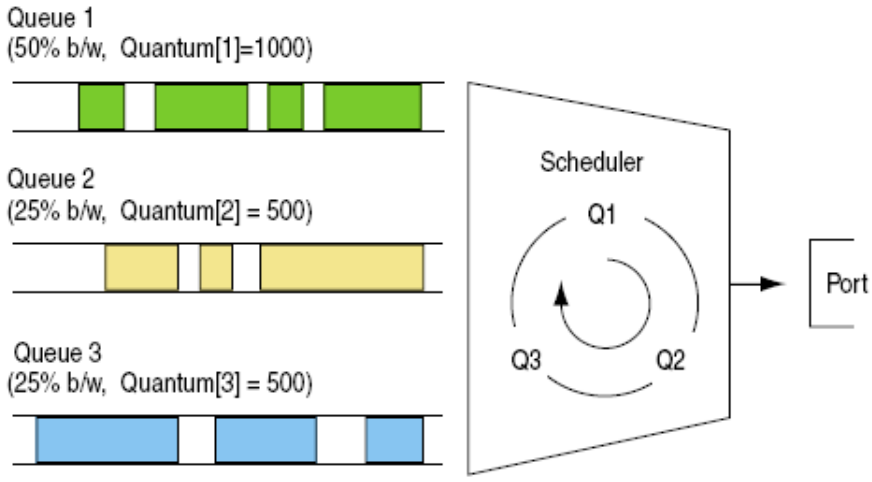
Koska WRR voidaan toteuttaa laitteistopohjaisesti, voidaan sitä käyttää sekä sisäverkossa että verkon reunoilla jakamaan reitittimen ulostuloportin kaistaa kiinteästi määriteltyjen palveluluokkien kesken. Tästä huolimatta WRR-skeduleri on kykenemätön tarjoamaan riittävän tarkkaa kaistanjakomallia muuttavanmittaisille paketeille ja se on rajoitus johon tarvitaan toimiva ratkaisu. [6]

2.2.6 Deficit Weighted Round Robin

DWRR perustuu jononhallinta-periaatteisiin, jotka on kehitetty korjaamaan WRR- ja WFQ-skedulerien heikkouksia. DWRR tukee tarkasti painotettua, reilua kaistanjakoa silloinkin kun palvellaan jonoja, jotka sisältävät erimittaisia paketteja. Toisin kuin WFQ-menetelmässä, DWRR määrittelee pakettienkäsittelijän, joka ei ole laskennallisesti monimutkainen ja näin ollen algoritmi voidaan toteuttaa laitteistopohjaisena. Tämä mahdollistaa ulosmenoportin kaistan säätämisen nopeissa liitynnöissä niin verkon sisällä kuin verkon reunoillakin.

DWRR-skedulerissa jokaiselle jonolle asetetaan joukko parametreja. Painokerroin (engl. *weight*) määrittää jonolle varattavan ulosmenoportin kaistan prosentuaalisena arvona. Niin kutsuttu hävikkilaskuri (engl. *DeficitCounter*) määrittää kokonaistavujen määrän, jonka verran jonon on sallittu lähettää joka suorituskerralla. Jos jonon kärjessä olevan paketin koko on suurempi kuin hävikkilaskurin arvo, pakettia ei voida lähettää kyseisellä suoritusvuorolla. Sen sijaan paketin koko tallennetaan hävikkilaskuriin ja jono voi käyttää seuraavalla suoritusvuorolla tallettamansa tavut ja näin ollen saa reilua kohtelua huolimatta väliin jääneestä suoritusvuorosta. Palvelun määrä (engl. *quantum*) on verrannollinen jonon painokertoimeen ja se ilmoitetaan tavuina. Hävikkilaskuria kasvatetaan palvelunmäärällä jokaisella suorituskerralla. Jos *quantum[i] = 2*quantum[x]*, silloin jono i saa kaksi kertaa enemmän kaistaa käyttöönsä kuin jono x, kun molemmat jonot ovat aktiivisina.

DWRR:n toteuttavassa algoritmissa skeduleri vierailee jokaisessa paketteja sisältävässä jonossa ja selvittää jonon ensimmäisenä olevan paketin tavujen määrän. Hävikkilaskuria kasvatetaan quantum-parametrin arvolla. Jos jonon ensimmäisenä olevan paketin koko ylittää hävikkilaskurin arvon, skeduleri siirtyy palvelemaan seuraavaa jonoa. Mikäli jonossa ensimmäisenä olevan paketin koko on pienempi tai yhtäsuuri kuin hävikkilaskurin arvo, vähennetään hävikkilaskurista paketin koko tavuissa ja paketti lähetetään ulosmenoportille. Skeduleri jatkaa jonon purkamista kunnes jonon ensimmäisen paketin koko ylittää hävikkilaskurin arvon tai jono tyhjenee. Jonon tyhjetessä sen hävikkilaskurin arvo nollataan. Tämän jälkeen skeduleri siirtyy palvelemaan paketteja sisältäviä jonoja.



Kuva 2.7: Deficit Weighted Round Robin skeduleri

DWRR mekanismin hyötyjä:

- Suojaa eri voita siten, että huonosti käyttäytyvä palveluluokka yhdessä jonossa ei vaikuta muiden samassa jonossa sijaitsevien palveluluokkien suorituskykyyn huonontavasti.
- Tarjoaa täsmälliset mekanismit ulosmenoportin kaistan jakamiseen: jokaiselle palveluluokalle voidaan määrittää prosentuaalinen osuus kaistasta myös välitettäessä vaihtelevanmittaisia paketteja.
- Kaikille palveluluokille voidaan taata tietynsuuruinen osuus ulosmenoportin kaistasta.
- Algoritmin toteutus on kohtuullisen yksinkertainen ja laskennallisen monimutkaisuuden kannalta edullinen eikä se tarvitse toimiakseen kovin monien tilatietojen ylläpitoa.

DWRR-skedulerin rajoitteet:

- Paljon käytetyissä palveluluokissa huonosti käyttäytyvä vuo voi vaikuttaa muiden samassa luokassa olevien voiden suorituskykyyn.
- DWRR ei takaa päästä-päähän viiveelle maksimiarvoa yhtä tarkasti kuin käytettäessä muita jononhallintamekanismeja.
- DWRR ei ole kaistansäädössä välttämättä yhtä täsmällinen kuin muut jononhallintamekanismit.

2.2.7 Class-Based Queuing

CBQ on skeduleri, joka tukee hierarkista linkkien jakoa ja huolehtii, että tietylle yhteydelle pystytään tarjoamaan sovittu kaista. Link-sharing ominaisuus mahdollistaa sen, että useat organisaatiot tai protokollat voivat jakaa linkin kaistaa keskenään ja levittää vapaana olevan kaistan puumaisen luokkahierarkian mukaisesti. Jokaiselle luokalle määritellään oma jono ja sen käyttöön varataan tietty määrä kaistaa. Hierarkisesti alempana oleva lapsiluokka voi lainata kaistaa hierarkiassa ylempänä olevalta vanhemmaltaan mikäli vapaana oleva kaista on käytettävissä.

2.2.8 Vuokohtainen vai luokkakohtainen jonotus?

Kun tarjotaan sovelluksille palvelunlaatua, palveluntarjoaja voi toteuttaa skedulerikokoonpanon kahdella eri tavalla. Ensimmäisessä tavassa kullekin vuolle määritellään oma erillinen jono, jolloin kyseinen vuo saadaan eristettyä huonosti käyttäytyviltä voilta. Huonosti käyttäytyvät vuot voivat lähettää dataa liian suurella nopeudella saaden aikaan puskurien täyttymisen ja lopulta jouduttaisiin pudottamaan paketteja. Vaikka tämä ratkaisu on kaikista tarkin resurssienjaon kannalta, toimii se vain mikäli voiden määrä pysyy suhteellisen pienenä. Samalla kun jonojen määrä kasvaa, joutuu reititin käyttämään paljon resursseja etsiessään paketeille oikeaa jonoa. Toinen tapa toteuttaa skedulerikokoonpano on luokitella vaatimuksiltaan samankaltaiset vuot riittävän pieneksi määräksi luokkia ja ohjata kunkin luokan paketit omaan jonoonsa. Tässä ratkaisussa luokittelijat saadaan pidettyä yksinkertaisina, jonojen etsiminen toimii nopeasti ja skeduleri ei kuormitu, koska sessioitten määrä pysyy pienenä. Myös tässä vaihtoehdossa täytyy toteuttaa jollain tavalla voiden eristäminen toisistaan. Tämän ongelman ratkaisemiseksi on kehitetty useita erilaisia ratkaisuja, kuten Stochastic Fair Queuing [12], joka arvioi tasavertaista kaistanjakoa voiden kesken. Toinen ratkaisu on käyttää vuokohtaista puskurinhallintaa järjestämällä virtuaalisia jonoja [13]. Huomioitavaa on, että palveluntarjoajan ei tarvitse käyttää samaa kokoonpanoa kaikilla reitittimillä vaan se voi käyttää esimerkiksi luokkakohtaista skedulerikokoonpanoa verkon reunoilla ja vuokohtaista kokoonpanoa verkon sisäreitittimillä. Näin voidaan löytää kompromissiratkaisu jolla on mahdollista tarjota sekä nopeaa että tarkkaa suorituskykyä. [1]

2.2.9 Arviointimallit adaptiivisessa skeduloinnissa

Arviointimalleja voidaan käyttää lisäämään kokonaistuottoa. Malli arvioi käytössä olevia parametreja ja säätää käyttäjän lähettämiä QoS-vaatimuksia. Kun malli tun-

nistaa, että tietylle vuolle/palveluluokalle riittää pienemmät resurssit, uudet QoS-vaatimukset rakennetaan ja adaptiiviset mallit ajetaan uudestaan. Tietoliikenneverkon palveluiden hinnoittelu voi riippua useista ulkoisista tekijöistä kuten vuorokauden ajasta, verkon tilasta tai tilatusta kaistanleveydestä. Hinnat voivat riippua muun muassa palveluluokasta, ruuhkasta, sekä käyttäjät voivat vaikuttaa hintaan vaihtamalla palveluluokkaa tai säätämällä yhteyden nopeutta. Myös sovellukset ja verkko voivat käydä jatkuvaa neuvottelua hinnasta, perustuen sovittuihin QoS-vaatimuksiin. Adaptiivisen verkkoresurssien jakamisen ja hinnoittelumallien integrointi toisiinsa on kohtuullisen yksinkertaista: kun hinnoittelumallit ovat laskeneet uudet hinnat voidaan ne lähettää adaptiivisille malleille samalla tavalla kuin uudet QoS-parametrit asetetaan.

2.3 Yhteenveto

Toteutettaessa suuria IP-verkkoja tarvitaan räätälöityjä jononhallintamekanismeja joita reititinvalmistajat kehittävät. Räätälöity jononhallintamekanismi voi koostua edellä mainittujen menetelmien parhaista puolista, jolloin voidaan ruuhkatilanteissa jakaa ulosmenoporttien kaistaa tarkasti eri palveluluokkien kesken. Kunkin valmistajan toteutukset pyrkivät etsimään sopivaa tasapainoa suorituskyvyn, resurssienjaon tarkkuuden ja algoritmin toteutuksen yksinkertaisuuden väliltä. Skeduleri on yksi tärkeimmistä komponenteista kun puhutaan palvelunlaadusta. Oikeanlaisen skedulerin valinta on ratkaisevaa siinä vaiheessa kun palveluntarjoaja suorittaa QoS-vaatimuksiin ja hinnoitteluun pohjautuvaa adaptiivista resurssien jakamista. Tässä luvussa esiteltiin tunnetuimmat skedulerityypit ja käytiin läpi niiden vahvuudet, heikkoudet, sekä sovelluskohteet. Esimerkiksi FQ-skedulerilla pystytään takaamaan tarkasti palvelunlaadun parametrit, mutta menetelmä on laskennallisesti raskas. RR-skedulerit ovat toteutukseltaan yksinkertaisia, mutta eivät pysty tarjoamaan tarkkoja takeita viiveen suuruudesta. Tavalliset skedulerit jakavat kaistan luokille painoarvojen perusteella ja täten ylimääräistä kaistaa ei voi jakaa esimerkiksi vain yhdelle luokalle. Hierarkinen tapa jakaa resurssit linkkien kesken tarjoaa palveluntarjoajalle tavan hallita vapaaksi jääneen kaistan käyttöä. Yksinkertaisin implementaatio tälle on **HRR**-skeduleri ja tunnetuin **CBQ**-skeduleri. Hierarkisen linkkien jakamisen huono puoli on laskennallinen taakka. Kunkin luokan keskimääräistä kaistankulutusta pitää seurata koko ajan, koska tämän perusteella reititin päättää mille luokalle resurssit jaetaan. Lisäksi asetettavia parametreja voi olla lukuisia, jolloin hallinnasta tulee monimutkaista ja tämä voi johtaa epätarkkaan kaistanjakamiseen.

3 Verkkoresurssien dynaaminen jakaminen ja hinnoittelu

Tässä kappaleessa käsitellään tarkemmin varsinaista verkkoresurssien jakamista ja palveluiden hinnoittelua. Kappaleessa esitellään tutkimuksia, joissa on etsitty ratkaisuja erilaisista näkökulmista.

3.1 Verkkoresurssien jakaminen

Skedulerista riippumatta palveluntarjoajan täytyy säätää reitittimien asetukset, jotta sovittu palvelunlaatu voidaan taata. Yksi ratkaisu on käyttää staattista konfiguraatiota. Tämä ratkaisu johtaa tehottomaan resurssien jakoon, koska verkkoa on jaettava kullekin palveluluokalle ylimitoitetusti QoS takeiden saavuttamiseksi. Parempi ratkaisu on jäljittää aktiiviset liikennevuot ja jakaa resursseja vain tarvittaessa. Tästä ylijäänyt vapaa kaista voidaan jakaa esimerkiksi tasan voiden kesken, viivekriittisille voille (minimoi viivettä ja sen vaihtelua) tai BE-voille (vähentää pakettihäviötä). Kun tiedetään aktiivisten voitten määrä, pystytään selvittämään myös käytetty kaista. Kaistanjaossa käytettäviä eri kriteereitä ovat muun muassa keskimääräinen pakettikoko, jonon koko ja pakettihäviö. Palveluntarjoaja voi määritellä eri palveluluokkia paremmalla tai rajoitetulla palvelunlaadulla. Laskutuksen suuruus riippuu suorituskyvyn määrästä ja laadusta. Vapaata kaistaa ei siis ole järkevää jakaa tavallisille *Best Effort* -käyttäjille jotka maksavat verkon käytöstä kiinteää hintaa tai käyttävät sitä ilmaiseksi. Vapaita resursseja kannattaa tarjota sellaisille asiakkaille jotka ovat valmiita maksamaan paremmasta palvelusta. Tässä kappaleessa esitellään hinnan käyttöä verkkoresurssien jaon yhtenä pääkriteerinä. Mukana on kuitenkin yksi tutkimus, jossa vapaita resursseja halutaan jakaa ruuhkatilanteiden aikana myös BE-voille. [1]

Palveluntarjoajat takaavat tietyntasoista palvelunlaatua asiakkailleen. Pakettivuolle määritellyn SLA:n mukaisesti verkon täytyy varata riittävästi resursseja, jotta vuon vaatima palvelutaso saadaan taattua. Resurssien varaus voidaan tehdä monella tapaa, riippuen verkon vapaiden resurssien kokonaismäärästä ja käyttäjien määrästä. Jossain vaiheessa käyttäjien määrä lähestyy maksimaalista raja-arvoa, jolloin resursseista alkaa olla pulaa ja käyttäjien vaatimaa palvelunlaatua ei voida toteut-

taa. Tästä syystä verkkoon täytyy toteuttaa jonkinlainen pääsynvalvonta (engl. *Call Admission Control*), joka suojaa jo hyväksyttyjen käyttäjien palvelunlaadun toteutumista uusilta yhteyksiltä. [17]

3.2 Verkkoresurssien hinnoittelu

Nykyisessä Internet-yhteyksien hinnoittelussa yleisin tapa on kiinteähintainen laskutus (engl. *flat-rate pricing*). Esimerkiksi koteihin hankittavat laajakaistaliittymät toimivat tällä hetkellä kiinteällä hinnoittelulla: käyttäjä maksaa palveluntarjoajalle tietyn kuukausimaksun joka oikeuttaa verkon rajattomaan käyttöön sovittujen liikennerajoitusten puitteissa. Shin ja Weiss luettelevat artikkelissaan [14] seuraavat kolme syytä siihen miksi kiinteästä maksusta on tullut yleisin tapa Internet-yhteyksien hinnoittelussa:

- Palveluntarjoajien ei tarvitse mitata asiakkaidensa liikennettä laskutusta varten käytettäessä kiinteää hinnoittelua.
- Asiakkaat suosivat kiinteätä laskutusta ennemmin kuin käyttöönperustuvaa laskutusta.
- Kiinteähintainen laskutus kannustaa Internetin käyttöön, koska käyttäjien ei tarvitse huolehtia yllättävistä lisäkustannuksista.

Useimmiten kiinteähintaisella laskutuksella voidaan saavuttaa suurimmat tuotot. Joka tapauksessa kiinteähintainen laskutus aiheuttaa verkon ruuhkautumista, josta koituu lisäkustannuksia palveluntarjoajille. Nykyisellä laskutustavalla käyttäjät ottavat käyttöönsä kaiken kaistan mitä yhteyden kapasiteetin puitteissa on mahdollista käyttää. Tämä johtaa ruuhkatilanteisiin ja aiemmin kappaleessa 1 mainittuun *Tragedy of Commons* -ilmiöön. Kiinteä hinnoittelu ei siis ole sopiva tuoton maksimointiin, koska se veloittaa kaikkia käyttäjiä saman verran eikä ota huomioon siirretyn datan määrää tai sitä kuinka kauan tietty palvelu on ollut käytettävissä. Nykyinen ongelma voidaan kuitenkin ratkaista, mikäli onnistutaan kehittämään oikeanlainen hinnoittelumalli, jonka myös käyttäjät ottavat vastaan. Puhelumarkkinoilla laskutus hoidetaan edelleen käyttöön perustuvilla menetelmillä, joten kyseinen hinnoittelu on ihmisille ennestään tuttu. Internetin palvelut käyttävät usein siirretyn datan määrään perustuvaa hinnoittelua aikaan pohjautuvan hinnoittelun sijasta, koska aiemmin mainittu vaikuttaa yhteyden kestoon ja nopeuteen.

Shin ja Weiss [14] esittelevät QoS-hinnoittelumallin, jossa käytetään niin kutsuttua kaksiosaista laskutusta (engl. *two-part tariff*). Mallissa hinta koostuu kiinteästä

summasta (engl. *flat pricing*), sekä lisämaksusta. Kiinteä summa oikeuttaa asiakkaan käyttämään alinta palveluluokkaa (*Best Effort*), ja lisämaksua (engl. *usage-based pricing*) peritään parempien palveluluokkien käytön perusteella. Shin ja Weiss uskovat, että kaksiosainen hinnoittelu johtaa parempaan asiakastyytyväisyyteen. Kiinteään hinnoitteluun tottuneet asiakkaat voivat käyttää pelkästään BE-palveluluokkaa, jolloin erillistä lisämaksua ei peritä. Mikäli asiakkaalla on tarvetta esimerkiksi VoIP-puheluille, on heillä mahdollisuus käyttää tähän tarkoitukseen sopivaa parempaa palveluluokkaa. Tutkimuksessa kiinnitettiin huomiota kiinteän ja kaksiosaisen hinnoittelun toteutuskustannusten väliseen eroon. Lisäkustannuksia palveluntarjoajalle aiheuttavat esimerkiksi seuraavat seikat:

- QoS-mekanismeja tukeva reititin,
- IntServ- ja/tai DiffServ-arkkitehtuurin toteutus,
- QoS-laskutusmekanismien toteutus ja
- QoS-takeiden täyttäminen SLA-sopimuksen mukaisesti huolellisen liikennesuunnittelun avulla.

Shin ja Weiss esittelevät artikkelissaan QoS-ympäristössä ajettua simulaatiota ja sen tuloksia. Siinä vertaillaan kahta palveluntarjoajaa, joista ensimmäinen käyttää kiinteätä hinnoittelua ja toinen kaksi-osaista hinnoittelua. Käyttäjien kokema palvelunlaatu on sama riippumatta palveluntarjoajasta, joten simulaatio keskittyy selvittämään kuinka kaksi-osaisella laskutuksella voidaan saada suurempi tuotto kuin kiinteällä hinnoittelulla. Tutkimuksessa kävi ilmi, että kokonaan kiinteähintaista laskutusta käyttävä palveluntarjoaja pystyy saavuttamaan suuremman tuoton asettamalla hintansa lähelle kaksi-osaista hinnoittelua käyttävän palveluntarjoajan kiinteää hintaa. Kun otetaan huomioon palveluntarjoajalle koituvat lisäkustannukset, kokonaan kiinteää hinnoittelua käyttävän palveluntarjoajan hinta pitäisi olla suurempi kuin kaksi-osaista hinnoittelua käyttävän palveluntarjoajan hinta. Kuitenkin rajattomasti palvelunlaatua tarjoavan Internet-yhteyden laskutuskulut ovat halvemmat kuin kaksiosaisessa hinnoittelussa, koska käyttöönperustuvaa hinnoittelua varten joudutaan keräämään liikennestatistiikkaa. Tästä johtuen palveluntarjoajan tulee ottaa lisäkustannukset tarkasti huomioon miettiessään palveluidensa hintoja.

Seuraavissa kappaleissa esiteltävissä tutkimuksissa keskitytään enimmäkseen resurssien jakamiseen ja tuoton maksimointiin, mutta myös hinnoittelumenetelmiä sivutaan kappaleissa 3.3, 3.8, 3.9, 3.11 esitellyissä tutkimuksissa.

3.3 Pariisin metro hinnoittelu

Artikkelissa [10] esitellään *Paris Metro Pricing* nimeä kantava hinnoittelumalli, jossa jaetaan verkko muutamaan erilliseen loogiseen kanavaan. Malli kohtelee kaikkia paketteja BE-liikenteenä, mutta kunkin kanavan käytöstä peritään erisuuruinen maksu. Käyttäjät saavat tehdä päätöksen siitä mitä kanavaa käyttävät ja maksavat sen mukaista maksua. Tutkimuksessa uskotaan, että kalliimmiksi hinnoitellut kanavat olisivat vähemmän ruuhkaantuneita kuin halvemmat kanavat. Mallin etuna on, että se osaa säätää itse itseään ja näin ollen tarjoaa lähes ilmaisen mekanismin ruuhkanhallintaan.

PMP-mallin pääasiallinen tarkoitus on pitää liikenteen hinnoittelun, loppukäyttäjän ja verkon välinen vuorovaikutus mahdollisimman yksinkertaisena. Kunkin kanavan sisäistä suorituskykyä voidaan kuitenkin tarvittaessa parantaa skedulerien ja erilaisten työkalujen avulla. Jotta toteutus saadaan pidettyä yksinkertaisena, on kanavien kapasiteettien ja hintojen pysyttävä vakiona pitkiä jaksoja, jolloin kanavien suorituskyky pysyy ennustettavana. Verkon käyttöastetta voisi kuitenkin parantaa kapasiteetin ja hintojen muuttuminen iltaisin ja viikonloppuisin. Eri kanavien suorituskyky on oltava ainakin keskimääräisesti ennustettavissa, jotta PMP saadaan toteutettua. On oletettava, että kaikki PMP-kanavat kärsivät jossain vaiheessa ruuhkista. Voidaan olettaa myös, että kalliimmiksi hinnoitellut kanavat ovat vähemmän ruuhkaisia kuin halvemmat kanavat. Kalleimmatkin kanavat voivat kuitenkin ruuhkautua, mikäli halvempien kanavien käyttäjät päättävät maksaa ruuhka-aikoina lisähintaa ja lähettävät pakettejaan kalliimmalla kanavalla. Tätä käyttäytymistä voidaan rajoittaa esimerkiksi vaihtamalla laskutusmekanismia tai kehittämällä halvimpiin kanaviin keinotekoisesti viivettä, jolloin halvimpia kanavia on mahdoton käyttää esimerkiksi videokonferenssien pitämiseen.

PMP-mallin toteuttamiseen voidaan käyttää nykyisen IPv4-protokollan prioriteettikenttää kuvaamaan yksittäiselle paketille käytettävää kanavaa. Mikäli prioriteettia ei ole määritelty, ohjataan paketti suoraan halvimmalle kanavalle. Muita kanavia käytettäessä täytyisi jollain tavalla hoitaa pakettien merkitseminen lähetyspäässä: esimerkiksi ohjelmallisesti valittaisiin mihin kanavaan sovelluksen lähettämät paketit tulee ohjata. Reitittimessä paketit täytyy ohjata oikeisiin jonoihin ja määritellä kullekin kanavalle oma jono. Myös kunkin käyttäjän lähettämien pakettien lukumäärää tulee ylläpitää. Laskeminen pystytään hoitamaan parhaiten verkon reunalaitteilla, jossa hoidetaan muitakin vaativampia toimintoja.

PMP tarjoaa yksinkertaisen hinnoittelumallin, joka ei kuitenkaan takaa täsmällistä palvelunlaatua. Toisaalta monia ostoksia tehdään nimenomaan oletuksiin pe-

rustuen: ensimmäisen luokan paikat lentokoneessa ei vielä takaa rauhallista matkaa mikäli muut matkustajat aiheuttavat häiriötä. Hyväksi havaitussa ravintolassa ei välttämättä saa kerta toisensa jälkeen samanlaatuista ruokaa. Jos ei voida taata täsmällistä palvelunlaatua, silloin täytyy yrittää palvella jokaista asiakasta reilusti. PMP on toteutukseltaan yksinkertainen ja halpa ratkaisu, joka tarjoaa ilmaisen ruuhkanhallinnan. Lisäksi kanavien sisäistä palvelunlaatua voidaan parantaa esimerkiksi WFQ-skedulerilla.

3.4 QoS-liikenteen suorituskyvyn analysointi ruuhkaanperustuvan hinnoittelun tapauksessa

Artikkeli [15] esittelee ratkaisun, jolla voidaan arvioida käyttäjän saamaa kokonaishyötyä kunkin liikenneluokan kohdalla. Ratkaisulla voidaan hallita resurssienjakoa monipalveluympäristössä säännöstelemällä verkon resurssien käyttöä. Kirjoittajien aiemmassa tutkimuksessa [16] esitellyssä ruuhkaan perustuvassa hinnoittelumallissa hinnat asetetaan siten, että käyttäjän ostaman kokonaispalvelun arvo maksimoituu. Käyttäjän ostaman palvelun kokonaisarvo lasketaan sallitun QoS-liikenteen, BE-liikenteen arvon ja BE-liikenteelle aiheutuvien viivekustannusten avulla. QoS-yhteyspyyntö estetään, mikäli verkon asettama hinta on korkeampi kuin sen tarjoama hyöty asiakkaalle. Pyyntö voidaan hylätä myös jos resursseja eli kaistaa ei ole tarpeeksi uutta yhteyttä varten. Toisin kuin QoS-liikenteen kohdalla, BE-liikennettä pudotetaan ainoastaan liiallisen ruuhkan aikana. Ratkaisu perustuu hinnoitteluun, jossa varatun resurssin hintaan vaikuttaa BE-liikenteelle aiheutetun haitan kustannukset. Mallissa uusille QoS-yhteydeksille suoritetaan hintaan perustuva pääsynvalvontatesti. Mikäli yhteys läpäisee ensimmäisen testin, suoritetaan sille seuraavaksi resursseihin perustuva pääsynvalvontatesti, jonka läpäistyään yhteys voi alkaa käyttää palvelua. Resursseihin perustuvan pääsynvalvontatestin tarkoituksena on varmistaa, että uudelle yhteydelle voidaan tarjota QoS-vaatimusten mukaiset verkkoresurssit. Testit läpäisseille QoS-yhteyksille annetaan korkeampi prioriteetti kuin BE-liikenteelle. Malli edellyttää monipalveluverkkoa, jossa BE-liikenteen laskuttamiseen käytetään kiinteähintaista laskutusta. BE-liikenteelle ei myöskään tehdä mallissa minkäänlaista pääsynvalvontaa. Malli olettaa myös, että kaikki verkon reitittimet pystyvät pitämään yllä tietoa saapuvien BE-liikenteen keskimääräisestä nopeudesta.

Käyttämällä esiteltyä arviointimallia, pystytään ymmärtämään monipalveluverkossa tapahtuvaa suorituskyvyn vaihtelua kun käytössä on ruuhkaan pohjautuva

hinnoittelumalli. Arviointimallin avulla saadut tiedot ovat hyödyllisiä suunniteltaessa ja optimoidessa monipalveluverkkoa.

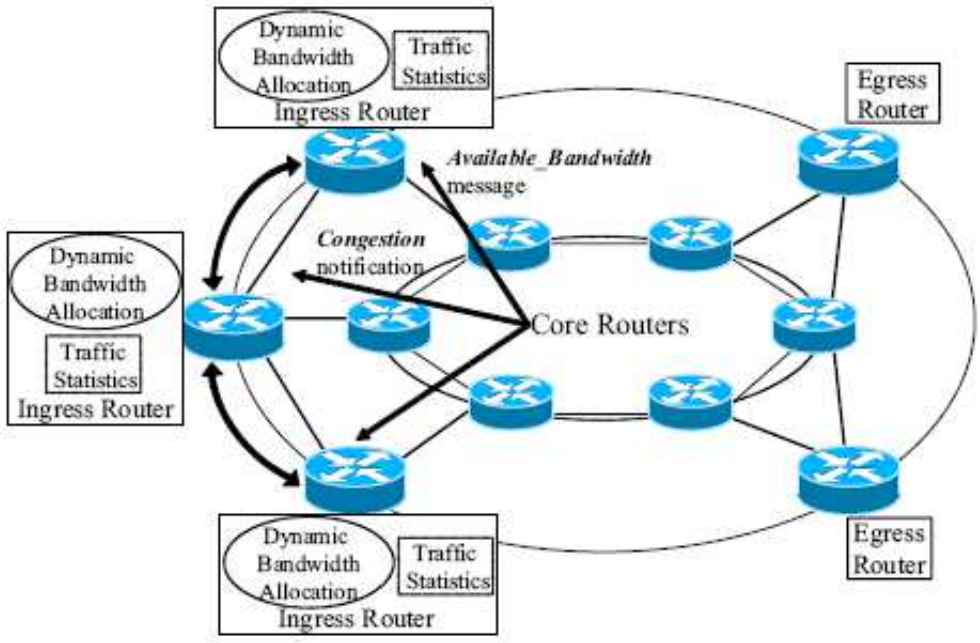
3.5 Dynaaminen resurssienvaraus kommunikaatioverkoissa

Artikkelissa [18] esitellään palvelumalli, jossa verkon asiakkaat voivat tilata käyttöönsä liittymän tietyllä peruskaistanleveydellä. Tietyin väliajoin verkko kartoittaa käyttämättömien resurssien määrän. Vapaana olevaa kaistaa voidaan jakaa asiakkaille, joiden peruskaista on jo kokonaan käytössä ja jotka ovat halukkaita maksamaan lisäkaistasta. Lisäkaistaa myydään asiakkaiden käyttöön vain lyhyiksi ajanjaksoiksi kerrallaan, jonka jälkeen asiakkaan yhteyden nopeus palautuu peruskaistanleveydelle. Asiakkaiden voille määritellään eri painoarvoja, joiden mukaan lisäkaistaa jaetaan. Painoarvoja voidaan asettaa joko staattisesti SLA-sopimuksen mukaisesti tai adaptiivisesti esimerkiksi asiakkaan hyötyfunktion avulla. Hyötyfunktio kuvaa asiakkaan mieltymykset ja tarpeet.

Palvelumallin arkkitehtuuri koostuu sisä- ja reunareitittimistä. Sisäreitittimien tehtävänä on ilmoittaa reunareitittimille kunkin linkin käyttöaste ja varoittaa linkeissä olevista ruuhkatilanteesta. Reunareitittimet keräävät mittatietoa verkkoliikenteen valvonnasta vastaavilta laitteilta ja vaihtavat keskenään tietyin väliajoin liikennestatistiikkaa.

Palvelumalliin kuuluu dynaaminen kaistanjako algoritmi, joka toteutetaan kaikkiin reunareitittimiin. Algoritmi käyttää kaistaa jakaakseen apuna reitittimien keräämää liikennestatistiikkaa. Algoritmia ajetaan jaksoittain. Sisäreitittimet tarkkailevat linkkien käyttöastetta: mikäli jonkun linkin todetaan olevan ruuhkainen, ajetaan algoritmi kyseisestä linkistä vastaavalla reitittimellä, jotta ruuhka saadaan purettua. Kaistaa jakava algoritmi koostuu kahdesta vaiheesta: ensimmäisessä vaiheessa aktiivisille yhteyksille jaetaan sovitun verran kaistaa perustuen reunareitittimien keräämään statistiikkaan. Toisessa vaiheessa ylijäänyt kaista, sekä aktiivisten ja levossa olevien yhteyksien käyttämättä jättämä kaista voidaan jakaa. Jako suoritetaan yhteyksille, jotka ovat erikseen ostaneet ylimääräistä kaistaa.

Tutkimuksessa esitelty kaistanjakoalgoritmi ottaa huomioon liikenteen tilastotiedot kasvattaakseen verkon käyttöastetta ja tuottoa. Ajettujen simulaatioiden perusteella on pystytty osoittamaan, että algoritmilla voidaan kasvattaa sekä käyttöastetta että tuottoa.



Kuva 3.1: Tutkimuksessa esitelty hajautettu arkkitehtuuri, joka tukee dynaamista kaistan jakoa.

3.6 Adaptiivisesti painotettu skedulointi korkean prioriteetin palveluille

Artikkelissa [19] esitellään adaptiivinen pakettien skedulointimekanismi, jonka avulla voidaan tarjota vähäviiveistä ja -hävikkistä liikennettä DiffServ-ympäristöissä EF-liikennevoille. Menetelmä on tarkoitettu toteutettavaksi WRR- tai WFQ-skedulereihin. Useimmiten EF-voille käytetään suurta puskuria, jotta voidaan pienentää verkon häiriöiden aiheuttamaa purskeisuutta, sekä vähentää EF-voiden hävikkiä. Liian suuri puskurikoko voi kuitenkin aiheuttaa EF-luokan paketeille pidempää jonotusviivettä, sekä suurempaa viiveen vaihtelua. Nämä aiheuttavat häiriöitä muun muassa reaaliaika-sovelluksille. Keskimääräinen jonon pituus voidaan pitää pienenä säätämällä jonojen painoarvoja dynaamisesti. Samalla pystytään takaamaan pienempi keskimääräinen jonotusviive. Menetelmässä käytetään alipäästösuodinta arvioimaan keskimääräistä jononpituutta ja samalla takaamaan alhainen viiveen vaihtelu.

Tämän adaptiivisen skedulointimenetelmän avulla voidaan lieventää verkon sisällä tapahtuvia liikennevääristymiä ilman, että viive tai viiveen vaihtelu kärsii. Tutkimuksen simulaatiotulokset osoittavat, että menetelmällä voidaan saavuttaa alhainen hävikki, alhainen viive ja alhainen viiveen vaihtelu korkean prioriteerin pal-

veluluokille.

3.7 CBQ-perustainen resurssienvarausmekanismi DiffServ-reitittimille

Artikkelissa [17] keskitytään säätämään verkon sisäreitittimien asetuksia DiffServ verkossa. Siinä esitellään menetelmä DRAM (engl. *Dynamic Resource Allocation Mechanism*), jonka avulla voidaan dynaamisesti säätää skedulerin painoarvoja. Painoarvot määräävät miten reitittimen ulostuloportin vapaana oleva kaista jaetaan DiffServ-luokkien kesken. Mekanismina DRAM takaa sen että AF- ja BE-luokat eivät häiritse toistensa liikennettä. Lisäksi menetelmällä voidaan taata, että ylimääräinen kaista jota ei ole jaettu korkean prioriteetin luokalle, jaetaan tasapuolisesti muiden luokkien kesken. Menetelmä säätää painoarvot automaattisesti, eikä se vaadi ollenkaan signalointeja.

DRAM tekee seuraavat olettamukset:

- Sisäverkossa on riittävästi resursseja, jotta verkon reunoilla merkittyä korkean prioriteetin liikennettä voidaan tukea.
- Verkon resurssit eivät ole riittävät tukemaan verkon reunoilta tulevaa kokonaisliikennettä.
- Uudet SLA-sopimukset hyväksytään verkon reunoilla ja sisäreitittimet säätävät dynaamisesti parametrinsa, jotta merkittyä liikennettä voidaan hillitä ja verkon resurssit voidaan jakaa tehokkaasti

DRAM mittaa korkean prioriteetin pakettien saapumisnopeutta ja asettaa CBQ-puskurin painoarvot siten, että korkean prioriteetin liikennettä hillitään (estetään ottamasta kaikkea ylimääräistä kaistaa) ja jäljelle jäänyt ylimääräinen kaista jaetaan matalan prioriteetin pakettien ja BE-pakettien kesken. Ylimääräisen kaistan jakoon annetaan kaksi esimerkkisääntöä:

- *Fair division.* Ylimääräinen kaista jaetaan luokkien kesken käyttäen luokille samoja painokertoimia
- *Weighted division.* Ylimääräinen kaista jaetaan luokkien kesken käyttäen priorisoituja painoarvoja

Ruuhkatilanteessa DRAM pudottaa ensin alemman prioriteetin paketteja ja mikäli ne loppuvat, ryhdytään pudottamaan korkean prioriteetin paketteja. Tutkimuksessa osoitetaan, että DRAM ja CBQ-skeduleri yhdessä käytettyinä johtavat parempaan kokonaissuorituskykyyn. CBQ kykenee tarjoamaan pienemmän viiveen korkean prioriteetin jonoille. Toisaalta CBQ aiheuttaa viiveen vaihtelua, joten se ei ole paras vaihtoehto reaali-aika sovelluksille. SLA-profiilien parametrit toteutuvat paremmin ja verkon resurssit käytetään tarkemmin yhdistämällä DRAM ja WFQ. Mekanismina DRAM käy sovitun kaistanleveyden takaamiseen ja sen avulla pystytään myös valvomaan SLA-sopimusten toteutumista. Lisäksi DRAM on tehokas työkalu verkon resurssien käyttöasteen parantamiseen.

3.8 Tuottoa maksimoiva adaptiivinen skedulointimenetelmä

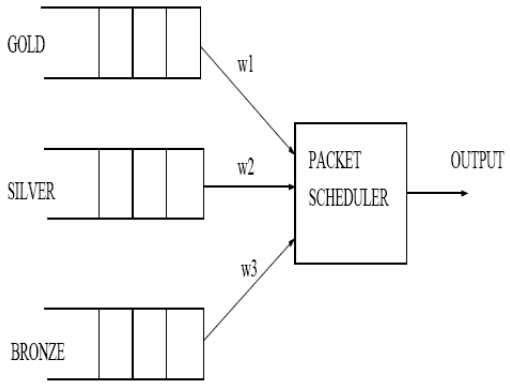
Artikkeleissa [20], [21], [22], [23], [24], [25] tutkitaan kirjoittajien kehittämää adaptiivista skedulointialgoritmia, joka yhdistää hinnoittelun ja verkkoresurssien dynaamisen jakamisen. Menetelmän avulla pystytään toteuttamaan ja takaamaan eri voiden QoS-vaatimukset ja samalla maksimoimaan palveluntarjoajan saaman kokonaistuoton. Tämä tapahtuu säätämällä reitittimellä olevan skedulerin jonojen painoarvoja. Menetelmässä käytetään kolmea palveluluokkaa (*kulta*, *hopea* ja *pronssi*), joiden QoS-vaatimukset on kuvattu taulukossa 3.1.

Taulukko 3.1: Liikenneluokkien QoS-vaatimukset

Liikenneluokka	**kaista**	**päästä-päähän viive**	**viiveen vaihtelu**	**pakettihäviö**
Kulta	x	x	x	x
Hopea	x	x		
Pronssi	x			
Best Effort				

Palveluntarjoajat laskuttavat asiakkaitaan hinnoittelufunktion avulla. Hinnoittelufunktioon vaikuttaa valittu hinnoittelumenetelmä. Tämän adaptiivisen mallin yhteydessä palveluntarjoaja voi käyttää esimerkiksi käyttöön perustuvaa hinnoittelua. Hinnoittelufunktio määritellään jokaiselle palveluluokalle ja määritelty kaista jaetaan luokkaan kuuluvien käyttäjien kesken. Hinnoittelufunktio ei riipu käyttäjien määrästä.

Toisin kuin useat muut hinnoittelu- ja resurssienjakomenetelmät, tämä adaptiivinen skedulointialgoritmi ei tarvitse toimiakseen ylimääräisiä parametreja, kuten esimerkiksi tietoja verkon asiakkaiden käyttäytymisestä. Mallin toteuttava algoritmi tarvitsee parametrikseen ainoastaan reitittimen jonoissa olevien yhteyksien määrän (käyttäjät/jono). Malli laskee skedulerin painoarvot dynaamisesti, jolloin optimaalisia painoarvoja ei tarvitse etsiä manuaalisesti. Skedulerin painoarvojen päivittämisproseduuri ei myöskään ole laskennallisesti erityisen raskas. Yksinkertaisuutensa vuoksi algoritmia voidaan käyttää hyvin epävakaissa ympäristöissä. Huomattava etu muihin menetelmiin nähden on se, että painoarvot voidaan päivittää paikallisesti kullakin reitittimellä, jolloin adaptiivista skedulointimenetelmää ei ole pakko toteuttaa verkon jokaiseen reitittimeen. Luonnollisesti verkon suorituskyky ja käyttöaste on sitä parempi mitä kattavammin mekanismia käytetään.



Kuva 3.2: Liikenteen luokittelu ulostulopuskureissa

Algoritmi toteutetaan DiffServ-arkkitehtuurin reunareitittimille, joissa suoritetaan myös suurin osa muista adaptiivisista toiminnoista. Reunareitittimet ylläpitävät vuokohtaista tietoa, jonka avulla voidaan toteuttaa myös luokittelua ja liikenteen käsittelyä. Näin sisäverkon reitittimet voidaan pitää toiminnoiltaan yksinkertaisina ja niitä ei kuormiteta ylimääräisillä adaptiivisilla operaatioilla, jotka hidastavat pakettien edelleenlähetystä. Tämä tukee myös DiffServ-tekniikoiden alkuperäistä ajatusta: reunareitittimet suorittavat kehittyneemmät toiminnot, kun taas sisäreitittimet keskittyvät yksinkertaisesti välittämään paketteja eteenpäin. Tässä adaptiivisessa ratkaisussa, adaptiivisten reunareitittimien rooli on hallita DiffServ-domainiin ohjattavan liikenteen määrää. Ylläpitämällä tietoja aktiivisten voitten määrästä ja niiden QoS-parametreista, adaptiiviset reunareitittimet voivat jakaa ulostulokaistaa optimaalisesti eri liikenneluokkien kesken. Tämä ratkaisu ei myöskään sulje pois muiden adaptiivisten ratkaisujen toteuttamista sisäreitittimille mikäli halutaan saavuttaa vielä täsmällisempi resurssien jako ja resurssien käyttöaste. Adap-

tiivisen skedulointimenetelmän toimintaa tutkitaan lisää teariaosuudessa ajettavissa simulaatioissa [kts. luku 4].

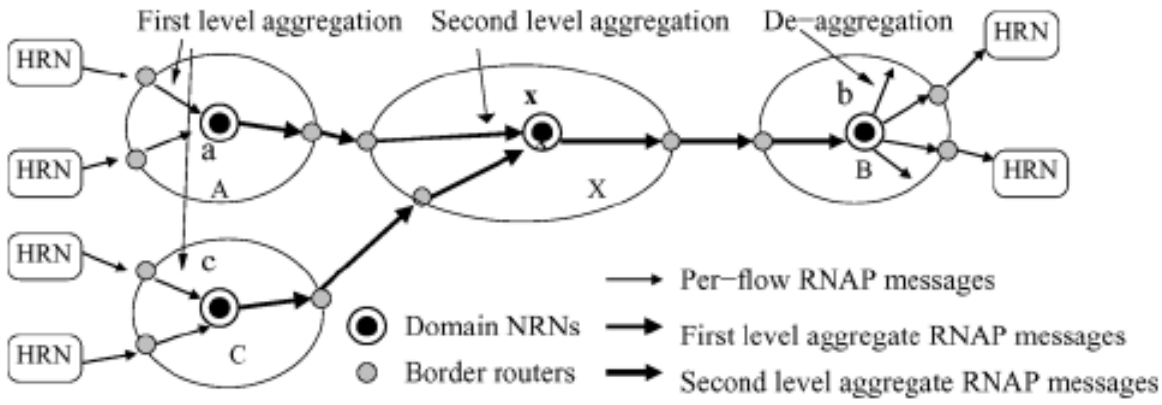
Artikkelissa [25] on käytetty edellä mainittua lähestymistapaa adaptiivisissa malleissa WFQ-, WRR- ja DDR-skedulereille. Samassa artikkelissa on myös simulaatioiden avulla todistettu, että adaptiiviset mallit voidaan ottaa helposti käyttöön DiffServ ja IntServ QoS-ympäristöissä (yhdessä RSVP-protokollan kanssa). Artikkelissa [21] esitellyistä tuloksista käy ilmi, että algoritmi on laskennallisesti edullinen kun painoarvojen päivittäminen ja pääsynvalvonta suoritetaan yhteyskerroksella. Simulaatiot osoittavat, että tuottokäyrät pysyvät positiivisina ja enimmäisviiveet pystytään takaamaan. Artikkelissa ajetuista simulaatioista käy myös ilmi, että malli jakaa verkkoresursseja tasapuolisesti käyttäjien kesken. Käytettäessä kiinteää hinnoittelua, kerätty kokonaistuotto kasvaa kun DiffServ-parametreille on löydetty optimaaliset arvot.

3.9 Verkkoresurssien hinnoittelu adaptiivisille sovelluksille

Artikkelissa [26] esitellään hinnoittelu- ja resurssienjakoratkaisu DiffServ-ympäristöön. Mallissa hinnoittelualgoritmi on integroitu ympäristöön, jossa hinnat ja käytetyt palvelut voidaan neuvotella dynaamisesti. Dynaaminen hinnoittelu tehdään palvelunlaatuun, käyttöön ja ruuhkiin perustuen. Näin pystytään käyttämään verkkoa tehokkaammin ja sovellukset voivat mukauttaa palvelupyyntönsä verkon sen hetkiseen tilaan sopiviksi. Hintojen ja palveluiden neuvotteluun käytetään **RNAP**-protokollaa (engl. *Resource Negotiation and Pricing*) ja arkkitehtuuria [27]. Käyttäjä voi valita RNAP-protokollan yli tarjottavia verkkopalveluita, joille on määritelty erilaisia QoS-ominaisuuksia ja neuvotella aiemmin sovittuja palvelupyyntöjä. Protokollan avulla myös verkko voi muokata dynaamisesti palveluiden hintoja ja välittää viimeisimmät hinnat käyttäjille.

Tutkimuksessa keskitytään kahteen tiedonsiirron määrään perustuvaan hinnoitteluun: kiinteähintaiseen, jossa määritetään kiinteä yksikköhinta (engl. *fixed unit volume price*) ja ruuhkanaikaisiin hintoihin perustuvaan hinnoitteluun, jossa kiinteässä yksikköhinnassa on lisänä ruuhkatilanteeseen mukautuva elementti. Käyttäjän puolella olevan **HRN**-elementin (engl. *Host Resource Negotiator*) ja verkon **NRN**-elementin (engl. *Network Resource Negotiator*) neuvoteltua, käyttäjän sovellus saa käyttöönsä verkkoresurssin. Eri palveluluokkien kiinteät hinnat ovat käyttäjien tiedossa. NRN tarjoaa jaksottaisesti HRN-elementille päivitetyt ruuhkahinnat. Tämän tiedon ja sen hetkisen sovelluksen vaatimusten perusteella HRN määrittelee optimaalisen lähetysnopeuden ja palveluparametrin kullekin sovellukselle. Tämän jäl-

keen HRN neuvottelee uudelleen aiemmin sovitut palvelusopimukset NRN-elementin kanssa.



Kuva 3.3: RNAP-protokollan toimintaa

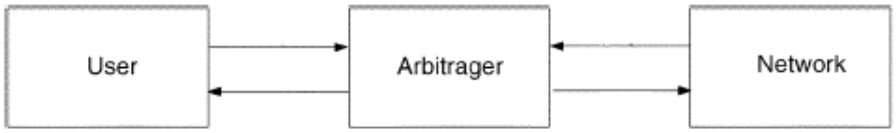
NRN ylläpitää verkkoalueen paikallisia tilatietoja muun muassa laskutusta varten, tekee pääsynvalvontaan liittyvät päätökset ja määrittelee palveluiden hinnat. NRN ylläpitää myös resurssitaulua, josta ilmenee resurssien saatavuus ja käytössä jo olevat resurssit, kuten kaista ja puskuritila. Lisäksi NRN laskee paikallisen hinnan jokaiselle reitittimelle. Jotta vuota voidaan laskuttaa, täytyy sisäreitittimien ylläpitää vuokohtaista tilatietoa neuvottelun aikana lähetetyn liikenteen määrästä. Tämä tieto lähetetään tietyin väliajoin NRN-elementille, jotta laskutus voidaan suorittaa.

Artikkelissa esiteltyjen simulaatiotulosten mukaan ruuhkaan sopeutuva hinnoittelumalli yhdessä käyttäjän lähetysnopeuden dynaamisen säätämisen kanssa tarjoavat hyvät työkalut ruuhkanhallintaan. Myös palvelunluokan suorituskykyvaatimukset täyttyvät jopa silloin kun verkko on raskaasti kuormitettu. Ruuhkiin sopeutuva, eriytetty hinnoittelu tarjoaa luonnollisen ja tasapuolisen mekanismin, jota käyttämällä sovellukset voivat mukauttaa palvelusopimuksensa verkko-olosuhteisiin sopiviksi.

3.10 Ruuhkaanperustuva verkkoresurssien hinnoittelu varauspohjaisessa QoS-arkkitehtuurissa

Artikkelissa [28] käsitellään verkkoresurssien hinnoittelua varauspohjaisessa QoS-arkkitehtuurissa. Esitelty hinnoittelumekanismi on geneerinen ja perustuu kysyntään ja tarjontaan. Se toteuttaa hajautetun resurssien jaon, joka tarjoaa taatut rajat pakettihäviölle ja päästä-päähän viiveelle. Jokaisella käyttäjällä, verkkosolmulla ja näi-

den välissä toimivalla sovittelijalla on oma hinnoitteluroolinsa. Siinä tapauksessa, että viiverajoituksia ei ole, tutkitaan kahta dynaamista hinnoittelualgoritmia: ensimmäisessä käytetään gradienttimenetelmää ja toisessa Newtonin menetelmää päivittämään hintoja. Kun taas viiverajoitukset ovat määritelty, tutkitaan aligradienttimenetelmiä, joilla voidaan saavuttaa optimaalinen resurssien jako. Artikkelissa oletetaan, että käytössä on varauspohjainen QoS-arkkitehtuuri, joka käyttää virtuaalisia piirejä reaali-aika sovelluksille. Lisäksi oletetaan, että käytetään skedulereita, jotka pystyvät varaamaan kaistaa ja puskuritilaa palveluluokille. Varauspohjaisessa QoS-arkkitehtuurissa joidenkin mekanismien täytyy päättää, mitä verkkoresursseja varataan kullekin vuolle tai palveluluokalle. Artikkelissa ehdotetaan, että päätöksen tulisi perustua sekä siihen miten sovellus arvostaa palvelunlaatua ja siihen minkälainen ruuhka verkossa on. Ensimmäinen ongelma on löytää mekanismi, jossa vaihdetaan minimaalinen määrä tietoa asiakkaan sovelluksen ja verkon välillä. Toinen ongelma on se kuinka hinnoittelua voitaisiin hyödyntää ratkaisussa. Toteutus on hajautettu käyttäjien ja verkon reitittimien välille. Artikkelissa osoitetaan, että resurssit on jaettu optimaalisesti vain jos kaikki käyttäjät ja reitittimet ovat tasapainossa (engl. *equilibrium*). Tasapainotilalla tarkoitetaan tässä yhteydessä sitä, että verkko asettaa resurssien hinnat oikeisiin arvoihin jolloin kysyntä kohtaa tarjonnan. Käytettäessä pelkästään gradienttimenetelmää ja Newtonin menetelmää, käyttäjä joutuisi valitsemaan hinnan perusteella optimaalisesti verkon resursseja jokaiselle käyttämälleen linkille ja reitittimelle. Käyttäjän tehtävä ei kuitenkaan ole tehdä resurssien varauksia tällä tasolla: käyttäjä on kiinnostunut ainoastaan siitä kuinka hyvää palvelunlaatua hän saa. Sen vuoksi kirjoittajat esittelevät myös sovittelijan, joka toimii sovelluksen ja verkon välissä (kuva 3.4).



Kuva 3.4: Kommunikaatio käyttäjän, sovittelijan ja verkon välillä

Sovittelija myy QoS-takeita käyttäjälle ja ostaa niitä verkolta tarpeellisen määrän, jotta riittävä palvelunlaatu saavutetaan. Tällä tavoin käyttäjän ei tarvitse keskittyä verkkoresursseihin, vaan hän voi keskittyä pelkästään palvelunlaatuun. Artikkelin kirjoittajien mukaan tämä niin kutsuttu kolmitasomalli varmistaa sen, että resurssien varaus on optimaalinen mikäli käyttäjät, sovittelijat ja reitittimet ovat tasapainossa. Artikkelissa esitelty menetelmä on geneerinen eikä se nojaa mihinkään tiettyyn

verkkoarkkitehtuuriin, signalointiprotokollaan tai liikenteen piirteeseen. Siitä syystä menetelmän ohessa voidaankin käyttää useita erilaisia ruuhkaanpohjautuvia hinnoittelualgoritmeja.

3.11 Palvelun laadun takaava optimaalinen hinnoittelu

Artikkeli [29] esittelee optimointimallin ratkaisuksi tuotonmaksimointiongelmaan. Esitelty malli erottelee tarpeet luokkiin palvelutyypin, lähde- ja kohdeosoitteen mukaan. Malli pyrkii maksimoimaan odotetun tuoton, saavuttamaan optimaaliset hinnat, sekä jakamaan optimaaliset ja sovitut resurssit palveluluokille. Resurssien jakaminen ja hintojen säätäminen perustuu luokkien yhteydenmuodostustarpeisiin. Tuotto maksimoidaan hakualgoritmin avulla (*huutokauppa-algoritmi*). Hinnan valintaan vaikuttaa myös verkon rajalliset resurssit ja asetetut QoS-takeet. Hakualgoritmi etsii marginaalisia kaistanleveyksiä kaikilta reitittimiltä, jotta tuotto saadaan maksimoitua. Haun täytyy testata ja säätää marginaaliset kaistat jokaiselle reitittimelle ennen kuin löytyy ratkaisu, joka jakaa koko tarjolla olevan kaistan palveluluokille kaikilla reitittimillä. Ennen kuin tätä hinnoitteluratkaisua aletaan käytännössä toteuttamaan, täytyy tehdä päätös siitä käytetäänkö keskitettyä vai hajautettua arkkitehtuuria. Kummassakin tapauksessa resursseja ja hintoja laskevan laitteen on hyödynnettävä tietoja yhteydenmuodostuspyynnöistä ennustaakseen eri palveluluokkien tulevaisuuden resurssitarvetta. Keskitetyssä arkkitehtuurissa pääreititin saa suoraan tiedot kunkin reitittimen kapasiteettitarpeista. Pääreititin laskee näiden tietojen ja ennusteiden avulla marginaaliarvot kullekin reitittimelle ja määrittelee jokaiselle palveluluokalle hinnat ja jaettavat resurssit. Tiedot päivitetään kohtuullisin väliajoin. Hajautetussa arkkitehtuurissa jokaisen reitittimen täytyy laskea hinnat ja tarvittavat resurssit palveluluokille. Jotta optimaalinen ratkaisu saavutetaan, täytyy reitittimen välittää omat tietonsa verkon muille reitittimille.

Tutkimuksessa optimaalista hinnoittelua verrataan kahteen kiinteään hinnoitteluun perustuvaan menetelmään: hyppypohjaiseen ja yhteyspohjaiseen hinnoitteluun. Numeeristen kokeilujen avulla artikkelissa näytetään, että optimaalisen hinnoittelumekanismin avulla saavutetaan suurempi tuotto kuin kummallakaan kiinteähintaisella menetelmällä. Artikkelissa esitelty malli määrittää erään tavan jolla palveluntarjoajat voivat asettaa palveluille hintoja, sekä jakaa verkon resursseja siten, että QoS-vaatimukset täyttyvät ja odotetut tuotot maksimoituvat.

3.12 Best Effort -liikenteen maksimointiin perustuva reitittimen skedulerikokoonpano

Artikkeli [30] esittelee uuden näkökulman WFQ-skedulerien parametrien asettamiseen, joka takaa palveluntarjoajalle maksimaalisen tuoton huonoimpana ruuhkahetkenä. Tutkimuksessa oletetaan, että reitittimessä on saatavilla vuokohtainen kanavointi (*multipleksointi*) ja WFQ-pohjainen skeduleri. Skeduleri priorisoi EF-liikennettä. Voille voi varata resursseja säätämällä painoarvoja. Menetelmän periaate on säätää BE-voiden painoarvoja siten, että ruuhka-aikana suositaan voita, joissa kuljetetun liikenteen määrä on mahdollisimman suuri. Kuljetetun liikenteen määrä aikayksikköä kohden on suurin kun liikennevuolle valitaan lyhyin mahdollinen reitti pisteestä A pisteeseen B. Valitsemalla BE-voiden painokertoimet oikein, on mahdollista maksimoida tuotto ja kuljetettu liikenne. Mikäli WFQ-keduleria käytetään BE-liikenteelle, voidaan sille taata minimikaista. Vaikka yleensä BE-liikenteelle ei määritellä minimikaistaa, ei määrityksen vastaisuudesta ole haittaakaan: WFQ-skeduleri jakaa ylijäävän kaistan kunhan vaan jonoissa on paketteja. Tutkimuksessa käytetään *Linear Programming* lähestymistapaa laskemaan BE-liikenteelle painokertoimia ja samalla maksimoimaan verkon kuormitusta.

Tutkimuksessa osoitetaan, että liikenteen ja tuoton määrää voidaan parantaa valitsemalla BE-voille optimaaliset painokertoimet. Painokertoimet valitaan ilman erityisiä QoS takeita, koska kyseessä on BE-liikenne. Mikäli käyttäjätyytyväisyyttä ja reiluutta halutaan parantaa, voidaan kaistaminimi kuitenkin asettaa.

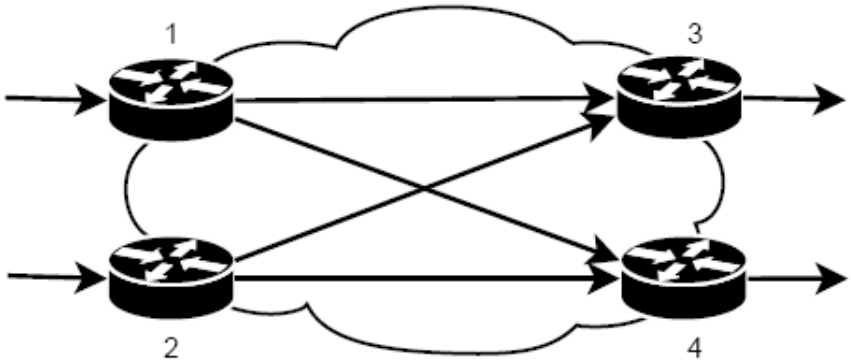
3.13 Yhteenveto

Tässä kappaleessa käsiteltiin verkkoresurssien jakamista ja verkon palveluiden hinnoittelua. Lisäksi luvussa esiteltiin menetelmiä, joiden avulla verkon resursseja voidaan jakaa tehokkaasti. Osassa menetelmistä verkon ja loppukäyttäjän välinen vuorovaikutus pyritään pitämään vähäisenä, jolloin toteutus on helpompaa. Joissain menetelmissä taas hinnat ja käytetyt palvelut neuvotellaan dynaamisesti esimerkiksi ruuhkaan perustuen. Tällöin menetelmän toteutus on huomattavasti raskaampi, koska menetelmä täytyy toteuttaa useammille verkonlaitteille ja loppukäyttäjien sovelluksiin. Verkon resursseja jaetaan säätämällä reitittimien luokkakohtaisten jonojen painoarvoja. Painoarvoja voidaan säätää muun muassa SLA-sopimuksen perusteella tai adaptiivisesti asiakkaan hetkellisten tarpeiden mukaisesti. Reitittimien keräämää verkon statistiikkaa voidaan käyttää avuksi jaettaessa resursseja. Tässä kappaleessa esitellyt menetelmät eroavat toisistaan muun muassa toteutuksen mo-

nimutkaisuuden, resurssien jaon tehokkuuden ja tarkkuuden osalta. Valitessaan sopivaa resurssienjakomenetelmää, tulee palveluntarjoajan miettiä minkä verran menetelmän toteutukseen ja ylläpitoon on varaa panostaa. Lisäksi täytyy valita tuoton maksimoinnin kannalta sopivin hinnoittelu ja päättää kuinka paljon loppukäyttäjälle voidaan tarjota vuorovaikutusta verkon resurssien varaamiseen.

4 Tutkimusskenaario: kokonaistuotto käytettäessä adaptiivista resurssienjakoa

Tässä kappaleessa tutkitaan adaptiivista resurssienjakomallia, joka on esitelty artikkelissa [22]. Tutkimus suoritetaan ajamalla simulaatioita matemaattisessa mallinnusympäristössä. Simulaatioajoissa muutetaan mallille parametreina syötettäviä luokkakohtaisia **gain**-arvoja ja katsotaan kuinka palveluntarjoajan saama kokonaistuotto muuttuu. Seuraavaksi käydään läpi työkalut, simulaatioympäristö, simulaatiotulokset ja johtopäätökset .



Kuva 4.1: Neljän reunareitittimen verkko

4.1 Työkalut

Simulaatio ajetaan *Matlab*-ohjelmistolla, joka on kehittynyt matemaattinen laskentaohjelmisto. Ohjelmalla voidaan esittää matemaattisia suureita graafisesti erilaisina käyrinä, joita tässä kappaleessa esitellään tutkimustuloksina. Simulaatiot on ajettu adaptiivisen resurssienjakomallin toiminnan käsittävillä Matlab-koodeilla joita on alunperin käytetty tutkimuksessa [22].

4.2 Ympäristö

Simulaatioympäristö koostuu neljästä reunareitittimestä (kuva 4.1), joista kaksi on niin kutsuttuja ingress-reitittimiä (operoivat voita niiden tullessa verkkoon) ja loput kaksi ovat egress-reitittimiä (operoivat voita niiden lähtiessä verkosta). Jokaisella reitittimellä on toteutettuna adaptiivinen resurssienjako algoritmi. Dataa lähetetään reitittimiltä 1 ja 2 reitittimille 3 ja 4.

Palveluluokkia on kolme: **kulta**-, **hopea**- ja **pronssi**-luokka. Kultaluokan asiakkaat maksavat saamastaan palvelusta enemmän kuin hopea- ja pronssiluokan asiakkaat. Luokkien vaatimukset on kuvattu taulukossa 3.1.

4.3 Simulaatiossa käytetty adaptiivinen malli

Seuraavaksi esitellään adaptiivinen resurssienjakomalli [22] matemaattisesta näkökulmasta. Reititin 1:n *j*:nessä jonossa on kahdentyyppisiä yhteyksiä. Ne merkitään $N_j^{1\to3}$ ja $N_j^{1\to4}$. Merkintä $N_j^{i\to p}$ tarkoittaa, että on N määrä yhteyksiä (asiakkaita) *j*:nnessä jonossa, joka kuljettaa liikennettä *i* ja *j* reitittimien läpi. Näin ollen yhteyksien kokonaismäärä reitittimessä *i* saadaan laskettua kaavalla 4.1.

$$N_{ij} = \sum_{p=1}^{n} N_j^{i\to p}, \tag{4.1}$$

jossa *n* tarkoittaa käytettävissä olevan verkon reitittimien määrää. Tässä simulaatioympäristössä reitittimiä on siis 4, joten reititinkohtaiset yhteyksien määrät saadaan selville kaavoilla 4.2, 4.3, 4.4 ja 4.5.

$$N_{1j} = N_j^{1\to3} + N_j^{1\to4}, \tag{4.2}$$

$$N_{2j} = N_j^{2\to3} + N_j^{2\to4}, \tag{4.3}$$

$$N_{3j} = N_j^{1\to3} + N_j^{2\to3}, \tag{4.4}$$

$$N_{4j} = N_j^{1\to4} + N_j^{2\to4}. \tag{4.5}$$

Käyttäjien tietyn luokan *j* käytöstä maksama hinta riippuu yhteyden nopeudesta (engl. *bit rate*). Hinta $r_j(B)$ kasvaa nopeuden *B* mukaan. Tässä adaptiivisessa resurssienjakomallissa käytetään polynomista hinnoittelufunktiota:

$$r_j(B) = r_j B^p \tag{4.6}$$

BB1-nimisellä algoritmilla (engl. *bandwidth broker*) voidaan arvioida optimaalinen skedulerikokoonpano siten, että tuotto maksimoituu ja silti kaistaa jaetaan reilulla tavalla. Tuotto lasketaan kaavalla 4.7

$$F = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} r_j \sum_{k=1}^{N_{ij}} \left(\frac{b_{ijk} w_{ij}}{\sum_{l=1}^{N_{ij}} b_{ijl}} \right)^p + \sum_{i=1}^{n} \lambda_i \left(1 - \sum_{j=1}^{m} w_{ij} \right), \tag{4.7}$$

jossa m = luokkien määrä, n = reitittimien määrä, r = hinta (engl. *gain factor*) ja b = paketin pituus. Painokertoimet w_{ij} saadaan laskettu suljetun kaavan ratkaisulla (kaavat 4.8 ja 4.9).

$$w_{ij} = \frac{r_j^{-\frac{1}{p-1}} \left[\sum_{k=1}^{N_{ij}} \left(\frac{b_{ijk}}{\sum_{l=1}^{N_{ij}} b_{ijl}} \right)^p \right]^{-\frac{1}{p-1}}}{\sum_{q=1}^{m} r_q^{-\frac{1}{p-1}} \left[\sum_{s=1}^{N_{iq}} \left(\frac{b_{iqs}}{\sum_{h=1}^{N_{iq}} b_{iqh}} \right)^p \right]^{-\frac{1}{p-1}}} \tag{4.8}$$

$$\frac{\partial^2 F}{\partial w_{ij}^2} < 0, p \in (0,1) \text{ (globaali optimaaliarvo kaavalle 4.7)} \tag{4.9}$$

4.4 Tulokset

Käytännön osuutta varten ajettiin 14 erilaista simulaatiota, jotka eroavat toisistaan annettujen luokkakohtaisten gain-parametrien osalta. Ajettujen simulaatioiden tulokset on kirjattu taulukkoihin 4.1, 4.2 ja 4.3. Simulaatioajoista tuloksena saadut käyrät kuvaavat tuoton, sekä luokkien painokertoimien ja kaistankäytön kehittymistä tietyllä ajanhetkellä. Tuotolla tarkoitetaan tässä tapauksessa kaikkien neljän simulaatioympäristöön kuuluvan reitittimen keräämää yhteistuottoa. Painokertoimien ja kaistankäytön kehitystä kuvaavat luvut ja käyrät on mitattu ainoastaan solmusta 1.

Simulaatioissa 1-10 aika kulkee yhdessä simulaatioajossa 0:sta 2000:een. Toisinsanoen se tarkoittaa, että adaptiivista mallia ajetaan jokaisella reitittimellä 2000 kertaa. Simulaatioajoissa 11-14 mallia ajetaan jokaisella reitittimellä 4000 kertaa. Ku-

vaajissa yhtenäinen viiva kertoo tuoton tietyllä ajanhetkellä, kun taas katkoviiva näyttää keskimääräisen tuoton laskettuna aikaväliltä 0-2000 (0-4000).

Simulaatioissa tutkitaan kokonaistuoton käyttäytymistä muutettaessa kullekin palveluluokalle määriteltävää **gain**-parametria. Lisäksi kussakin ajossa otetaan huomioon luokkien painokertoimien ja kaistankäytön kehitys. Simulaatioiden vertailukohdaksi on valittu simulaatioajo, jossa on käytetty gain-arvoja 500, 300, 100 (vasemmalta järjestyksessä kulta, hopea, pronssi). Näiden arvojen tuottokäyrä näkyy kuvassa 4.2. Kyseisiä arvoja käytetään vertailukohtana siitä syystä, että samoja gain-arvoja on käytetty aikaisemmassa *bandwidth broker* -algoritmia käsittelevässä tutkimuksessa [22].

Taulukko 4.1: Yhteenveto simulaatioajojen tuotoista. (*)-merkitty rivi on vertailukohta

Ajo	**gain-kulta**	**gain-hopea**	**gain-pronssi**	**maksimi-tuotto**	**keskimääräinen tuotto**
2000 kierrosta					
1 (*)	500	300	100	9090	8110
2	500	300	0	8587	7651
3	500	50	20	7024	6088
4	500	70	0	7047	6113
5	500	450	400	15780	14190
6	600	300	0	8821	8628
7	1000	300	100	15080	13200
8	1000	600	200	18180	16220
9	3000	300	100	42100	36480
10	3000	300	0	41990	36380
4000 kierrosta					
11	500	450	400	15810	14660
12	900	450	0	14620	13120
13	1000	600	200	18210	16530
14	3000	300	100	36600	42100

Taulukko 4.2: Yhteenveto simulaatioajojen luokkien painokertoimista. (*)-merkitty rivi on vertailukohta

Ajo	**gain-kulta**	**gain-hopea**	**gain-pronssi**	**paino-kulta**	**paino-hopea**	**paino-pronssi**
2000 kierrosta						
1 (*)	500	300	100	0,5345	0,3598	0,1057
2	500	300	0	0,5969	0,4029	0,0002
3	500	50	20	0,9734	0,0186	0,0080
4	500	70	0	0,9640	0,0360	0,0002
5	500	450	400	0,5490	0,2690	0,1860
6	600	300	0	0,6800	0,3190	0,0002
7	1000	300	100	0,8190	0,1390	0,0410
8	1000	600	200	0,5350	0,3600	0,1060
9	3000	300	100	0,9758	0,0186	0,0056
10	3000	300	0	0,9811	0,0187	0,0002
4000 kierrosta						
11	500	450	400	0,1726	0,2160	0,5655
12	900	450	0	0,6784	0,3215	0,0008
13	1000	600	200	0,5289	0,3597	0,1115
14	3000	300	100	0,9755	0,0187	0,0059

Taulukko 4.3: Yhteenveto simulaatioajojen luokkien kaistankäytöstä. (*)-merkitty rivi on vertailukohta

Ajo	**gain-kulta**	**gain-hopea**	**gain-pronssi**	**kaista-kulta**	**kaista-hopea**	**kaista-pronssi**
2000 kierrosta						
1 (*)	500	300	100	10,89	3,98	0,51
2	500	300	0	12,04	4,39	0,08
3	500	50	20	19,19	0,27	0,11
4	500	70	0	19,01	0,46	0,08
5	500	450	400	4,01	3,27	2,57
6	600	300	0	13,64	3,47	0,08
7	1000	300	100	16,32	1,55	0,24
8	1000	600	200	10,89	3,98	0,51
9	3000	300	100	19,23	0,28	0,10
10	3000	300	0	19,33	0,28	0,08
4000 kierrosta						
11	500	450	400	3,41	2,77	2,18
12	900	450	0	12,67	3,19	0,04
13	1000	600	200	9,96	3,61	0,44
14	3000	300	100	18,08	0,22	0,06

4.5 Simulaatiotulosten analysointi

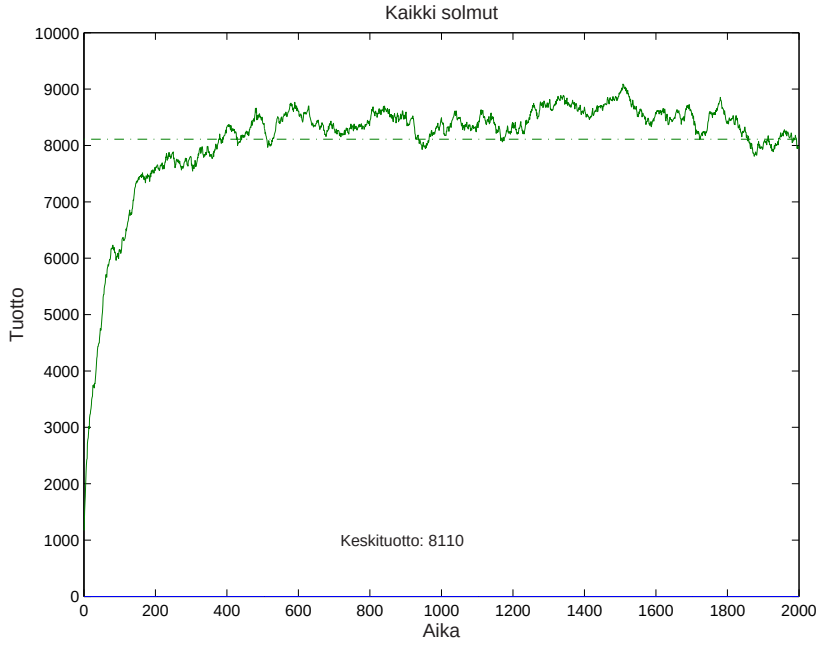
Tässä kappaleessa analysoidaan kunkin simulaatioajon tuoton käyttäytymistä tuottokäyrän ja keskimääräisen tuoton avulla. Taulukkoon 4.1 on koottu yhteenveto simulaatioajojen keskimääräisistä ja maksimaalisista tuotoista. Taulukko 4.2 kuvaa simulaatioajojen luokkakohtaiset keskimääräiset painokertoimet ja taulukko 4.3 kuvaa luokkien keskimääräiset kaistankäytöt (engl. *bitrate*).

4.5.1 Ajo 1

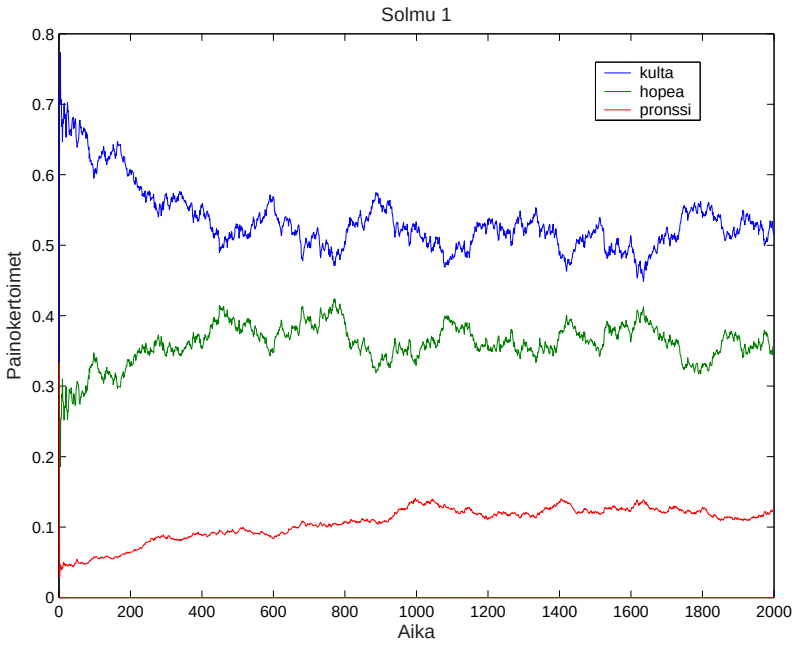
Gain-arvot: kulta 500, hopea 300, pronssi 100

Kuvaajassa 4.2 tuottokäyrä nousee rauhallisesti keskimääräisen tuoton tasolle ja tuoton kasvu rauhoittuu 400 kierroksen kohdalla. Keskimääräinen tuotto on 8110.

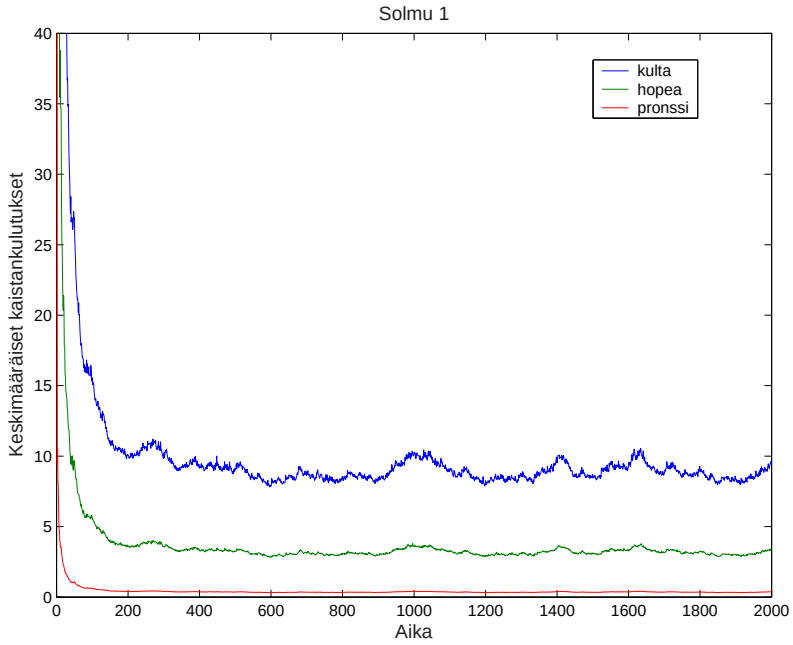
Kultaluokan keskimääräinen painokerroin on 0,5345 eli se saa jononkäsittelijän suoritusajasta noin 53 prosenttia. Hopealuokan keskimääräinen painokerroin on 0,3598 (n. 36 % suoritusajasta) ja pronssiluokan vastaava arvo on 0,1057 (n. 11 % suoritusajasta). Luokkien painokertoimet ovat siis suurin piirtein samassa suhteessa gain-arvoihin nähden (kuvaaja 4.3). Keskimääräisestä kaistankäytöstä voi todeta, että hopealuokka käyttää vajaan puolet ja pronssiluokkan kahdeskymmenesosan kultaluokan käyttämästä kaistanleveydestä (kuvaaja 4.4).



Kuva 4.2: Ajon 1 tuottokäyrä



Kuva 4.3: Ajo 1 - Luokkien painokerrointen kehitys



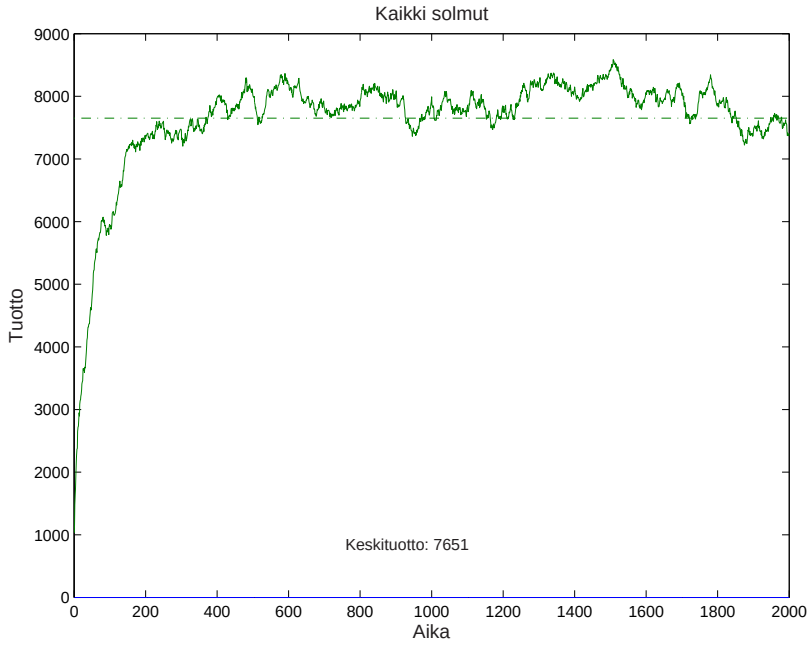
Kuva 4.4: Ajo 1 - Luokkien kaistankäytön kehitys

4.5.2 Ajo 2

Gain-arvot: kulta 500, hopea 300, pronssi 0

Kuvaajassa 4.5 nähdään ajon 2 tuottokäyrä, jossa pronssiluokan gain-arvo on asetettu nollaksi. Keskimääräinen tuotto on 459 yksikköä pienempi kuin ajossa 1. Erotus on 5.6 prosenttia ajon 1 keskimääräisestä tuotosta.

Pronssiluokan gain-arvon tiputtaminen nollaan saa myös luokan keskimääräisen painokertoimen tippumaan nollan tienoille (kuvaaja 4.6). Tästä syystä sekä kulta- ja hopealuokkien keskimääräiset suoritusajat ja kaistankäytöt kasvavat hieman suuremmiksi kun verrataan ajoon 1 (kuvaaja 4.7).



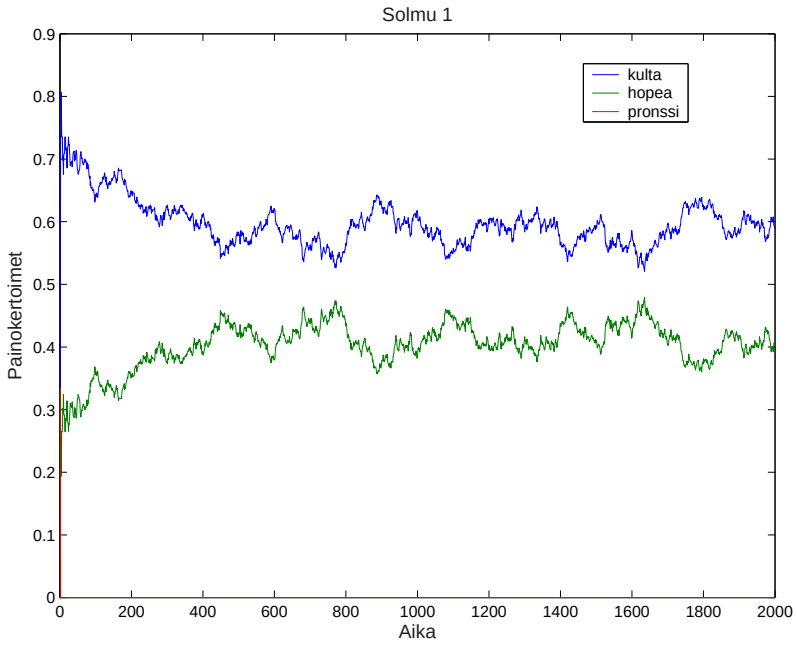
Kuva 4.5: Ajon 2 tuottokäyrä

4.5.3 Ajo 3

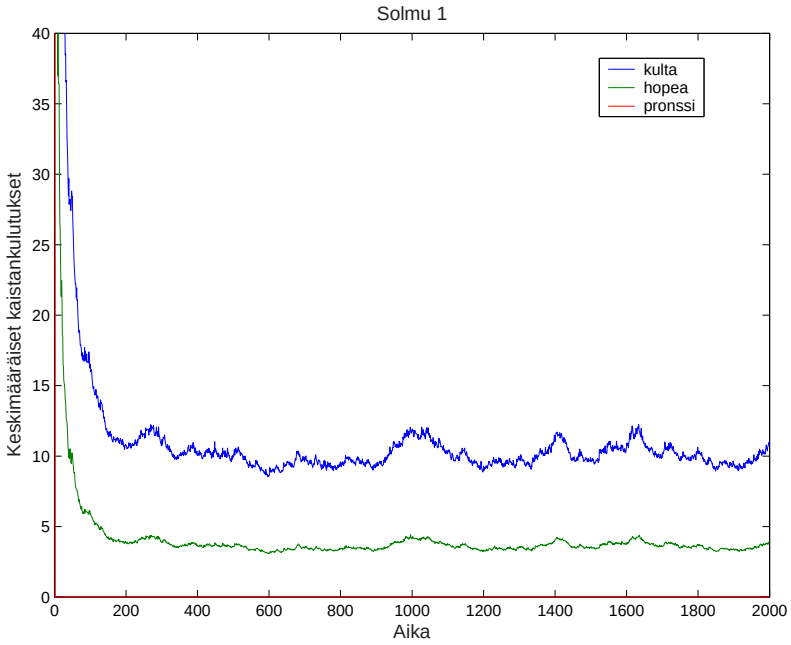
Gain-arvot: kulta 500, hopea 50, pronssi 20

Ajon 3 tuottoa kuvaava käyrä nousee keskimääräisen tuoton tasolle jo kierroksen 200 kohdalla. Tuotto vaihtelee terävästi keskimääräisen tuoton ylä- ja alapuolella.

Hopea- ja pronssiluokkien gain-arvojen huomattava pienentäminen saa aikaan niiden keskimääräisten painokertoimien romahtamisen ja vastaavasti kultaluokan saaman suoritusajan kasvamisen. Kultaluokka käyttää suurimman osan kaistasta, koska hopea- ja pronssiluokille jää vain hieman suoritusaikaa (2 % ja 1 %).



Kuva 4.6: Ajo 2 - Luokkien painokerrointen kehitys



Kuva 4.7: Ajo 2 - Luokkien kaistankäytön kehitys

4.5.4 Ajo 4

Gain-arvot: kulta 500, hopea 70, pronssi 0

Ajon 4 tuottokäyrä on lähes identtinen edellisen ajon käyrän kanssa. Ainoastaan keskimääräinen tuotto on 25 yksikköä pienempi. Pienet luokkienväliset gain-arvon muutokset eivät siis vaikuta kokonaistuottoon merkittävästi.

Tässä ajossa luokkien painokerrointen ja kaistankäytön kehitys on likimain sama kuin edellisessä ajossa: kultaluokka saa käyttöönsä suurimman osan suoritusajasta ja kaistasta.

4.5.5 Ajo 5

Gain-arvot: kulta 500, hopea 450, pronssi 400

Kuvaajassa 4.8 gain-arvot on asetettu lähemmäksi toisiaan. Tämä vaikuttaa tuottokäyrään siten että tuotto kasvaa hitaasti, mutta kasvun hidastuttua hetkellistä tuottoa kuvaa käyrä pysyy keskimääräisen tuoton yläpuolella. Pidemmällä aikavälillä voidaan olettaa, että hetkellinen tuotto on myös ajoittain keskimääräistä tuottoa pienempi.

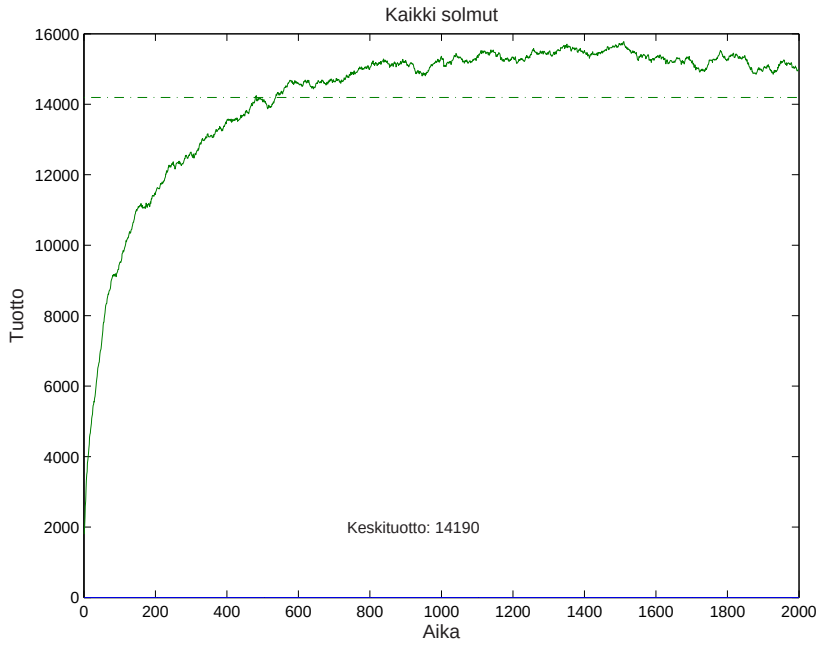
Kun gain-arvot asetetaan lähemmäksi toisiaan saa pronssiluokka enemmän suoritusaikaa ja myös sen keskimääräinen kaistankäyttö kasvaa. Kultaluokan keskimääräinen kaistankäyttö sen sijaan pienenee noin puolella verrattuna ajoon 1 (kuvaajat 4.9 ja 4.10).

4.5.6 Ajo 6

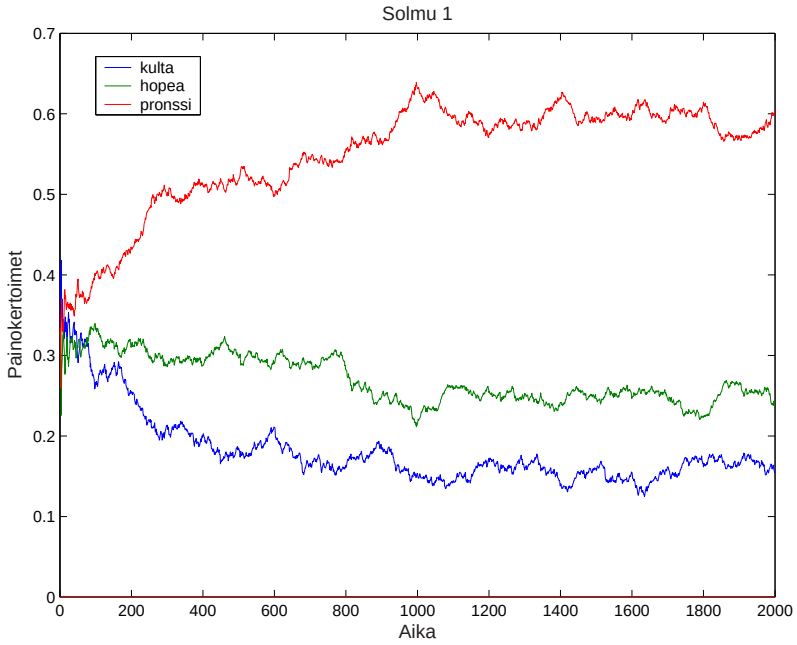
Gain-arvot: kulta 600, hopea 300, pronssi 0

Kuvan 4.11 kuvaajassa hetkellinen tuotto saavuttaa ensimmäisen kerran keskimääräisen tuoton tason hieman yli 200:n kierroksen kohdalla. Tuotto vaihtelee suurimmillaan 1500 yksiköllä. Verrattuna ajoon 1, pronssiluokan gain-arvo on siirretty kultaluokalle, jolla keskimääräistä tuottoa saadaan kasvatettua noin 500:llä yksiköllä.

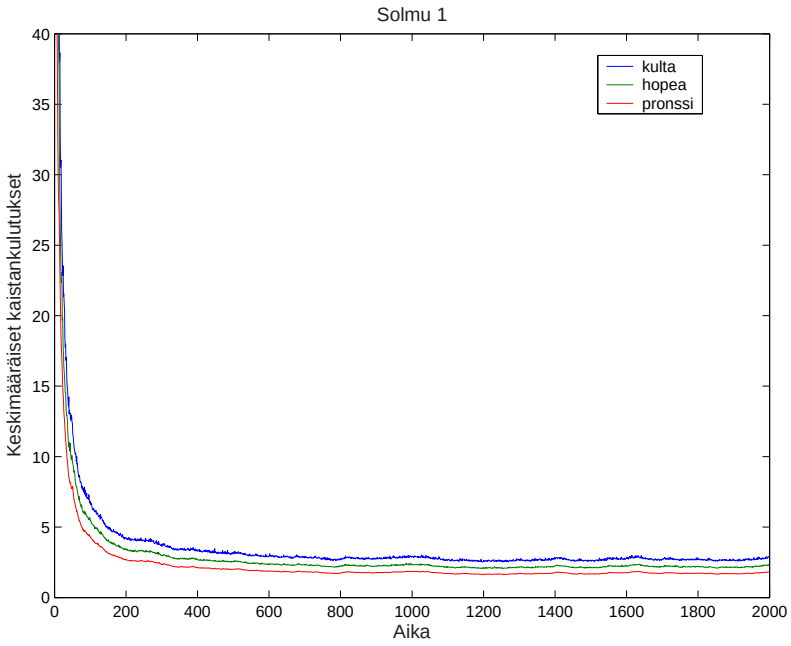
Tässä ajossa pronssiluokan keskimääräinen painokerroin on lähes nolla, jonka vuoksi myös kaistankäyttö on vähäistä. Verrattuna ajoon 1 pronssiluokan gain-arvon asettaminen nollaksi ja kultaluokan gain-arvon kasvattaminen lisäävät kultaluokan saamaa suoritusaikaa ja keskimääräistä kaistankäyttöä. Hopealuokan vastaavat arvot laskevat hieman verrattuna ajoon 1.



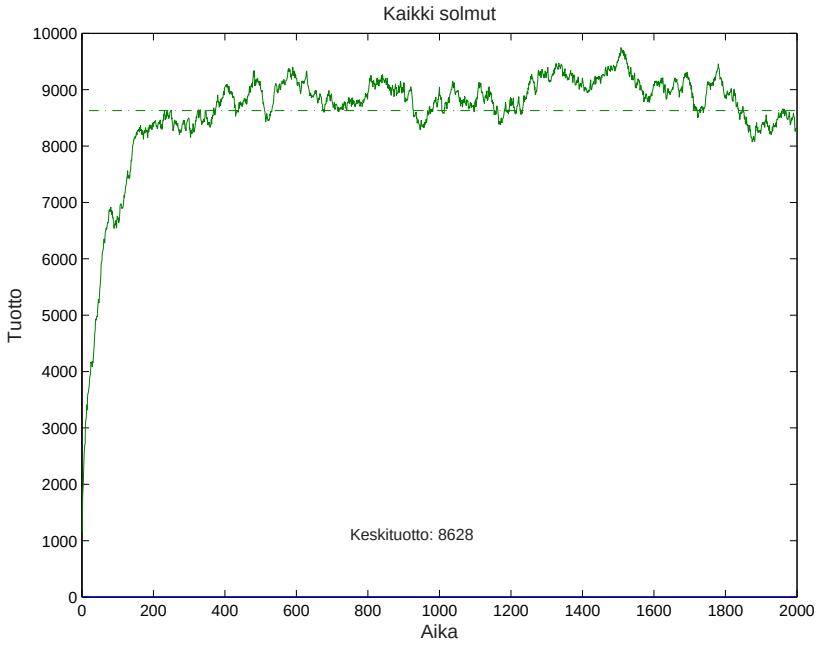
Kuva 4.8: Ajon 5 tuottokäyrä



Kuva 4.9: Ajo 5 - Luokkien painokerrointen kehitys



Kuva 4.10: Ajo 5 - Luokkien kaistankäytön kehitys



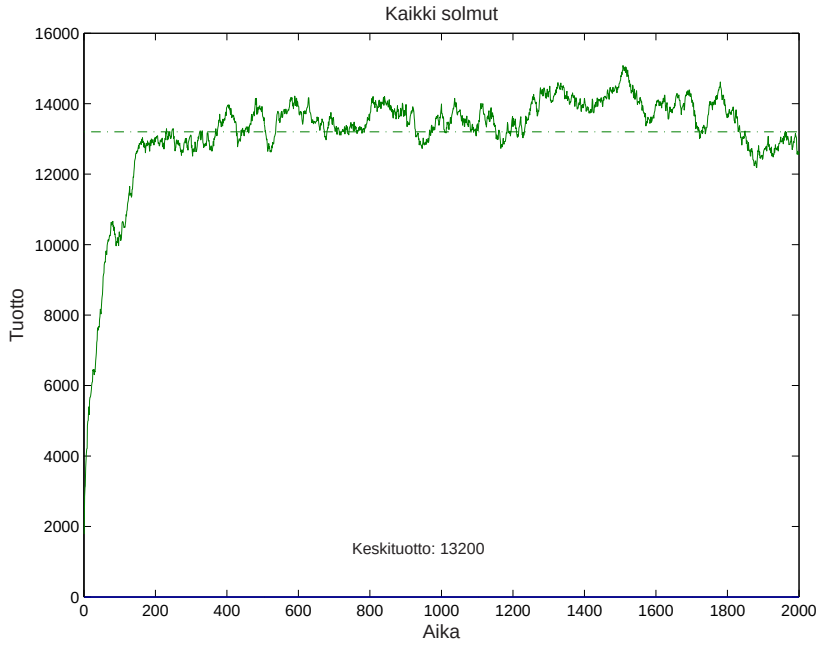
Kuva 4.11: Ajon 6 tuottokäyrä

4.5.7 Ajo 7

Gain-arvot: kulta 1000, hopea 300, pronssi 100

Simulaatioajossa 7, kultaluokan gain-arvo on tuplattu ajoon 1 nähden. Tuottokäyrä muuttuu huomattavasti jyrkemmin ja saavuttaa keskimääräisen tuoton tason ajoa 1 nopeammin. Keskimääräinen tuotto on noin 5000 yksikköä suurempi.

Tästä ajossa huomataan, että kultaluokan gain-arvon tuplaaminen kasvattaa sen suoritusaikaa huomattavasti. Kultaluokan keskimääräisten painokerrointen kasvu tarkoittaa samalla, että muiden luokkien saama suoritusaika vähenee. Ajossa 1 suoritusajat jakautuvat niin, että kultaluokka saa suoritusajasta keskimäärin 53 prosenttia, hopealuokka 36 prosenttia ja pronssiluokka 11 prosenttia. Nyt kultaluokka saa keskimäärin peräti 82 prosenttia koko suoritusajasta kun hopealuokalla jää 14 prosenttia ja pronssiluokalle ainoastaan 4 prosenttia. Ajoon 1 verrattuna hopea- ja pronssiluokkien suoritusaika ja kaistankäyttö ovat pienentyneet likimain puolella.



Kuva 4.12: Ajon 7 tuottokäyrä

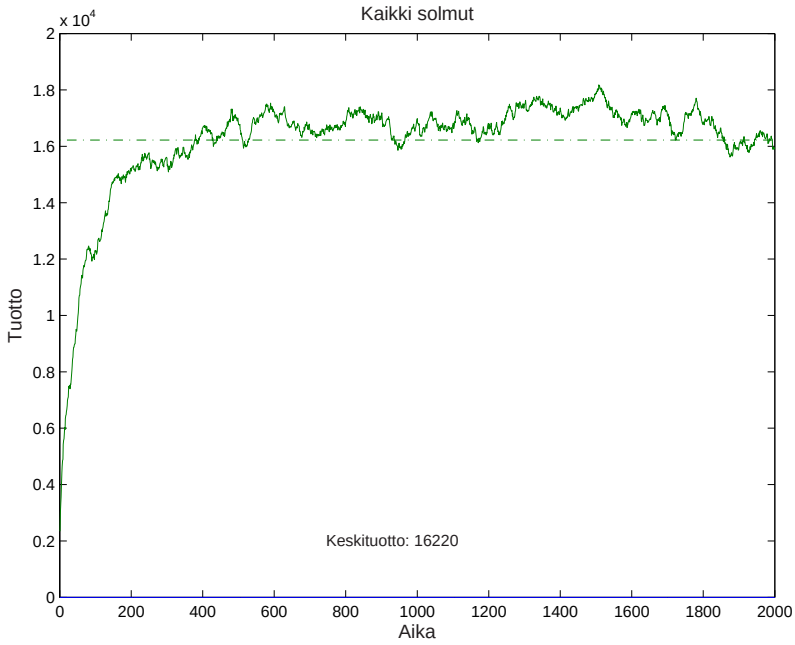
4.5.8 Ajo 8

Gain-arvot: kulta 1000, hopea 600, pronssi 200

Simulaatioajossa 8 (kuva 4.13) jokaisen luokan gain-arvot on tuplattu ajon 1 arvoihin nähden. Ajojen 1 ja 8 tuottokäyrät ovat graafisesti täsmälleen identtiset. Molemmat tuottokäyrät saavuttavat keskimääräisen tuoton tason noin 400 kierroksen

kohdalla. Ajon 8 keskimääräinen tuotto ja maksimituotto ovat tasan kaksi kertaa suuremmat kuin esimerkkitapauksessa.

Gain-arvojen tuplaus ei vaikuta luokkien keskimääräisiin painokertoimiin tai kaistankäyttöihin, vaan ne pysyvät samoina kuin ajossa 1.



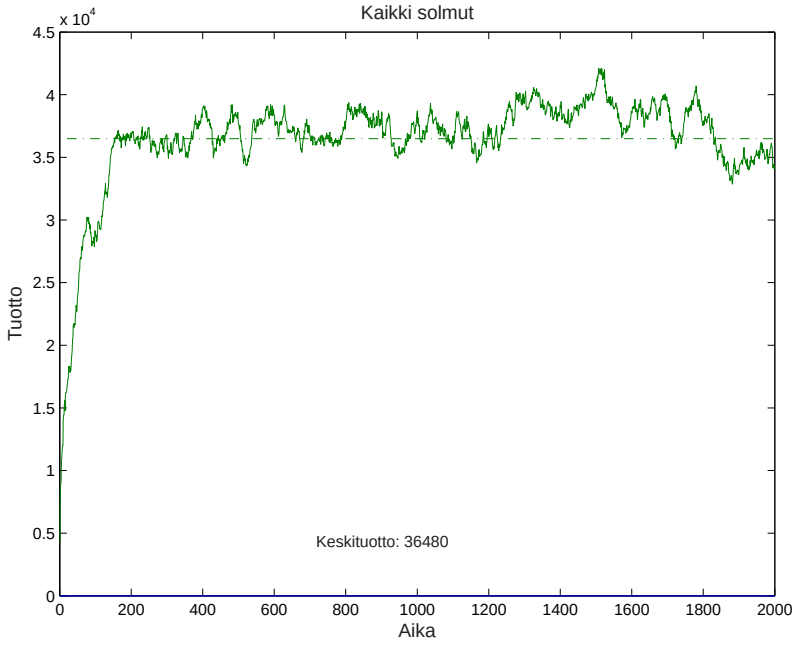
Kuva 4.13: Ajon 8 tuottokäyrä

4.5.9 Ajot 9 ja 10

Gain-arvot: kulta 3000, hopea 300, pronssi 100 – kulta 3000, hopea 300, pronssi 0

2000 kierroksen simulaatioista suurimmat keskimääräiset tuotot saavutettiin ajoissa 9 ja 10 (katso taulukko 4.1, kuvaajat 4.14 ja 4.15). Huomattavasti suurempi tuotto oli odotettavissa, koska kultaluokan gain-arvo asetettiin kuusi kertaa suuremmaksi kuin esimerkkiajossa 1. Mielenkiintoinen seikka vertaillessa ajoja 9 ja 10 on se, että pronssiluokan gain-arvon muuttaminen nollaksi ei pienennä tuottoa yhtä paljon kuin simulaatioajoissa 1 ja 2 (kuvaajat 4.2 ja 4.5). Ajojen 1 ja 2 keskimääräisten tuottojen ero on 459 yksikköä, kun taas ajojen 9 ja 10 vastaava arvo on 100 yksikköä. Tästä voi siis päätellä, että mitä suuremmaksi asettaa korkeamman prioriteetin gain-arvot sitä vähemmän pronssiluokan gain-arvo vaikuttaa kokonaistuottoon. Lisäksi ajojen 9 ja 10 käyrät saavuttavat keskimääräisen tuoton tason simulaatioajoista lyhyimmässä ajassa, 150:n kierroksen kohdalla. Myös muutokset tuottokäyrässä ovat huomattavasti jyrkempiä kuin edellisissä ajoissa.

Ajossa 9 kasvatetaan kultaluokan gain-arvoa kuusinkertaiseksi verrattuna ajoon, jolloin kultaluokka saa suurimman osan suoritusajasta ja kaistasta (kuvaajat 4.16 ja 4.18). Tuotto nelinkertaistuu verrattuna ajoon 1. Ajossa 10 tilanne on muuten sama kuin ajossa 9 paitsi pronssiluokan gain-arvo on laskettu nollaan. Tästä johtuen pronssiluokan keskimääräinen painokerroin ja suoritusaika laskevat myös nollaan, kun taas kultaluokan vastaavat arvot kasvavat hiukan (kuvaajat 4.17 ja 4.19).



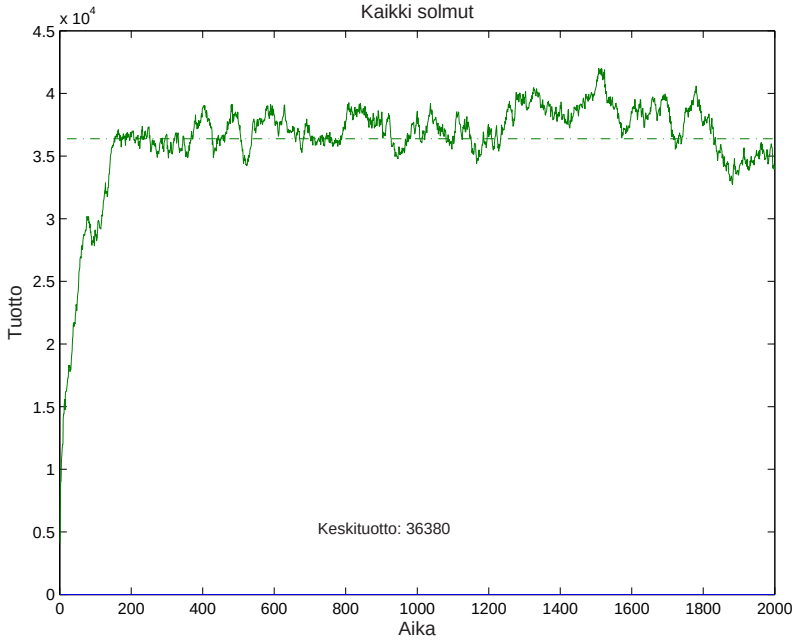
Kuva 4.14: Ajon 9 tuottokäyrä

4.5.10 Ajo 11

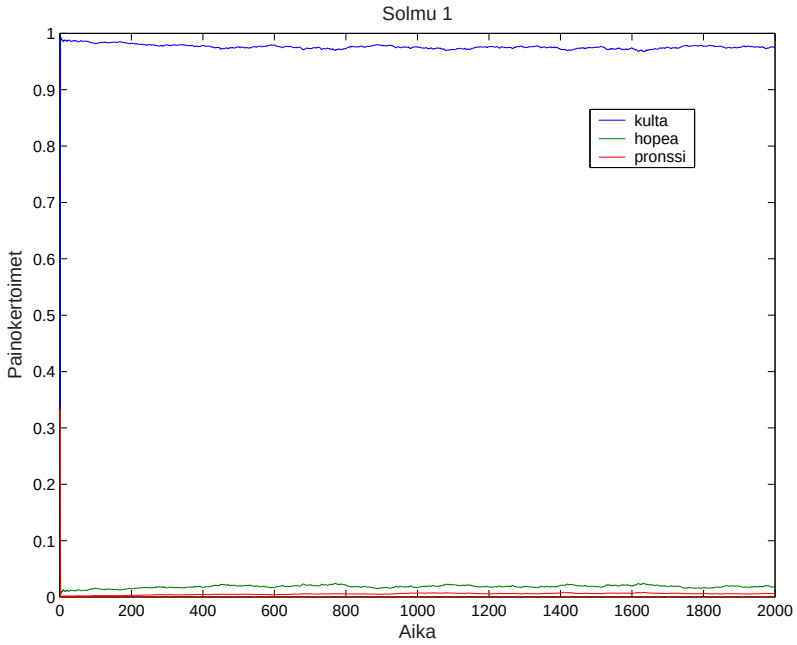
Gain-arvot: kulta 500, hopea 450, pronssi 400

Kuvaajassa 4.20 näkyy ajon 11 tuottokäyrä joka on ajettu käyttäen samoja parametreja kuin simulaatioajossa 5. Ajokierroksia on sen sijaan puolet enemmän (4000 kierrosta), jotta nähdään paremmin kuinka tuotto käyttäytyy hieman pidemmällä aikavälillä. Keskimääräinen tuotto on 470 yksikköä suurempi kuin ajossa 5. Lisäksi nähdään, että tuottokäyrä käy myös keskimääräisen tuoton alapuolella.

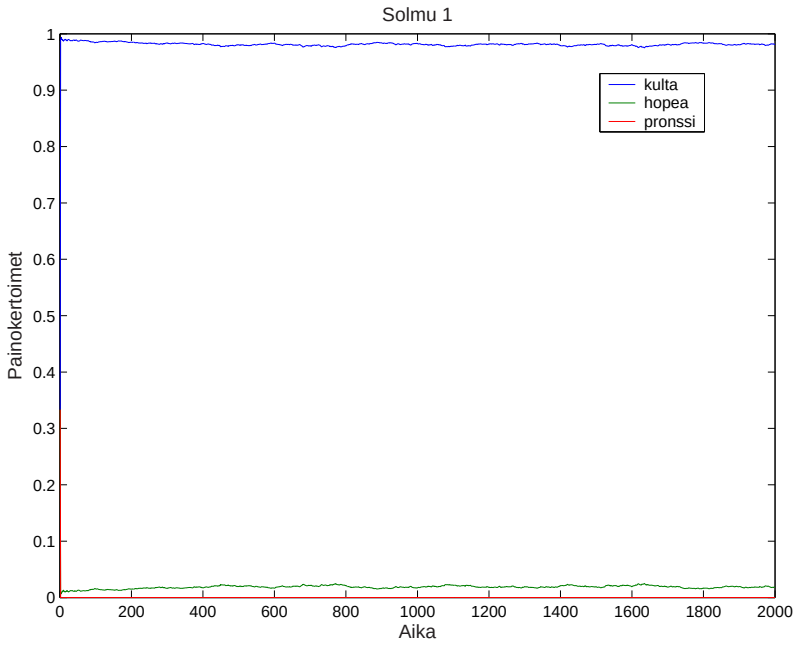
Kuvaajasta 4.21 nähdään, että vaikka pronssiluokalla on pienin gain-arvo, saa se eniten suoritusaikaa. Luokkien keskimääräiset kaistankäytöt ja suoritusajat ovat vain hieman pienempiä kuin ajossa 5, jossa on käytetty samoja gain-arvoja (kuvaaja 4.22).



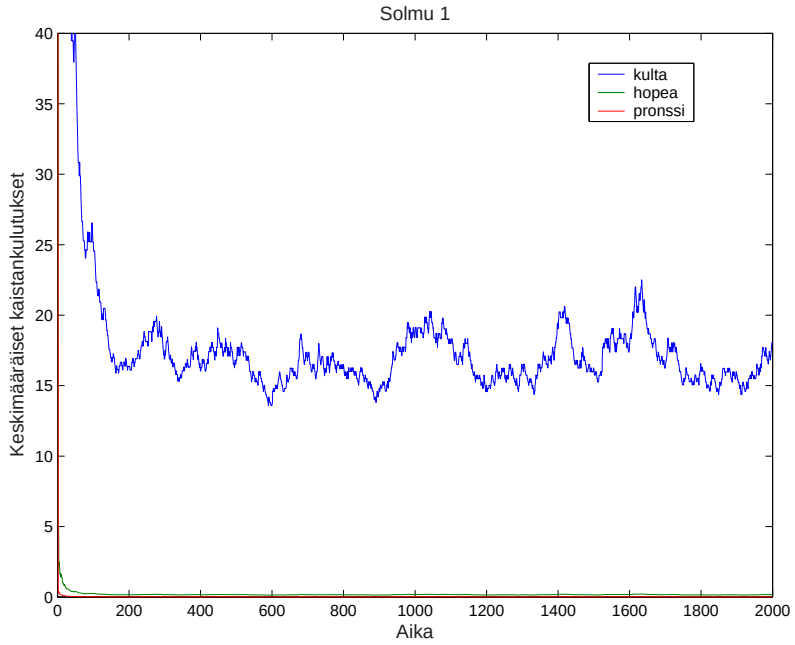
Kuva 4.15: Ajon 10 tuottokäyrä



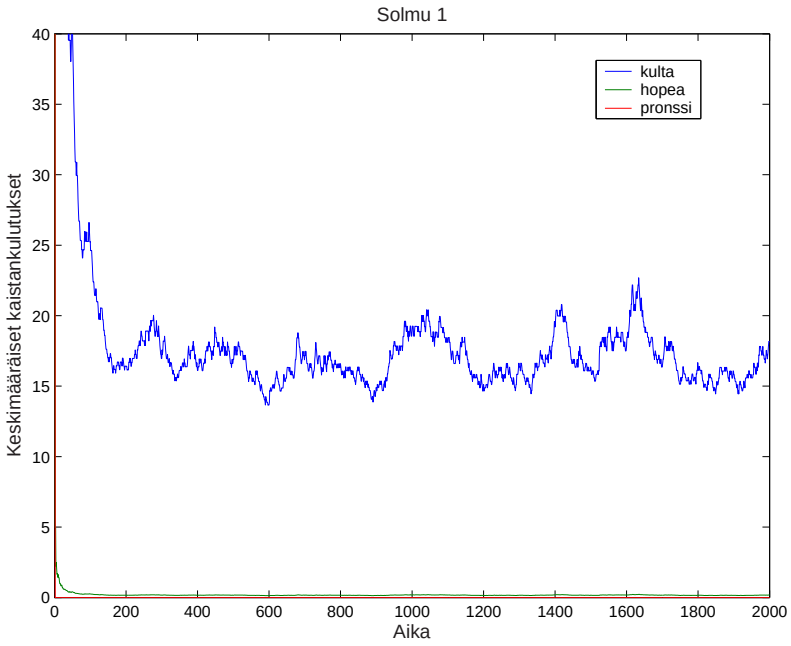
Kuva 4.16: Ajo 9 - Luokkien painokerrointen kehitys



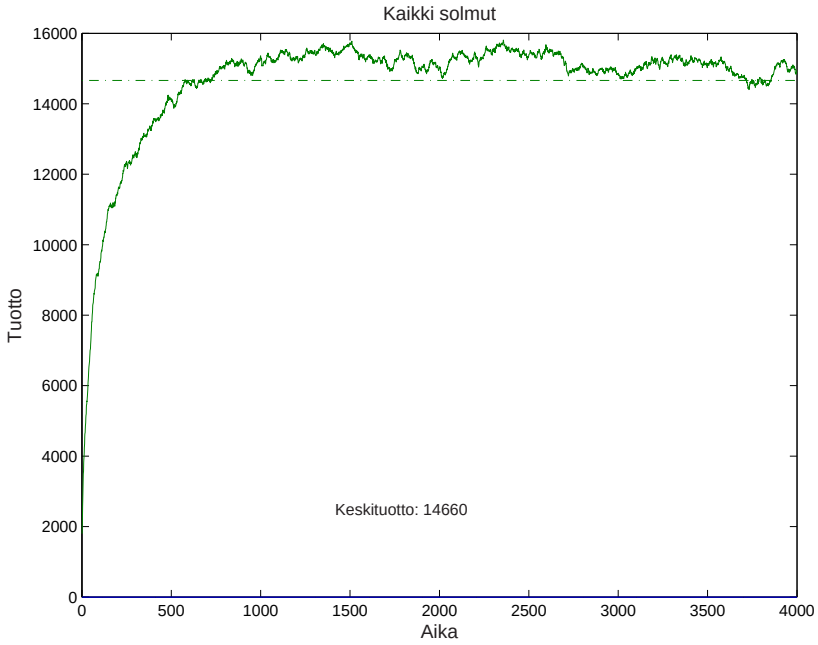
Kuva 4.17: Ajo 10 - Luokkien painokerrointen kehitys



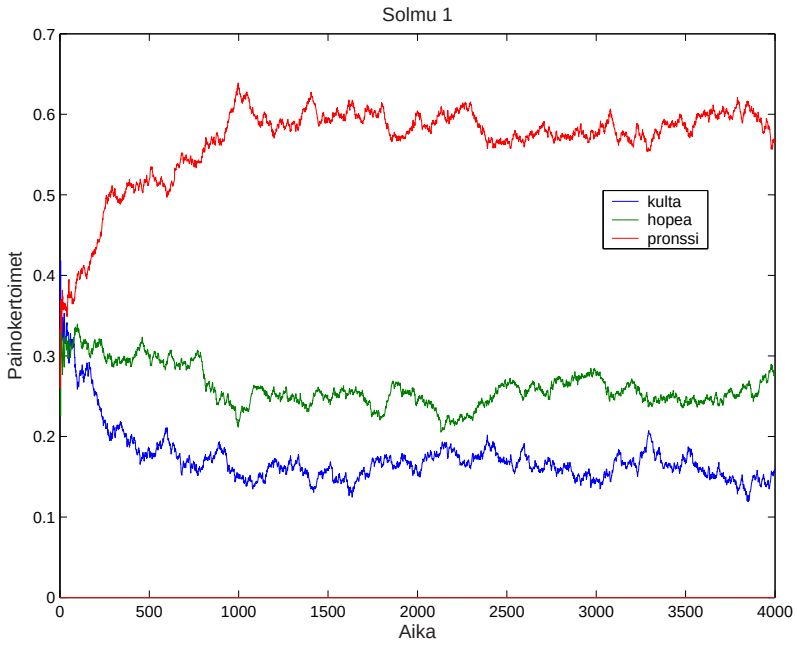
Kuva 4.18: Ajo 9 - Luokkien kaistankäytön kehitys



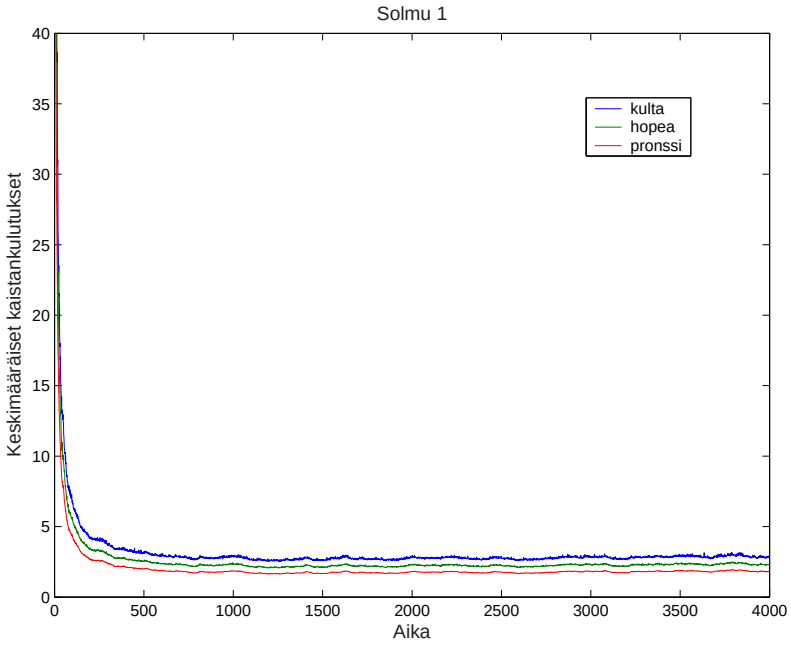
Kuva 4.19: Ajo 10 - Luokkien kaistankäytön kehitys



Kuva 4.20: Ajon 11 tuottokäyrä, 4000 kierrosta



Kuva 4.21: Ajo 11 - Luokkien painokerrointen kehitys, 4000 kierrosta



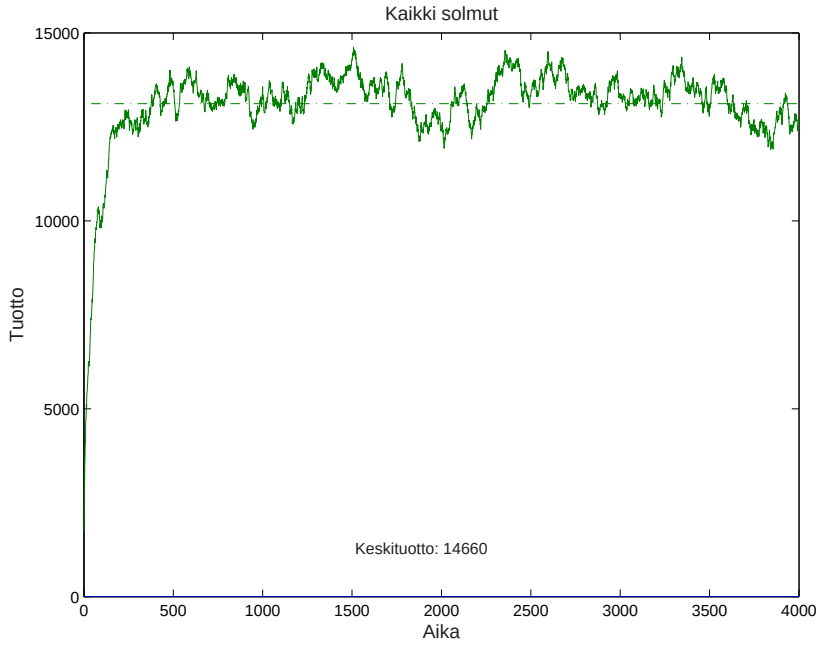
Kuva 4.22: Ajo 11 - Luokkien kaistankäytön kehitys, 4000 kierrosta

4.5.11 Ajo 12

Gain-arvot: kulta 900, hopea 450, pronssi 0

Kuvaajassa 4.23 nähdään 12. ajon tuottokäyrä, jossa edelliseen ajoon verrattuna pronssiluokan gain-arvo on siirretty kultaluokalle. Ajossa on käytetty 4000 kierrosta. Keskimääräinen tuotto laskee 1540 yksikköä verrattuna edelliseen ajoon. Lisäksi tuottokäyrä heittelee edestakaisin kun edellisessä ajossa käyrä pysytteli suurimman osan ajasta keskimääräisen tuoton yläpuolella.

Näillä gain-arvoilla kultaluokka saa suoritusaikaa puolet enemmän kuin hopealuokka ja pronssiluokka ei saa lainkaan suoritusaikaa.Kultaluokka käyttää kaistaa keskimäärin 4 kertaa enemmän kuin hopealuokka.



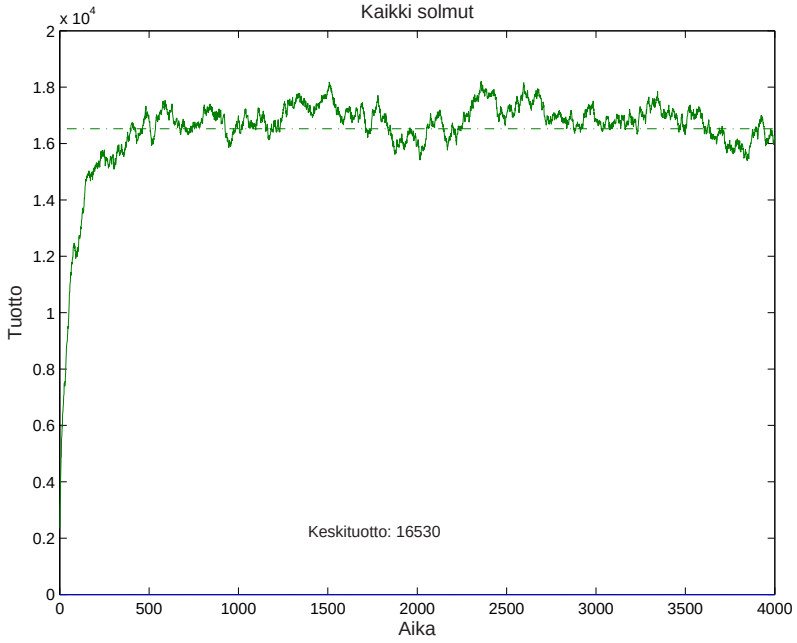
Kuva 4.23: Ajon 12 tuottokäyrä, 4000 kierrosta

4.5.12 Ajo 13

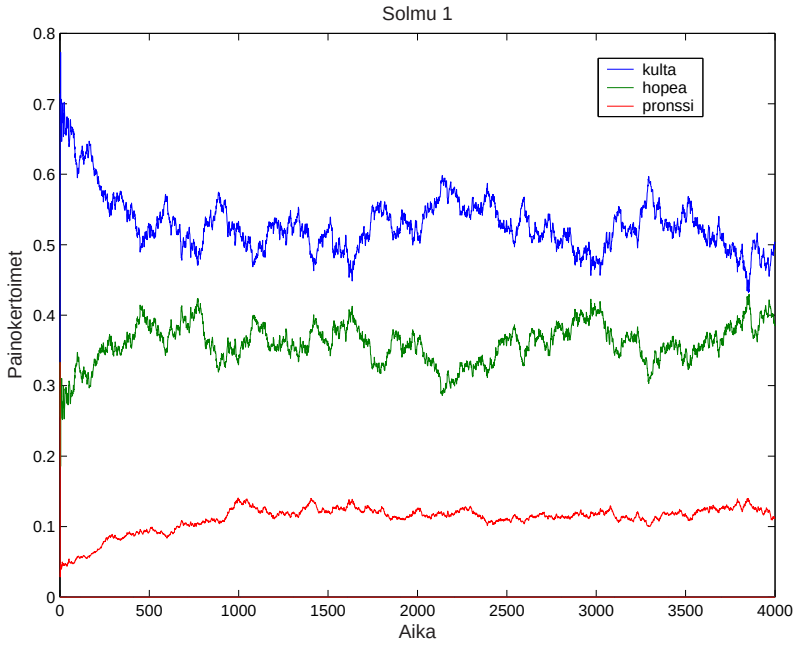
Gain-arvot: kulta 1000, hopea 600, pronssi 200

Ajon 13 (kuvaaja 4.24) tuottokäyrä on ajettu samoilla gain-arvoilla kuin ajossa 8, mutta mallia on ajettu 4000 kierrosta. Keskimääräinen tuotto on noussut 310 yksikköä verrattuna ajoon 8.

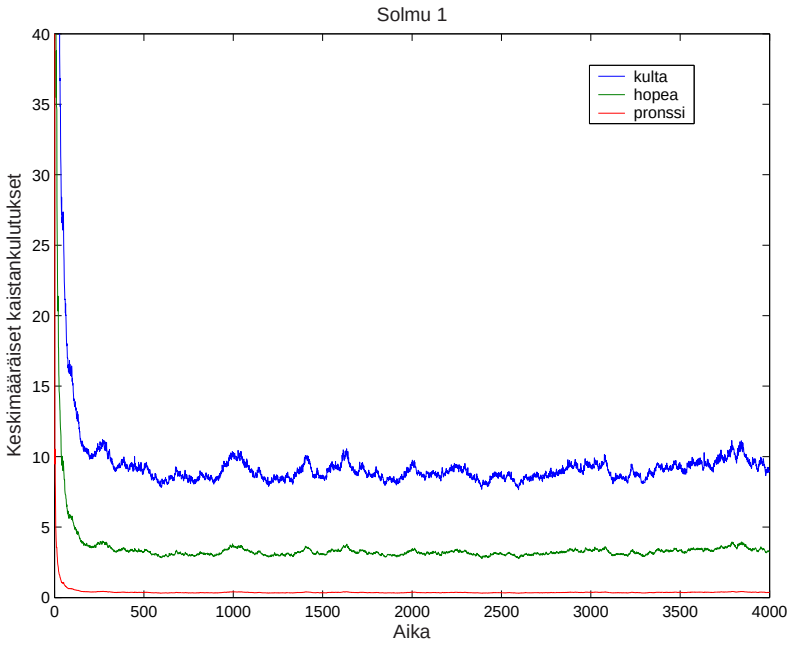
Ajoa 8 puolet pidempi ajoaika näkyy siinä, että luokkien keskimääräiset kaistankäytöt ja painokertoimet ovat laskeneet aavistuksen ajoon 8 nähden (kuvaajat 4.25 ja 4.26).



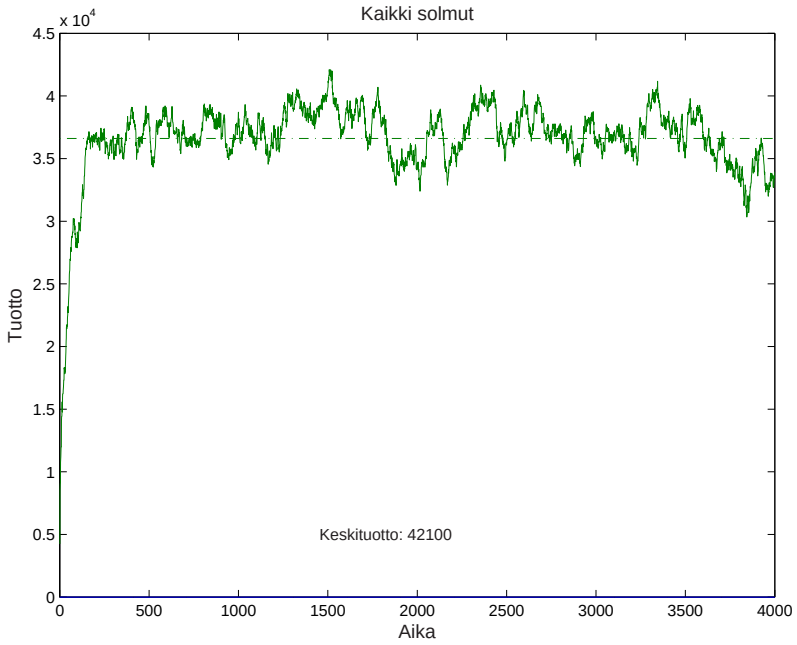
Kuva 4.24: Ajon 13 tuottokäyrä, 4000 kierrosta



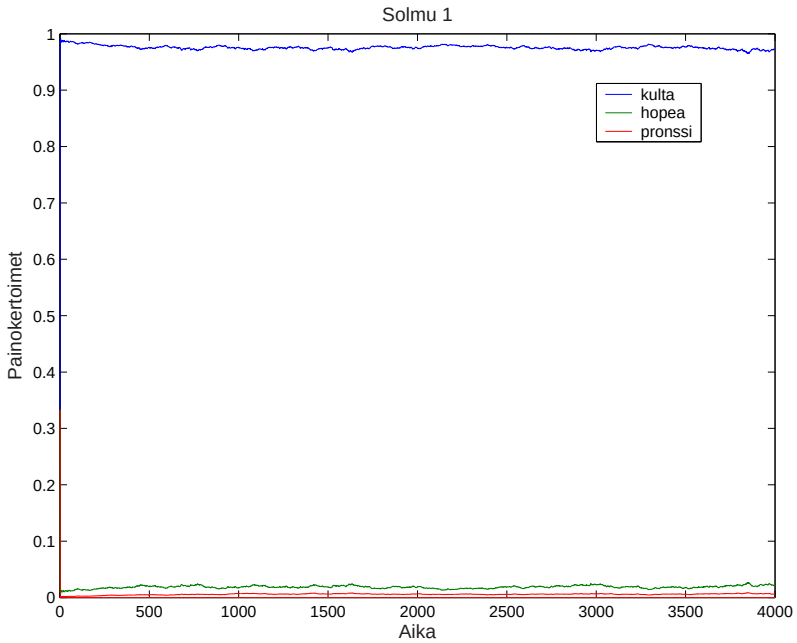
Kuva 4.25: Ajo 13 - Luokkien painokerrointen kehitys, 4000 kierrosta



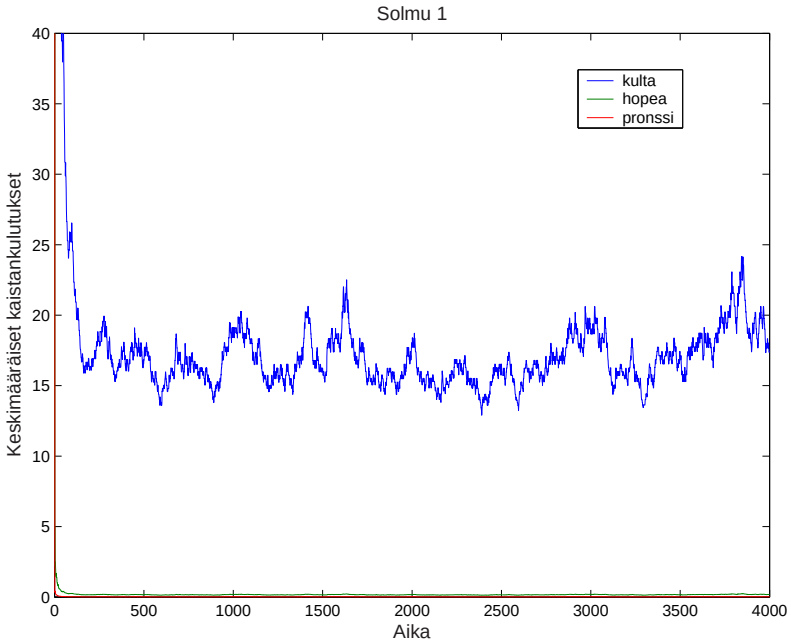
Kuva 4.26: Ajo 13 - Luokkien kaistankäytön kehitys, 4000 kierrosta



Kuva 4.27: Ajon 14 tuottokäyrä, 4000 kierrosta



Kuva 4.28: Ajo 14 - Luokkien painokerrointen kehitys, 4000 kierrosta



Kuva 4.29: Ajo 14 - Luokkien kaistankäytön kehitys, 4000 kierrosta

4.5.13 Ajo 14

Gain-arvot: kulta 3000, hopea 300, pronssi 100

Kuvaajassa 4.27 on ajon 14 tuottokäyrä, jossa kultaluokan gain-arvoksi on asetettu 3000, hopealuokan gain-arvoksi 300 ja pronssiluokan gain-arvoksi 100 (samat kuin ajossa 9). Käyrästä näkee, että tuotto vaihtelee enemmän suuremmalla määrällä kierroksia. Keskimääräinen tuotto on 120 yksikköä suurempi kuin pienemmällä kierrosmäärällä suoritetussa ajossa.

Myös tässä ajossa pidempi ajoaika vaikuttaa keskimääräisiin kaistankäyttöihin ja painokertoimiin aavistuksen laskevasti (kuvaajat 4.28 ja 4.29)

4.6 Johtopäätökset

Tässä kappaleessa kootaan vielä yhteen simulaatiotulosten perusteella aikaansaadut johtopäätökset. Simulaatioissa ei ole otettu huomioon sitä kuinka palveluiden hinnanmuutokset mahdollisesti vaikuttavat asiakkaiden palveluluokan valintaan.

Ensimmäinen johtopäätös: mitä suuremmaksi korkeamman prioriteetin luokkien gain-arvot asetetaan sitä vähemmän matalamman prioriteetin luokkien gain-arvoilla on vaikutusta kokonaistuottoon. Esimerkiksi ajojen 9 ja 10 keskimääräiset tuotot olivat miltei samat, vaikka pronssiluokan gain-arvo laskettiin nollaan. Tarkastelemalla ajoja 3 ja 4 huomataan, että keskimääräinen tuotto kasvaa kun pronssiluokan gain-arvo asetetaan nollaksi ja hopealuokan gain-arvoa kasvatetaan 20 yksiköllä. Samaan tapaan ajosta 6 nähdään, että ”siirtämällä” pronssiluokan gain-arvo kultaluokalle saavutetaan ajoa 4 suurempi keskimääräinen tuotto. Siinä tapauksessa, että korkean prioriteetin luokan gain-arvoa kasvatetaan, täytyy ottaa huomioon muiden luokkien saama palvelunlaatu. Kun korkean prioriteetin omaavan luokan gain-arvoa kasvatetaan, matalamman prioriteetin luokkien suoritusaika vähenee ja näin ollen sen käyttämä kaista pienenee. Näistä tuloksista pystytään päättelemään, että alimmille luokille voidaan tarjota edullista laskutusta nostamalla hieman muiden luokkien gain-arvoa. Tämän voi tehdä ilman, että kokonaistuotto kärsii. Gain-arvojen valinta tulisi kuitenkin tehdä siten, että matalamman prioriteetin luokan saavat kohtuullista palvelunlaatua.

Toinen johtopäätös: asettamalla luokkien gain-arvot mahdollisimman lähelle toisiaan, saavutetaan tasainen tuotto ilman suuria hetkellisiä vaihteluja tai keskimääräisen tuoton pienenemistä. Tuottokäyrän tasaantuminen tietylle tasolle vaatii tässä tapauksessa hieman pidemmän ajan kuin esimerkiksi ajoissa 9 ja 10. Lisäksi lähekkäin asetetuilla gain-arvoilla voidaan saavuttaa parempi tuotto: esimerkiksi ajon 5

keskimääräinen tuotto on jopa 1000 yksikköä suurempi kuin ajossa 7 (gain-arvojen summat lähes yhtä suuret). Tämä vaikuttaa luokkien saamaan suoritusaikaan ja kaistankäyttöön tasaavasti: korkean prioriteetin luokkien suoritusaika laskee ja matalan prioriteetin luokat puolestaan saavat hieman lisää suoritusaikaa. Mikäli verkon kokonaistuoton halutaan pysyvän tasaisena ja luokkien saavan tasaista palvelunlaatua, kannattaa palveluntarjoajan asettaa luokkien gain-arvot lähemmäksi toisiaan.

Kolmas johtopäätös: asettamalla korkeamman prioriteetin luokkien gain-arvot huomattavasti suuremmaksi kuin matalan prioriteetin luokkien gain-arvot, nousee tuotto nopeasti, mutta vaihtelee keskimääräisen tuoton molemmin puolin äkkinäisesti. Tuoton vaihtelu on sitä suurempi mitä suurempi on luokkien gain-arvojen välinen ero. Mikäli verkon kokonaistuoton halutaan saavuttavan mahdollisimman nopeasti maksimitasonsa, kannattaa korkean prioriteetin gain-arvot asettaa huomattavasti suuremmiksi kuin matalan prioriteetin luokkien gain-arvot. Gain-arvojen jakaminen luokkien välillä ei vaikuta tuottoon huomattavasti niin kauan kuin luokkien gain-arvojen summa pysyy samana. Sen sijaan suuret gain-erot luokkien välillä aikaansaavat suurempia eroja luokkien painokertoimissa ja kaistankulutuksissa.

Neljäs johtopäätös: Pidemmällä ajanjaksolla mitattuna luokkien keskimääräiset kaistankäytöt pienenevät hieman ja verkon kokonaistuotto näyttää kasvavan (Ajot 11-14).

Monipalveluverkon tuottoa simuloivien ajojen perusteella saatiin siis selville, että gain-arvoja muuttamalla pystytään vaikuttamaan palveluntarjoajan keräämän kokonaistuoton kehittymiseen, mutta myös luokkien kaistankulutukseen. Gain-arvoja säätäessä täytyy myös ottaa huomioon loppukäyttäjät: kuinka gain-arvon muuttaminen tulee näkymään käyttäjien laskutuksessa, palvelunlaadussa ja miten arvojen muuttaminen vaikuttaa siihen kuinka asiakas valitsee itselleen sopivan palveluluokan. Epäsopivilla gain-arvoilla voidaan vaikuttaa hinnoitteluun siten, että asiakkaat alkavat esimerkiksi suosia tiettyä luokkaa ja ajan myötä aiheuttaa kyseisen liikenneluokan ruuhkautumisen.

5 Yhteenveto

Tässä tutkielmassa käsiteltiin verkkoresurssien dynaamista jakoa keskittyen hinnoitteluun ja adaptiiviseen skedulointiin QoS-verkoissa. Verkon nykytila on se, että pääosin käytössä on vain yksi luokka, joka ei takaa erityistä palvelunlaatua. Internetin käyttäjät pyrkivät ottamaan käyttöönsä mahdollisimman suuren osan kaistasta. Tämänlainen verkkokäyttäytyminen aiheuttaa ongelmia muun muassa reaaliaikaisesti toimiville sovelluksille. Verkon palveluiden monimuotoistuessa palveluita kannattaa jakaa liikennevaatimusten mukaisiin luokkiin, jolloin verkossa tapahtuvat ruuhkat tai häiriökäyttäytymiset eivät vaikuta viivekriittisten sovellusten toimintaan. Palveluntarjoajan täytyy osata valita käyttöönsä sopivat mekanismit, joilla rajallinen kaista saadaan jaettua reilulla tavalla liikenneluokkien luokkien kesken. Vaikka kaistaa pystytään jakamaan palveluluokille nykyisten skedulointimenetelmien avulla, ne eivät reagoi toivotulla tavalla vaihteleviin voiden määriin tai vaihteleviin kaistavaatimuksiin. Staattiset reititinasetukset tullaan tulevaisuudessa korvaamaan hienostuneemmilla algoritmeilla, jotka säätävät asetuksia dynaamisesti verkon tilan mukaan. Staattisella reititinkokoonpanolla ei voida taata, että luokan sisällä olevat tietoliikennevuot saisivat toivottua palvelunlaatua. Aiemmissa kappaleissa esiteltiin palveluiden luokitteluun, verkon resurssien jakoon ja hinnoitteluun liittyviä perustekniikoita, sekä uusia menetelmiä joilla verkon käyttöastetta voidaan parantaa, palveluntarjoajan tuottoa maksimoida, sekä tarjota asiakkaille sellaista palvelunlaatua, josta he ovat maksaneet. Työn käytännön osuudessa tutkittiin simulaatioiden avulla adaptiivista resurssienjako menetelmää, jonka avulla tuottoa voidaan maksimoida ja jakaa verkon resurssit reilusti palveluluokkien kesken.

Viitteet

[1] Alexander Sayenko, *Adaptive scheduling for the QoS supported networks*, Thesis, Jyväskylän yliopisto, 2005

[2] Garrett Hardin, *The Tragedy of the Commons*, Science, New Series, Vol. 162, Number 3859 (Dec. 13, 1968) s. 1243-1248

[3] Kimmo Kaario, *TCP/IP-verkot*, Docendo, Jyväskylä, 2002

[4] Cisco Systems, *Cisco IOS 12.0 Quality of Service*, Indianapolis: Cisco Press, 1999

[5] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, *RFC2475: An Architecture for Differentiated Services*

[6] Chuck Semeria, *Supporting Differentiated Services Classes: Queue scheduling disciplines*, White paper, Juniper networks 2001

[7] R. Braden, D. Clark, *RFC1633: Integrated Services in the Internet Architecture: an Overview*

[8] Cisco Systems, *DiffServ - The Scalable End-to-End QoS Model*, White paper, 2002

[9] Andrew M. Odlyzko, *Internet traffic growth: Sources and implications*, Optical Transmission Systems and Equipment for WDM Networking II, Proc. SPIE, vol. 5247, 2003, s. 1-15.

[10] Andrew M. Odlyzko, *Paris Metro Pricing for the Internet*, Proc. ACM Conference on Electronic Commerce (EC'99), ACM, 1999, s. 140-147

[11] Hui Zhang, *Providing End-to-End Performance Guarantees Using Non-Work-Conserving Disciplines*, Computer Communications: Special Issue on System Support for Multimedia Computing, 18(10), Oct 1995.

[12] P.E. McKenny, *Stochastic Fairness Queueing*, 9th Annual Joint Conference of the IEEE Computer and Communication Societies, volume 2, s. 733-740, Jun 1990.

[13] R. Guérin, S. Kamat, V. Peris ja R. Rajan, *Scalable QoS provision through buffer management*, SIGCOMM, s. 29-40, Aug 1998.

[14] Seung Jae Shin ja Martin B.H. Weiss, *Simulation Analysis of QoS Enabled Internet Pricing Strategies: Flat Rate Vs. Two-Part Tariff*, Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS2003)

[15] Woan Sun Chang ja Robert Simon, *Performance Analysis for Multi-Service Networks with Congestion-based Pricing for QoS Traffic*, proceedings of the 38th Annual Simulation Symposium (ANSS2005)

[16] B. Jukic, R. Simon ja W. Chang, *Congestion based resource sharing in multi-service networks*, Decision Support Systems, 37, 2004.

[17] Rares Serban, Chadi Barakat ja Walid Dabbous, *A CBQ-Based Dynamic Resource Allocation Mechanism for Diffserv Routers*, SETIT 2003, March 2003, Sousse, Tunisia

[18] A. Capone, J. Elias, F. MARTIGNON, G. Pujolle, *Dynamic Resource Allocation in Communication Networks*, Networking 2006, Coimbra, Portugal, 15-19 May 2006

[19] H. Wang, C. Shen ja K.G. Shin, *Adaptive-Weighted Packet Scheduling for Premium Service*, Communications, 2001. ICC 2001. IEEE International Conference, Volume: 6, 2001 s. 1846-1850 vol.6

[20] J. Joutsensalo, T. Hämäläinen, M. Pääkkönen ja A. Sayenko, *QoS– and revenue aware adaptive scheduling algorithm*, Journal of Communications and Networks, vol. 6, no. 1, s. 68-77, Mar. 2004.

[21] J. Joutsensalo, T. Hämäläinen, K. Luostarinen ja J. Siltanen, *Adaptive scheduling method for maximizing revenue in Flat Pricing Scenario*, AEU – International Journal of Electronics and Communications, vol. 60, issue 2, February 2006, s. 159-167

[22] J. Joutsensalo, A. Viinikainen, M. Wikström, ja T. Hämäläinen, *Bandwidth allocation and pricing in multinode network*, Proceedings of the 20th International Conference on Advanced Information Networking and Applications, Volume 1 (AINA'06), s. 573-578

[23] A. Sayenko, T. Hämäläinen, J. Joutsensalo, ja P. Raatikainen, *Adaptive scheduling using the revenue-based Weighted Round Robin*, The 12nd IEEE International Conference On Networks (ICON 2004)) in press.

[24] A. Sayenko, T. Hämäläinen, J. Joutsensalo, ja P. Raatikainen, *Revenue-based adaptive Deficit Round Robin*, 3rd International Workshop on QoS in multiservice IP networks, s. 600-612, Feb 2005.

[25] A. Sayenko, T. Hämäläinen, J. Joutsensalo, L. Kannisto, *Comparison and analysis of the revenue-based adaptive queuing models*, published in Journal of Computer Networks (Special Issue on Quality of Service) Copyrights 2005 Elsevier.

[26] X. Wang ja H. Schulzrinne, *Pricing Network Resources for Adaptive Applications in a Differentiated Services Network*, IEEEACM TRANSACTIONS ON NETWORKING, VOL. 14, NO. 3, JUNE 2006

[27] X. Wang and H. Schulzrinne, *RNAP: a resource negotiation and pricing protocol*, Proc. NOSSDAV'99, Basking Ridge, NJ, Jun.1999, s. 77-93.

[28] Nan Jin, Gayathri Venkitachalam, Scott Jordan, *Dynamic Congestion-Based Pricing of Bandwidth and Buffer* IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 13, NO. 6, DECEMBER 2005, s. 1233-1246

[29] Neil J. Keon, G. Anandalingam, *Optimal Pricing for Multiple Services in Telecommunications Networks Offering Quality-of-Service Guarantees*, IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 11, NO. 1, FEBRUARY 2003, s. 66-80

[30] E. MAGAÑA, D. MORATÓ, P. VARAIYA, *Router Scheduling Configuration Based on the Maximization of Benefit and Carried Best Effort Traffic*, Telecommunication Systems 24:2-4, s. 275-292, 2003