Institut für Kommunikationsnetze und Rechnersysteme

Universität Stuttgart
Pfaffenwaldring 47
D–70569 Stuttgart

Diplomarbeit Nr. 3155

Faire Ressourcenaufteilung mit mehreren Qos-Klassen in zellulären Mobilfunknetzen

Bo Cao

Studiengang:	Informatik
Prüfer:	Prof. Dr.-Ing. Andreas Kirstädter
Betreuer:	Dipl.-Ing. Magnus Proebster
	Dipl.-Ing. Matthias Kaschub
Begonnen am:	22.02.2011
Beendet am:	24.08.2011
CR-Klassifikation:	C.2 D.3 I.6

Inhaltsverzeichnis

1 Einführung

1.1 Motivation und Zielstellung

Die Mobilfunknutzung entwickelt sich schnell. Im Jahr 2010 verfügte 78% der Weltbevölkerung über ein Mobilfunkgerät. Die Abbildung 1.1 zeigt, dass ungefähr 30% Weltbevölkerung einen Zugang zum Internet haben. Seit einigen Jahren kommt eine weitere Nutzung dazu. Für jeden zweiten Internetnutzer spielt die Möglichkeit das Internet über den Mobilfunk nutzen zu können bereits bei der Auswahl bzw. dem Kauf von Mobilfunkgeräten sowie beim Abschluss eines neuen Mobilfunkvertrages eine sehr wichtige Rolle. [1]

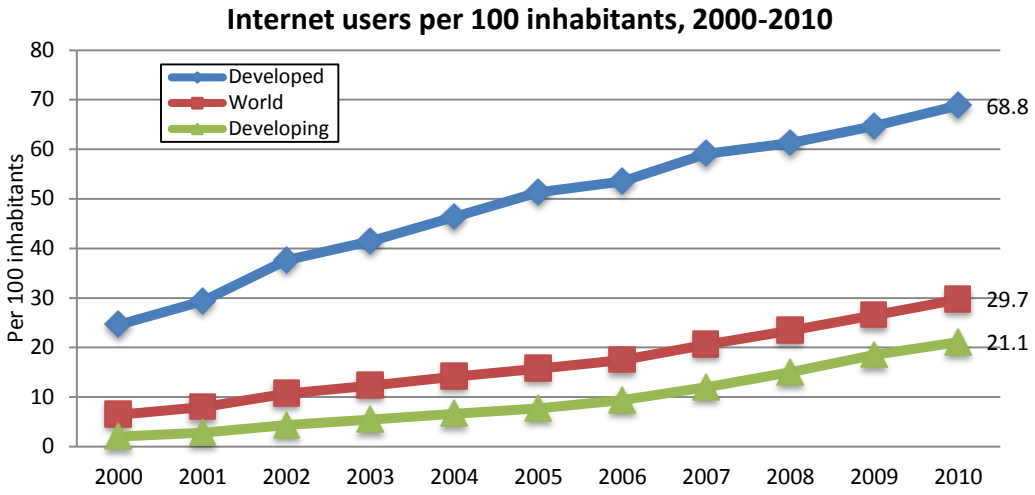

Abbildung 1.1 : Weltweite Internetnutzungsstatistiken [2]

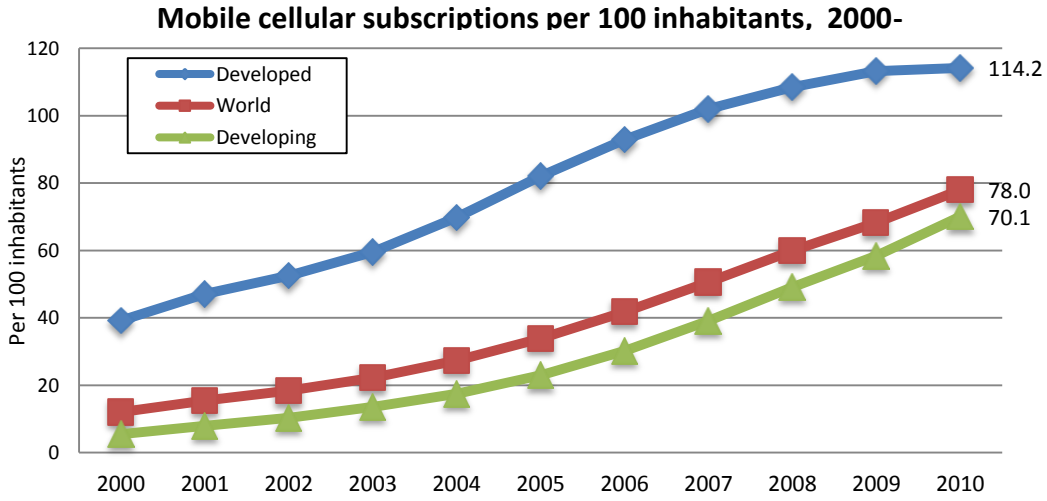

Abbildung 1.2 Weltweite Handynutzungsstatistiken [2]

Die zweite Generation des Mobilfunks heißt *Global System for Mobile Communications* (GSM). Diese Technologie gewährleistet den Dienst für Sprachverbindungen im Mobilfunknetz. Eine Internetnutzung ist nur mit geringen Datenübertragungsraten möglich. Eine Weiterentwicklung war die Einführung des U*niversal Mobile Telecommunications System* (UMTS). Mit Hilfe des H*igh Speed Downlink Packet Access* (HSDPA) Protokols und von H*igh Speed Up-link Packet Access* (HSUPA) verbessert sich sowohl die Geschwindigkeit des Downloads als auch des Uploads. Als Nachfolger von UMTS steht bereits der Mobilfunkstandard *Long Term Evolution* (LTE) zur Verfügung, der für das mobile Internet nochmals einen enormen Zuwachs der Übertragungsgeschwindigkeit bringen wird.

In drahtlosen Netzwerken war bisher vorwiegend der gesamte Durchsatz des Systems die wichtigste Performance-Metrik. Allerdings wird von der nächsten Generation drahtloser Funknetze erwartet, dass der Multimedia-Verkehr z.B Sprach- und Videoverbindungen, für den eine verzögerungsfreie Übertragung besonders wichtig ist, gesondert behandelt wird, da die Übertragungsbandbreite weniger stark ansteigt als die Nutzung durch Mobilfunkteilnehmer. Im modernen heterogenen Verkehr sind deswegen Q*uality Of Service* (QoS) und Systemfairness, die sich mit dieser Problematik befassen, ebenfalls wichtig. In diesem Fall ist der Konflikt zwischen steigendem Durchsatz des Systems und Fairness der verschiedenen Nutzer signifikant. Vor allem wird der Konflikt durch verschiedene anwendungsspezifische Szenarien balanciert.

Diese Arbeit beschäftigt sich mit dem Scheduler in der Basisstation. Die Abwägung zwischen Übertragungseffizienz und Fairness und einer adaptiven Parametrisierung der Fairnesseigenschaften eines Schedulers sollte untersucht werden. Es wurde gezeigt, dass die Fairness von der Last in der Zelle abhängt. In dieser Arbeit sollen die verschiedene Scheduler-Algorithmen implementiert werden. Die Leistungen, die durch Simulation aufgezeigt wurden, müssen hinsichtlich Fairness, Zelldurchsatz und Erfüllungsgrad der QoS-Anforderungen analysiert und diskutiert werden.

1.2 Gliederung der Arbeit

Die vorliegende Arbeit gibt im folgenden Kapitel einen grundlegenden Überblick über die Grundbegriffe der Mobilfunktechnologie, den Begriff *Quality of Service* sowie den Scheduler in der Basisstation. Im Kapital drei werden die Priorisierungsmechanismen erklärt. Die Funktionsweise, der Pseudocode und die erwartete Ergebnisse werden erläutert. Vor der Simulation werden in Kapitel vier die Simulationsumgebung und methodische Auswertung betrachtet. Kapitel fünf behandelt die Simulation der Algorithmen und zeigt die Ergebnisse des verschiedenen Verfahrens auf. Die Arbeit schließt mit dem sechsten Kapitel als Zusammenfassung und Ausblick auf mögliche weitere Einsätze.

2 Grundlagen

2.1 Long Term Evolution

Ein Vorläuferkonzept zu Long Term Evolution(LTE) wurde von Nortel Networks unter dem Namen *High Speed OFDM Packet Access* (HSOPA) vorgestellt. LTE verwendet *Orthogonal-Frequency-Division-Multiplexing-Techniken* (OFDM) sowie *Multiple-Input-Multiple-Output-Antennentechnologie* (MIMO). Damit soll es den Mobilfunkanbietern möglich sein, kostengünstig hochgradige Datendienste anzubieten und so das mobile Internet zum Massenmarkt zu machen. Die geringen Latenzzeiten bei LTE erlauben die Übertragung von Sprachdiensten (VoIP) und Videotelefonie über das Internetprotokoll sowie den Einsatz zeitkritischer Anwendungen wie zum Beispiel Online-Spielen[3].

Bereits jetzt hat UMTS sehr hohen Datendurchsatz ermöglicht. LTE unterstützt im Gegensatz zu UMTS verschiedene Bandbreiten (1, 3, 5, 10, 15 und 20 MHz) und kann so flexibel in unterschiedlichen zukünftigen Spektren eingesetzt werden. Bei 20 MHz (entspricht laut Standard der Benutzung von 1200 Unterträgern) sollen Spitzendatenraten von 300 Mbps im Downlink und 75 Mbps im Uplink mit Latenzzeiten unter 5 ms erreicht und so die langfristige Konkurrenzfähigkeit von UMTS-Systemen gesichert werden[4].

Category		1	2	3	4	5
Peak rate Mbps	DL	10	50	100	150	300
	UL	5	25	50	50	75
Capability for physical functionalities						
RF bandwidth		20MHz				
Modulation	DL	QPSK, 16QAM, 64QAM				
	UL	QPSK, 16QAM				QPSK, 16QAM, 64QAM
Multi-antenna						
2 Rx diversity		Assumed in performance requirements.				
2x2 MIMO		Not supported	Mandatory			
4x4 MIMO		Not supported				Mandatory

Abbildung 2.1 LTE Eigenschaften Übersicht[4]

Down Link

Beim Downlink wird die OFDM Technik eingesetzt. In einem OFDM System wird das

verfügbare Spektrum auf mehrere Träger, die als Unterträger bezeichnet werden, verteilt. Bei einer niedrigen Datenrate sind alle Unterträger unabhängig moduliert. OFDM ist auch in WLAN, WiMAX und Broadcast-Technologien wie DVB eingesetzt. OFDM hat mehrere Vorteile einschließlich ihrer Robustheit gegen Multipath Fading und seiner effizienten Empfängerarchitektur.

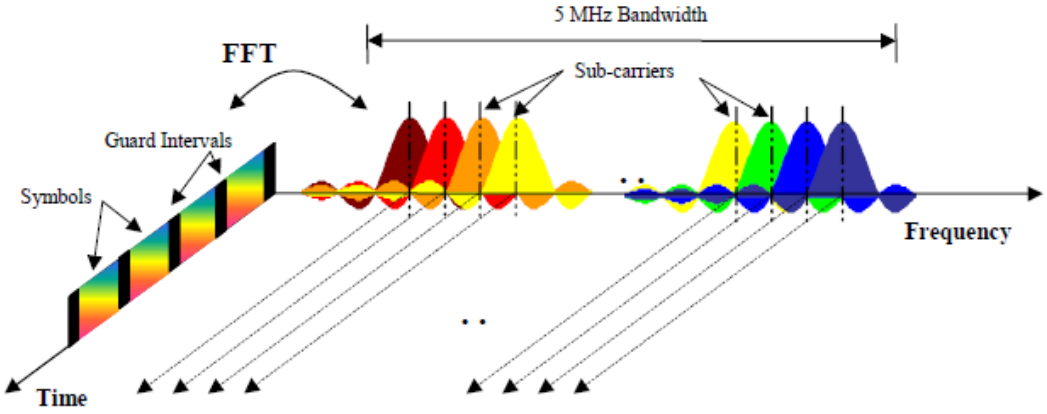

Abbildung 2.2 Frequenz-Zeit Darstellung eines OFDM Signals[6]

Abbildung 2.2 stellt ein OFDM Signal von [5] dar. In dieser Abbildung wird ein Signal mit 5 MHz Bandbreite gezeigt. Daten sind unabhängig moduliert und werden durch eng nebeneinander stehende Unterträger übertragen. Die maximale Downlink-Datenrate hängt von der Ausstattung ab und liegt bei bis zu über 100 MBit/s bei Kanalbandbreiten von 20 MHz[6].

Up Link

Die Uplink-Übertragung in LTE ist auf *Single Carrier Frequency Division Multiple Access* (SC-FDMA) mit zyklischem Präfix basiert. SC-FDMA Signal hat eine bessere PAPR Eigenschaften im Vergleich zu einem OFDMA-Signal. Dies ist einer der Hauptgründe, dass SC-FDMA von LTE Uplink Access Model ausgewählt wird.

Um die Daten zwischen LTE Basisstation und *User Equipment* (UE) zu übertragen, weist die Basisstation den UEs in bestimmter Frequenz die Ressourcen zu. Der Scheduler kann unter Berücksichtigung auf QoS Parameter, UE Puffer Status, Kanalqualität, etc. die Entscheidung fällen, welche UE die Ressourcen benutzen darf.

2.2 *Quality of Service*

Quality of Service (QoS) beschreibt die Güte eines Kommunikationsdienstes aus der Sicht der Anwender, das heißt, wie stark die Güte des Dienstes mit deren Anforderungen übereinstimmt. Formal ist QoS eine Menge von

Qualitätsanforderungen an das gemeinsame Verhalten beziehungsweise Zusammenspiel von mehreren Objekten[7].

2.2.1 Qualitätsparameter

VoIP, Video Streaming und FTP sind heute häufig verwendete Dienste im Internet. Solche Dienste haben eigene Anforderungen bei der Übertragung, wie geringe Verzögerung oder garantiert konstante Übertragungsraten. Jede Anwendung erwartet von einem Netzwerk bestimmte Eigenschaften. Dabei sind die Parameter wichtig, durch die die Qualität der Übertragung bestimmt wenden kann[8]. Hierbei spielen die folgenden Parameter eine tragende Rolle: Laufzeitschwankungen (Jitter), Übertragungsverzögerung (Delay), die Paketverlustrate (Packet-Loss-Rate) und der Datendurchsatz (Throughput).

Laufzeitschwankungen

Laufzeitschwankungen stellen die Abweichung der Latenzzeit von ihrem Mittelwert dar. Diese treten nur dann auf, wenn die Pakete von Echtzeit-Streaming-Anwendungen (z.B VoIP) mit Verzögerungen gesendet und empfangen werden. Die Gesprächsqualität wird dadurch stark beeinflusst und im schlimmsten Falle so schlecht, dass der Gesprächspartner nicht mehr zu verstehen ist.

Übertragungsverzögerung

Es geht um die Laufzeit eines Pakets bei der Übertragung von der Nutzer-Queue bis zum Ziel. Auf der Netzwerkebene ist die Übertragungsverzögerung die Summe der Verarbeitungszeit, sowie der Übertragungs-, der Warteschlangen- und der Ausbreitungsverzögerung.

Paketverlustrate

Die Paketverlustrate ist die Wahrscheinlichkeit, dass einzelne IP-Pakete bei der Übertragung verloren gehen. Paketverlust kann in allen Anwendungen vorkommen. Allerdings kann bei den Echtzeit-Anwendungen eine begrenzte Paketverlustmenge toleriert werden. Im Gegensatz dazu sind nicht Echtzeit-Anwendungen bei Datenverlust relativ sensibel.

Datendurchsatz

Das ist die Datenmenge, die pro Zeiteinheit übertragen wird. Datendurchsatz kann als maximale Rate oder durchschnittlichen Rate ausgedrückt werden.

Die Laufzeitschwankungen machen sich vor allem bei Echtzeitanwendungen

bemerkbar. Zum Beispiel erscheint bei großer Latenz eine Eingabe erst mit einer gewissen Zeitverzögerung auf dem Bildschirm. Für reine Dateitransfers ist üblicherweise der Gesamtdurchsatz der entscheidende Parameter, die individuelle Laufzeitschwankungen und Verlustrate hingegen sind hier weniger von Bedeutung. Für Echtzeitkommunikation wie z. B. VoIP hingegen spielen die Latenz, die Laufzeitschwankungen und die Verlustrate eine weitaus größere Rolle, weil sie maßgeblich die Sprachverständlichkeit beeinflussen[7].

Laufzeitschwankungen sind primär bei Echtzeitanwendungen, wie VoIP, von Bedeutung. Dort führen sie im Falle von VoIP zusammen mit einer hohen Latenz zu störenden Pausen oder bei einem hohen Paketverlust zu Aussetzern, die die Verständigung erschweren oder sogar unmöglich machen. Für reine Dateitransfers wie z.B. Dateiarchive spielen diese Faktoren keine Rolle. Dort ist vor allem die Gesamtübertragungsgeschwindigkeit der entscheidende Parameter.[7]

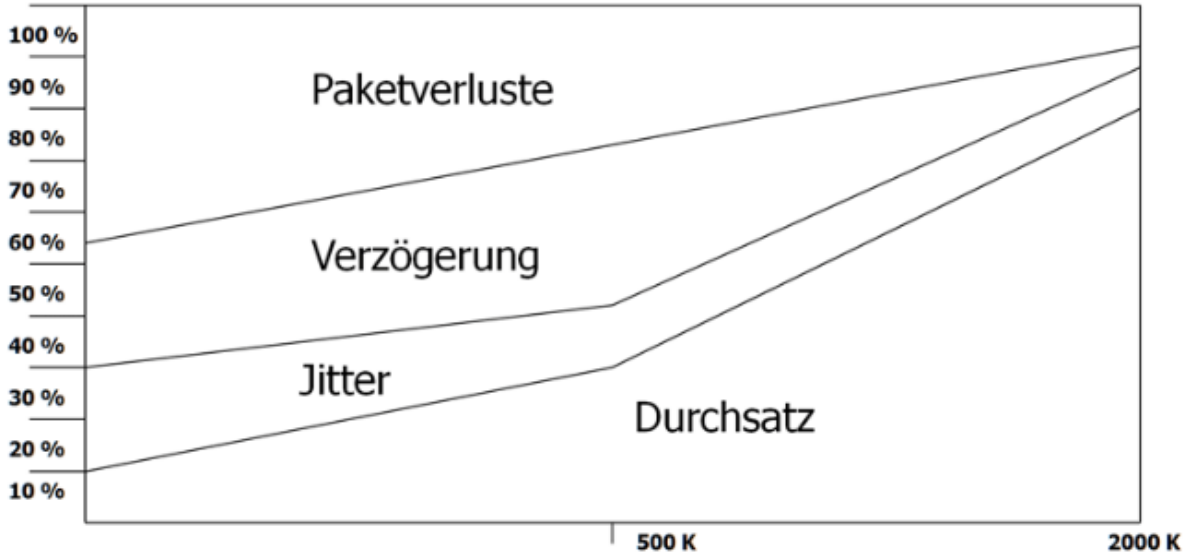

Abbildung 2.3 Die Gewichtung der der QoS-Parameter[8]

Darüber hinaus gibt es auch andere Parameter, die Qualität der Übertragung beeinflussen kann. z.B IP Packet Delay Variation, Spurious IP Packet Rate, IP Service Availability usw.

2.2.2 QoS Architekturen

In einer QoS-Architektur wird das Zusammenspiel der verschiedenen Ende-zu-Ende-Komponenten bzgl. Spezifikation, Verteilung, Bereitstellung von Dienstgüte definiert. Es gilt grundsätzlich, zwei unterschiedliche Arten des QoS zu unterscheiden, das „Best Effort“ QoS und „echtes“ QoS.

Best Effort bezeichnet eine minimalistische Dienstgüte-Zusicherung in Telekommunikationsnetzen. Der Betreiber des Netzes sagt dessen Benutzern zu,

eingehende Übermittlungsanfragen schnellstmöglich und im Rahmen der ihm zur Verfügung stehenden Ressourcen nach besten Möglichkeiten zu bedienen[9]. Bei „echtem“ QoS hingegen werden die Datenpakete bestimmte Verbindungen zugeordnet, deren QoS-Parameter permanent überwacht und bei Bedarf korrigiert werden. Für die Zuweisung von QoS auf einzelne Datenverbindungen unterscheidet man zwischen zwei maßgeblichen Ansätzen[10]:

Integrated Services (IntServ) : QoS für einzelne Verbindungen
Differentiated Services (DiffServ): QoS für Verkehrsklassen von Verbindungen

IntServ

IntServ ist ein feingranulares QoS-Verfahren zur Priorisierung von Datenpaketen. Im Gegensatz zum DiffServ-Verfahren werden die Ressourcen für einzelne Verbindungen und nicht für Verkehrsklassen angefordert. d.h Jeder Verbindung darf alle zur Verfügung stehenden Ressourcen bekommen.
Es ist zwei grundlegende Dienstklassen in Intserv erkannt:

Guaranteed QoS: Bietet eine zugesicherte Übertragungsrate und eine quantitative Aussage über die maximale Verzögerung durch Warteschlangen im Netzwerk. Weiterhin wird zugesichert, dass keine Pakete durch zu volle Warteschlangen verloren gehen.[11]

Controlled Load: Macht lediglich das Versprechen, dass ein sehr großer Teil der ankommenden Pakete vom Router nicht verworfen wird und dass die Verzögerung durch die Warteschlange gegen Null geht. Es werden allerdings keine genauen Zusagen in Zahlen ausgedrückt[11].

DiffServ

Der größte Unterschied zwischen IntServ und DiffServ ist, ob die Datenpakete zur Priorisierung klassifiziert sind. Es werden also keine Ressourcen wie bei IntServ reserviert, sondern die einzelnen Pakete werden für ihre Übermittlung über das Netz mehreren Verkehrsklassen zugeordnet.

Application	*Traffic Category*	*Percentage of Users*
FTP	Best effort	10 %
Web Browsing / HTTP	Interactive	20 %
Video Streaming	Streaming	20 %
VoIP	Real-time	30 %
Gaming	Interactive real-time	20 %

Abbildung 2.4 Qos-Klassen im UMTS Mobilfunk[18]

Im UMTS Mobilfunk sind vier QoS-Klassen definiert worden:

Background: Paketübertragung mit geringen Fehlertoleranzen und ohne kritische Anforderungen an Bandbreite, Laufzeitschwankungen und Übertragungsverzögerung.

Interactive: Interaktive Verwendung wie HTTP Web Browsing. Im Vergleich zum Background sind die Anforderungen jedoch höher. Um die Wartezeit bei der Nutzung zu minimieren, sollt die Übertragungsverzögerung kleiner werden.

Streaming: Z.B. Video Streaming. Eine minimale Bandbreite ist erforderlich. Weil bei der Empfängerseite ein Datenbuffer verwendet wird, sind kleine Laufzeitschwankungen nicht kritisch.

Conversational: Direkte Telefonie (VoIP). Eine bestimmte Bandbreite muss auch reserviert werden. Für eine zulässige Telefonie sind wenige Übertragungsverzögerungen und Laufzeitschwankungen erforderlich.

2.3 Scheduler

Die Aufgabe des Schedulers ist, die Entscheidung zu treffen, welche im Puffer gespeicherten Pakete zu welchem Zeitpunkt übertragen werden. Die Entscheidung muss in einer sehr kleinen Zeitspanne getroffen werden. Je nach gegebener UMST-QoS-Klasse muss der Scheduler andere Entscheidungen treffen. Um die QoS Bedürfnisse in verschiedene Klassen zu erfüllen, ist ein effektiver Scheduler in jede Basisstation erforderlich.

2.3.1 Scheduling Algorithmen

In Mobilfunknetzen stehen viele verschiedene Arten von Anwendungen zur Verfügung. Jede Anwendung hat bestimmte Anforderungen an das System. Darüber hinaus gibt es verschiedene Benutzer, die unterschiedliche Qualitätsansprüche haben. Dies erhöht die Notwendigkeit zwischen den verschiedenen Nutzeransprüchen und Anwendungsanforderungen abzuwägen. Einerseits müssen die Anforderungen von Benutzern gut beachtet werden, auf der anderen Seite muss das System den Durchsatz in einige Maßen gewährleisten.

***First In First Out* (FIFO)**

FIFO ist das einfachste Schedulingverfahren. Es gibt nur eine Warteschlange für alle Benutzer. Die Pakete von allen Benutzern werden nach Ankunftszeit in der Warteschlange gespeichert. Und die Pakete, die zuerst gespeichert wurden, werden auch zuerst wieder aus der Warteschlange entnommen. Der Scheduler kann die Pakete von verschiedenen Benutzern nicht unterscheiden, daher ist auch die Prioritätszuweisung unmöglich.

Round Robin

Die einfachste Form eines fairen Scheduling System ist Round Robin. Jede Benutzer haben eigene Warteschlange. Der Scheduler weist die Ressourcen in zyklische Weise an alle Benutzer. Wenn eine Warteschlange leer ist, wird diese Schlange übersprungen.

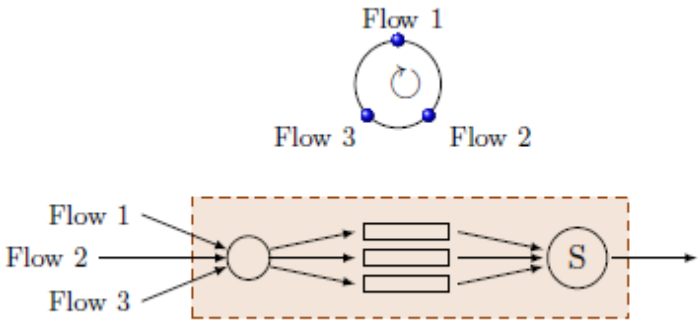

Abbildung 2.5 Round Robin Scheduling Verfahren[12]

Priority Scheduling

Prioritäten können entweder statisch zugewiesen werden oder dynamisch auf einzelne Pakete gemäß ihrer Verzögerung und Geschwindigkeit zugeordnet werden. Auch die Einteilung von Prozessen in verschiedene Prioritätenklassen ist möglich.

In *Priority Scheduling* werden nur die Pakete übertragen, wenn es keine andere Pakete aus höherer Priorität zu Übertragung steht. Innerhalb einen einzelnen Prioritätenklassen wird typischerweise der obengenannte *Round Robin Scheduling* oder FIFO Scheduling verwendet.

Queue Length Dependent Scheduling

Mit diesem Ansatz werden die Benutzer nach der Größe ihrer Warteschlange bedient. Es können entweder Benutzer mit langen Warteschlangen oder Benutzer mit kurzen Wartezeiten priorisiert werden, was mit *Long Queue First* (LQF) bzw. *Short Queue First* (SQF) Scheduling bezeichnet wird.

In Abbildung 2.6 werden die oben beschriebenen Scheduling Algorithmen zusammengefasst.

name	summary	advantages	disadvantages
First In First Out (FIFO)	Packets are served in the order of their arrival.	Easy to implement.	No protection between users. No priorities possible.
Round Robin (RR)	Resources are distributed in a cyclic and fair manner.	Provides fairness and protection.	In case of several user classes weitghted RR shall be applied.
Priority Scheduling	Priorities are assigned to packets either statically or dynamically. Packets with a higher priority are served first. Within one priority class, RR or FIFO shall be applied.	Flexible priority assignment possible. Protection between different priority classes.	High implementation complexity.
Queue Length Dependent Scheduling	Users are served according to their queue length.	Easy to implement.	Users with the opposite queue size often have very poor performance.

Abbildung 2.6 Scheduling Algorithmenvergleich[12]

2.3.2 Proportional Fair Scheduling Algorithmus

Proportional Fair Scheduling ist ein Algorithmus, der den Konflikt zwischen Datendurchsatz und Fairness balancieren kann. Proportional Fair Scheduling versucht den gesamten Durchsatz zu maximieren, und gleichzeitig allen Benutzern in gewissem Maß den Dienst zu garantieren. Dieses wird durch die Zuordnung aller Pakete mit einer Datenrate oder eine Scheduling-Priorität (je nach Implementierung) realisiert.

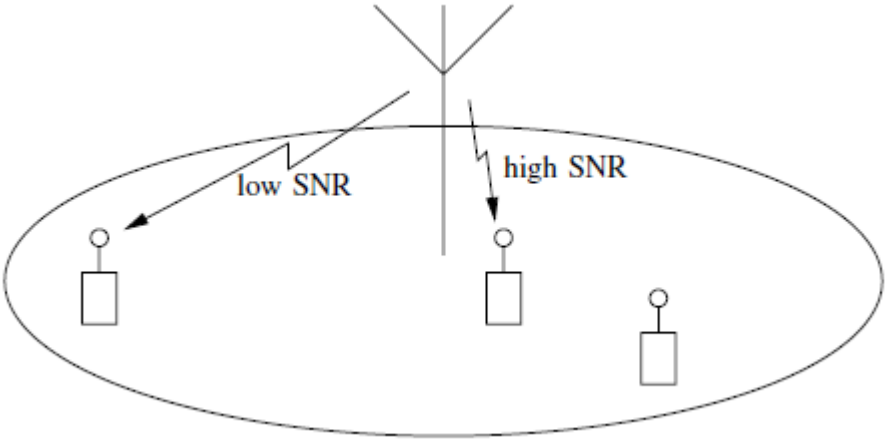

Abbildung 2.7 Ein Wireless-System [14]

Der Proportional Fair Scheduler basiert auf der Maximumprioritätsfunktion. Die Maximumprioritätsfunktion in der Basisstation sieht folgend aus:

$$P = \frac{T^{\alpha}}{R^{\beta}}$$ [13]

T bezeichnet die potentiell erreichbare Datenrate in einen Zeitpunkt. R ist die historische durchschnittliche Datenrate. α und β entsprechen den Fairnessparametern. Durch die Anpassung von α und β sind wir in der Lage, die Balance zwischen den Benutzern in den besten Kanalbedingungen und den höheren QoS Nutzerbedürfnissen zu regulieren[13].

Im Extremfall (α = 0 und β = 1) wirkt der Scheduler wie der Round-Robin Algorithmus und bedient alle Handys gleich häufig. Wenn α = 1 und β = 0, dann bedient der Scheduler immer nur die Nutzer mit den besten Kanal Bedingungen. In diesem Fall wird der Durchsatz maximiert. Mit α = 1 und β = 1 ist der Proportional Fair Scheduler getroffen.

In 3G Mobilfunknetz kennt der Scheduler die Datenrate im nächste Frame für alle Nutzer, weil jeder Nutzer in jedem Frame das momentane *signal to noise ratio* (SNR) in einer *Data Rate Control* Nachricht (DRC) an die Basisstation sendet. DRCi(t) hat für den Nutzer i zum Zeitpunkt t die erreichbare Data Rate. In dieser Arbeit wird ein unendliches Scheduler Modell simuliert. d.h es gibt genügend Pakete zu übertragen. Im Proportional Fair Scheduling wird die langfristige durchschnittliche Rate für jede Nutzer in Vektoren R = (R_1, . . . , R_i) gespeichert. Von *Gradient Algorithmen with Minimum/Maximum Rate* (GMR)

$$i \in \arg\max_{i \in I} e^{a_i T_i(t)} H_i'(R_i(t)) DRC_i(t)$$ [14]

Dabei ist R_i (t) die aktuelle durchschnittliche Rate für Nutzer i. T_i(t) ist ein Token Counter. Und a_i ist ein Parameter.

Proportional Fair with Minimum/Maximum Rates (PFMR) hat die Funktion:

$$i \in \arg\max_{i \in I} e^{a_i T_i(t)} \frac{DRC_i(t)}{R_i(t)}.$$ [14]

Bei der *Standard Proportional Fair* Scheduler sieht die Funktion so aus:

$$i \in \arg\max_{i \in I} DRC_i(t)/R_i(t).$$ [14]

Im *Proportional Fair* Scheduler ist die durchschnittliche Rate R_i(t) durch folgende Funktion aktualisiert:

$$R_i(t+1) = (1-\beta)R_i(t) + \beta\mu_i(t)$$ [14]

Hier ist β> 0 ein kleiner fester Parameter. Das Token Counter T_i wird wie folgt aktualisiert:

$$T_i(t+1) = T_i(t) + R_i^{token} - \mu_i(t)$$ [14]

Dabei sind $R_i^{token} = R_i^{min}$, wenn $T_I(t)>0$, und $R_i^{token} = R_i^{max}$, wenn $T_I(t)<0$. Wenn $R_i^{max} = \infty$, dann ist für einen Nutzer i, die vereinfachte Aktualisierung :

$$T_i(t+1) = \max\{0, T_i(t) + R_i^{\min} - \mu_i(t)\}$$ [14]

Genau umgekehrt ist es bei $R_i^{min} = 0$, dann aktualisiert das Token Counter mit:

$$T_i(t+1) = \min\{0, T_i(t) + R_i^{\max} - \mu_i(t)\}$$ [14]

Das Token Counter ist ein Schlüsselmechanismus im Proportional Fair Scheduler. Es kann sicherstellen, dass die Rate des Nutzers zwischen R^{min} und R^{max} bleibt. Für jede Nutzer gibt es eine virtuelle Token Schlange, die negativ sein kann. Die Schlange ist mit der Rate R_i^{min} (wenn T_i ist positiv) oder R_i^{max} (wenn T_i ist negative) wird in jedem Frame gefüllt. Wenn der Nutzer i in Frame t bereits bedient wird, müssen DRC_i(t) Pakete von der Schlange entfernt werden. Deshalb nimmt die

Schlange T_i in Frame *t* zu, wenn R_i kleine als R_i^{min} ist. Dadurch erhöht sich die Chance des Nutzers *i* im nächsten Frame Ressourcen zu bekommen. Wenn im Zeitpunkt *t* R_i größer als R_i^{max} ist, nimmt die Schlange ab. Die Chance bedient zu werden ist dann allmählich gesunken. Es ist auch möglich, dass R_i zwischen R_i^{min} und R_i^{max} liegt. In diesem Fall hat die Schlange eine positive Tendenz, wenn Ti negativ ist. Bei negativem Ti ist es umgekehrt[14]. In dieser Arbeit wird der Proportional Fair Scheduler in allen Simulationen eingesetzt. Auf diesem Scheduler werden einige Priorisierungsmechanismen implementiert.

In dieser Arbeit wurde der Proportional Fair Algorithmus bei der Bandbreiteverteilung innerhalb einer Prioritätskasse eingesetzt. Anhand der Shannon Kapazität berechnet der Scheduler für jede Frequenz für alle Nutzer in der ausgewählten Klasse die Übertragungskapazität. Der Nutzer mit der größten Kapazität im Vergleich zu allen anderen Nutzern in derselben Klasse darf seine Pakete in dieser Frequenz übertragen. In dieser Phase werden nur die Nutzer, in deren Schlange noch Pakete zur Übertragung sind, betrachtet. Sonst werden sie in diesem Frame übersprungen. Nachdem alle Bandbreite zugewiesen wurde, sind die Ressourcen in diesem Frame fertig zugeteilt.

3 Priorisierungsmechanismen

In diesem Kapitel werden die Vorgehensweise alle Algorithmen genau erklärt. Ferner werden die Pseudocode und die erwartete Ergebnisse erläutert.

3.1 Algorithmus für die statische Priorisierung

3.1.1 Vorgehensweise des Algorithmus

Bei statischer Priorisierung werden die Nutzer in Prioritätsklassen eingeteilt. Die Anzahl der Prioritätsklassen ist durch das Simulationskontrollsystem frei wählbar. Der Scheduler wird in jedem Frame alle Prioritätsklassen durchlaufen. Eine Klasse mit niedriger Priorität wird nur betrachtet, wenn alle Klassen mit höherer Priorität in diesen Frame keine Pakete zu übertragen haben. Sobald der Scheduler eine Prioritätsklasse ausgewählt hat, stehen alle freien Frequenzen für diese Klasse zur Verfügung. Bei der Bandbreiteverteilung innerhalb einer Prioritätskasse wurde der Proportional Fair Algorithmus eingesetzt.

Bei geringer Last kann es sein, dass alle Pakete in der ausgewählten Klasse übertragen werden, allerdings die Bandbreite in diesem Frame nicht ausgenutzt wird. Wenn in diesem Fall die Ressourcen weiter der ausgewählten Klasse zugeteilt werden, obwohl die Warteschlange der zugehörigen Nutzer leer ist, sind die Ressourcen für diese Klasse überflüssig. Ressourcen, die in einen Frame zu viel zugeteilt sind, werden als „*Padding*“ bezeichnet. Die nächste Prioritätsklasse, die eine Stufe niedriger als die ausgewählte Klasse ist, kann die Restbandbreite in diesen Frame zugeteilt bekommen. Dies kann das Padding vermeiden.

3.1.2 Pseudocode

Das folgende Listing enthält den Pseudocode des Algorithmus für die Statischen Priorisierung:

```
Liste priorityClasses //alle Klassen nach Priorität unterteilt
Liste selectedClasses
FÜR JEDE Klasse in priorityClasses
    WENN Klasse enthält Benutzer mit nicht leerer Warteschlange
        Füge Klasse zu selectedClasses hinzu
ENDE FÜR
```

```
FÜR JEDE Klasse in selectedClasses
    FÜR Jede Frequenz in allen momentan freien Frequenzen
        FÜR ALLE Benutzer in aktueller Klasse
            WENN Benutzer noch Pakete zu übertragen hat DANN
                Proportional Fair Algorithmus (Nutzerindex, Frequenz)
                WENN Kapazität des aktuellen Benutzers > Kapazität bisher bester
                Nutzer DANN
                    aktueller Benutzer wird zu bester Nutzer
                ENDE WENN
            ENDE WENN
        ENDFOR
        Weise aktueller Frequenz den besten Nutzer zu
    ENDFOR
ENDFOR
```

3.1.3 Erwartete Ergebnisse

Der Algorithmus für die statische Priorisierung ist relativ unfair, weil höhere Priorisierungsklassen bei hohen Systemlasten die gesamte Bandbreite beanspruchen und die anderen Klassen keine Chance haben Pakete zu übertragen. In der Anfangsphase der Simulation ist die Systemlast relativ leicht. Die hohe Prioritätsklasse bekommt die meisten Ressourcen, und die unteren Klassen bekommen nur die Ressourcen, die für obere Klasse zu viel sind.

3.2 Algorithmus für gewichtete Priorisierung

3.2.1 Vorgehensweise des Algorithmus

Wie im Fall der statischen Priorisierung sind alle Nutzer bei der gewichteten Priorisierung in verschiedene Klassen verteilt. Die Anzahl der Klassen ist frei wählbar. Der Unterschied bei dem statischen Priorisierung Algorithmus liegt darin, dass jede Klasse einen Gewichtsfaktor besitzt. Dieser Gewichtsfaktor entspricht der Priorität für die jeweilige Klasse. In jedem Frame berücksichtigt der Scheduler alle Nutzer bei der Verteilung. Für jede Frequenz wird für alle Nutzer der Proportional Fair Algorithmus verwendet um eine Kapazität zu berechnen. Die dadurch ausgerechnete Kapazität wird mit dem zugehörigen Gewichtsfaktor, der Klassenweise vergeben wurde, multipliziert. Der Scheduler vergleicht die entstandenen mathematischen Produkte aller Nutzer und teilt dann der Frequenz den Nutzer mit dem größten Produkt, also mit der maximalen gewichteten Kapazität zu.

Der Gewichtsfaktor spielt bei der gewichteten Priorisierung eine große Rolle. Er

stellt die Priorität jeder Klasse dar. Wenn alle Klassen den gleichen Gewichtsfaktor haben, bedeutet es, dass alle Klassen gleich behandelt werden. Wenn eine Klasse über einen sehr großen Gewichtsfaktor verfügt, ist die Priorität dieser Klasse auch viel höher als die der anderen Klassen. Und die Ressourcen sind auch meistes dieser Klasse zugeteilt.

Im Vergleich zur statischen Priorisierung hat die gewichtete Priorisierung die Priorität quantitativ, also in Zahle, formuliert. Zusätzlich geht auch der SINR in die Berechnung der Priorität ein. So können nun z.B. auch niedrig priorisierte Nutzer mit einem sehr guten SINR Wert an die Reihe kommen, was bei dem statischen Scheduler nicht möglich wäre. Das ist möglich, weil alle Nutzer klassenübergreifend betrachtet werden.

Um Padding zu vermeiden, werden Nutzer, deren Warteschlangen keine Pakete haben, nicht betrachtet

3.2.2 Pseudocode

Das folgende Listing enthält den Pseudocode des Algorithmus der gewichteten Priorisierung:

```
FÜR JEDE Frequenz aller freien Frequenzen
    BesteGewichtetePriorität = 0
    FÜR JEDEN Nutzer aller Nutzer
        WENN Warteschlange des Nutzers nicht Leer DANN
            AktuelleGewichtetePrioriät = Gewichtsfaktor [Nutzer] * Proportional Fair
            Algorithmus (Nutzer, Frequenz)
            WENN AktuelleGewichtetePriorität > BesteGewichtetePrioriät DANN
                BesteGewichetetPriorität = AktuelleGewichtetePriorität
                BesterNuzter = Nutzer
            ENDE WENN
        ENDE WENN
    ENDE FÜR
    Markiere BesterNutzer als Übertragenden für diese Frequenz
ENDE FÜR
```

3.2.3 Erwartete Ergebnisse

Beim gewichteten Priorisierung Algorithmus ist der gesamte Systemdurchsatz potentiell höher als beim statischen Priorisierung Algorithmus, weil der Scheduler im

gewichteten Priorisierung Algorithmus alle Nutzer auswählen kann und deswegen auch Nutzer berücksichtigt, die sehr hohen SINR-Werte haben und deswegen viele Daten übertragen können. Bei der Frequenzvergabe sind in der statischen Priorisierung nur die Nutzer in eine Klasse von Scheduler berechnet worden. Wenn das System unter hoher Last ist, kann der gewichtete Algorithmus gewährleisten, dass alle Klassen Ressourcen erhalten. Das Verhältnis der Ressourcenzuteilung zwischen zwei Klassen bei der Datenübertragung entspricht dem Verhältnis der Gewichtsfaktoren. Bei niedriger Systemlast sieht die Ressourcenverteilung ähnlich wie im statischen Priorisierung Algorithmus aus, die Klasse mit dem größtem Gewichtsfaktor bekommt die meisten Ressourcen, die anderen Kassen bekommen so viele Ressourcen, wie es ihrem Gewichtsfaktor entsprechen würde.

3.3 Algorithmus zur reservierten Klassebandbreit Priorisierung

3.3.1 Vorgehensweise des Algorithmus

Im Algorithmus zur reservierten Klassebandbreit Priorisierung(KBP) sind die Nutzer auch in verschiedene Klassen eingeteilt. Jede Klasse verfügt über einen Parameter, der den Bandbreitenanteil bezeichnet. Der Parameter gibt an, wie viel Prozent der gesamten Bandbreite diese Klasse maximal besetzen darf. Für die Parameter P_i muss die folgende Gleichung gelten:

$$\sum_{i=0}^{n} P_i = 1$$

Daher ist n die Anzahl der Klassen. i entspricht dem Klasseindex.

In jedem Frame fängt der Scheduler mit der obersten Klasse an. Die oberste Klasse kann aus allen Frequenzen frei auswählen, und bekommt die Frequenzen, bei der die Nutzer dieser Klasse eine maximale Kapazität erreichen können. Für die unteren Klassen stehen nur noch die Frequenzen zur Verfügung, die nach der Auswahl der oberen Klassen noch nicht besetzt worden sind. Bei niedriger Systemlast werden die freien Frequenzen, die für eine der höheren Klassen nicht benötigt werden, an alle unteren Klassen verteilt. Die Verteilung erfolgt hierbei proportional zu den ursprünglichen Bandbreitenanteilen der restlichen Klassen.

Durch den Parameter ist die Ressourcenzuteilung nach Klassen manipulierbar. Innerhalb einer Klasse ist der Proportional Fair Algorithmus im Einsatz. Wenn ein Nutzer in einer Klasse einen sehr guten Kanal hat, kann es sein, dass er die gesamte Bandbreite dieser Klasse bekommt. Nach dieser Auswahl bekommen die unteren Klassen nur die Restefrequenzbereiche bekommen. Hierbei werden Nutzer aus höheren Klassen trotz ihres schlechten Kanals gegenüber Nutzern mit einem

besseren Kanal aus unteren Klassen bevorzugt Deswegen liegt der gesamte Systemdurchsatz unter dem gewichteten Priorisierung Algorithmus.

3.3.2 Pseudocode

Das folgende Listing enthält den Pseudocode des Algorithmus zur KBP:

FÜR JEDE Prioritätsklasse in Prioritätsklassen
FÜR JEDE Frequenz in allen freie Frequenzen
FÜR JEDEN Nutzer in aktueller Prioritätsklasse
// Für alle Nutzer/Frequenz-Kombinationen den Proportional Fair
// Algorithmus anwenden.
*PufferArray[Nutzer*Frequenz] = Proportional Fair Algorithmus (Nutzer, Frequenz)*
ENDE FÜR
PufferArray absteigend sortieren

// Anzahl der Frequenzen, die aktuelle Klasse maximal besetzen darf
*MaxFrequenzen = GesamtFrequenzen * P_{Klasse} + $\frac{P_j}{\sum_{i=j}^{n} Pi} * FreieFrequenzen$*

// Verteile alle für die Klasse reservierten Frequenzen an die
// Klassennutzer
VergebeneFrequenzen = 0
FÜR JEDEN Wert im PufferArray
WENN VergebeneFrequenzen > MaxFrequenzen DANN
Breche Schleife ab
ENDE WENN
WENN Frequenz frei DANN
Weise Frequenz dem Nutzer zu
VergebeneFrequenzen = VergebeneFrequenzen + 1
END WENN
ENDE FÜR

// Berechne ob noch freie Frequenzen für die nächste Klasse übrig sind
FreieFrequenzen = 0
WENN VergebeneFrequenzen < MaxFrequenzen DANN
FreieFrequenzen = MaxFrequenzen - VergebeneFreuqenzen
ENDE WENN
ENDFÜR JEDE
ENDFÜR JEDE

3.3.3 Erwartete Ergebnisse

Im Algorithmus zur KBP sollten die Ergebnisse ähnlich wie beim gewichtete Priorisierung Algorithmus aussehen. Die Durchsatzverhältnisse der Klassen bleiben auch bei höherer Last stabil und entsprechen den gegebenen Gewichtungen. Alle Klassen haben also auch bei hoher Last einen Anteil an der Bandbreite, der ihrer Gewichtung entspricht. Untere Klassen haben nie weniger als den garantierten Anteil. Innerhalb einer Klasse ist die Fairness zwischen den Nutzern allerdings nicht berücksichtigt. Deswegen sollte der gesamte Systemdurchsatz unter dem gewichteten Priorisierung Algorithmus liegen.

3.4 Algorithmus zur reservierten Nutzerbandbreit Priorisierung

3.4.1 Vorgehensweise des Algorithmus

Ähnlich wie beim Algorithmus zur KBP bekommen alle Prioritätsklassen im Algorithmus zur reservierten Nutzerbandbreit Priorisierung(NBP) einen Parameter, der den Anteil an der Bandbreite darstellt. Der Parameter begrenzt nicht nur die Bandbreite pro Klasse, sondern auch die Ressourcenverteilung innerhalb einer Klasse. In einem Frame findet die Ressourcenverteilung zwei Mal statt. Die erste Verteilung ist genau wie beim Algorithmus zur KBP. Für jeden Benutzer der ersten Klasse und jede Frequenzkombination wird der Proportional Fair Algorithmus ausgeführt. Jede Frequenz wird durch den Nutzer besetzt, der mit dieser Frequenz die höchste Sendeleistung hat. Dabei hat jeder Nutzer eine bestimmte Anzahl der Frequenzen bekommen. Von diesen zugewiesenen Frequenzen muss er nun so viele an die nächste, und nur diese, Klasse „abgeben“, wie allen nächsten Klassen zusteht. Ist der Nutzer z.B. in einer Klasse, die 75% der Bandbreite für sich beansprucht, so muss also jeder Nutzer 25% seiner Frequenzen an die nächste Klasse abgeben.

Zur Berechnung des Anteils der besetzbaren Frequenzen für jede Nutzer gilt folgende Funktion:

> *Finale Frequenzanzahl eines Nutzers = Anteil der Klasse an der Bandbreite* $*$ *Häufigkeit eines Nutzers in Frequenzauswahl* **/** *Anzahl aller Frequenzen*

Im diesen Verfahren wird die Fairness innerhalb einer Klasse berücksichtigt. Obwohl ein Nutzer in einer Klasse eine sehr gute Leistung bringen kann, darf er nicht alle Ressourcen in seiner Klasse besitzen. Bei niedriger Systemlast werden auch die Frequenzen, die für eine obere Klasse zu viel sind, an die nächste Klasse weitergegeben.

3.4.2 Pseudocode

Das folgende Listing enthält den Pseudocode des Algorithmus zur NBP:

FÜR JEDE Prioritätsklasse aller Prioritätsklassen
    FÜR JEDE Frequenz aller freien Frequenzen
        FÜR JEDEN Nutzer aller Nutzer der Prioritätsklasse
            Vergleiche Proportional Fair Algorithmus (Nutzer, Frequenz) mit bisher bestem Nutzer der aktuellen Frequenz und setze das bessere in ein zweidimensionales Array namens FrequenzNutzer mit den Frequenzen und NutzerIDs ein.
        ENDE FÜR
    ENDE FÜR

    FÜR JEDEN Nutzer der Prioritätsklasse
        Suche Nutzer in FrequenzNutzer-Array und speichere Treffer in neuem Array namens NutzerFrequenzen
        Sortiere Array absteigend nach Sendeleistung
        *AnzahlNutzerFrequenzen = Anteil der Prioritätsklasse * Länge(NutzerFrequenzen)*
        Weise Nutzer AnzahlNutzerFrequenzen Frequenzen zu.
    ENDE FÜR

    Weise alle freien Frequenzen der nächsten Klasse zu
ENDE FÜR

3.4.3 Erwartete Ergebnisse

Die Ergebnisse im Algorithmus zur NBP könnten ähnlich wie im Algorithmus zur KBP aussehen. Der Parameter jeder Klasse ist genau der Anteil der Klasse vom gesamten Systemdurchsatz. Der gesamte Systemdurchsatz muss geringer als den Durchsatz des Algorithmus zur KBP sein. Der Grund dafür ist, dass innerhalb einer Klasse die Bandbreit proportional zu den Nutzern zugeteilt wird. Der Nutzer, der immer einen guten SINR-Wert hat, könnte im diesen Fall nicht gewählt werden.

4 Implementierungen der Simulation

Im oberen Kapitel sind die verschiedenen Priorisierungsmechanismen erklärt und implementiert. Um die Ergebnisse des jeweiligen Algorithmus‘ zu analysieren und zu vergleichen, ist eine Simulationsumgebung erforderlich. Zur Implementierung der Mechanismen wurde die Software Simulation Library(SimLib) des Instituts für Kommunikationsnetze und Rechnersysteme der Universität Stuttgart eingesetzt. Die SimLib ist ein Tool, das hauptsächlich die Simulation im drahtlosen Netzbereich bedient. SimLib besteht aus drei Hauptteilen: Simulation von Unterstützungssystemen, Simulation Kontrollsystemen und einem Ergebnisauswertungssystem[15].

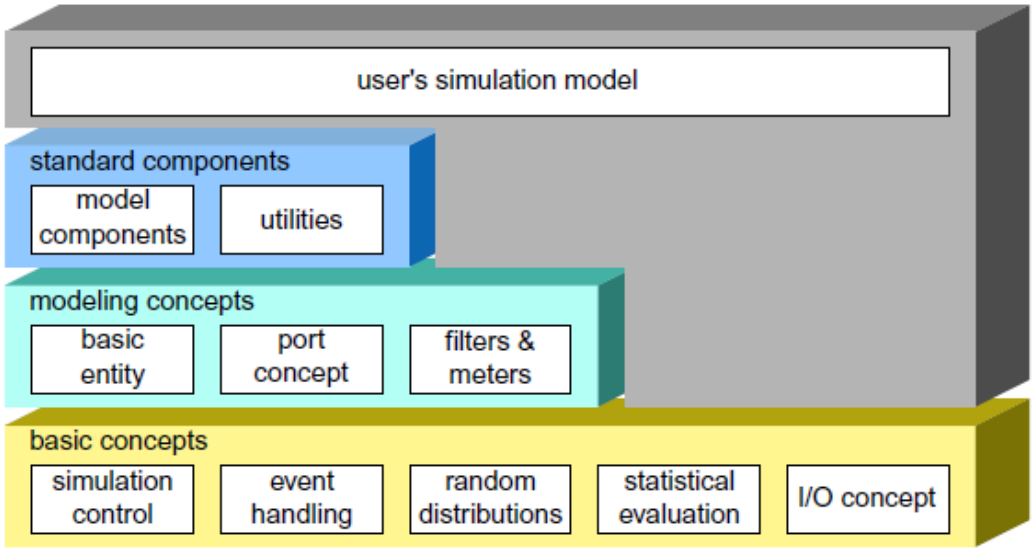

Abbildung 4.1 Struktur der IKR SimLib[15]

In dieser Arbeit soll ein Paket „Scheduler“ in SimLib erweitert werden. Der „Scheduler“ entscheidet, welche Ressourcen zu welchem Zeitpunkt einem bestimmten Nutzer zugeteilt werden. Im Scheduler wurden drei verschiedene QoS Priorisierungsverfahren implementiert, die statische Priorisierung, die gewichtete Priorisierung und die Priorisierung mit verteilter Bandbreite.

4.1 Simulationsumgebungen

Um die Simulationsergebnisse besser mit der theoretischen Analyse vergleichen zu können wurde der Algorithmus bei der Simulation mehrmals mit unterschiedlichen Parametern ausgeführt. Im SimLib sind die Parameter durch eine Datei eingegeben worden. Im Folgenden ist beispielsweise ein Teil der Datei *„sim.par“* für den gewichteter Priorisierung:

```
TimePerBatch = 10;
TransientPhaseTime = 1;
NoOfRBs = 50;
NoOfOscillators = 32;
DopplerShift = 10;
InterSiteDistance = 1000;
SchedulingInterval = 0.001;
PFScheduler{
    ForgettingFactor = 0.02;
    AdaptiveForgettingFactor = true;
    LogMovingAverages = "ma.log";
    WeightSelectionMechanism{
        AddPriorityClass = 10;
        AddPriorityClass = 10;
        AddPriorityClass = 10;

        AddClassWeight = 10;
        AddClassWeight = 5;
        AddClassWeight = 2;
    }
    NoOfUsers = 30;
    IATDist{ NegExp{Mean=%%Mean%%;} }
    LengthDist{DiscreteConstant{Mean=1000;}}
    StaticHandover;
```

In diesem Beispiel wurden drei Klassen mit jeweils zehn Nutzern definiert. Die erste Klasse hat einen Gewichtsfaktor von 10, die zweite Klasse hat 5 und letzte Klasse einen Gewichtsfaktor von 2. Der Parameter „*Mean“* wird als Simulationsparameter genommen. In der Simulation wird der Simulationsparameter mit verschiedenen Werten belegt. Nach der Simulation werden die Ergebnisse zusammengefasst. Die Leistung jeder Klasse wird auch in verschiedenen Werten verglichen.

4.2 Messgrößen

Zur Ergebnisanalyse wurden einige Messgrößen und Begriffe verwendet. Dabei bezeichnet der Durchsatz (engl. *Throughput*) die Menge, die innerhalb eines festgelegten Zeitraums durch eine vorher definierte Grenze verarbeitet oder übertragen wird[16].

Konfidenzintervall

Das Konfidenzintervall ist das Ergebnis von Intervallschätzungen. Das Konfidenzniveau ist die Wahrscheinlichkeit, mit der θ vom Konfidenzintervall überdeckt wird. Ein Konfidenzintervall für den Parameter θ zum Konfidenzniveau 1-α wird aus den Daten derart berechnet, dass in mindestens (1-α) ∗ 100% der Fälle der Parameter θ tatsächlich überdeckt wird[17]. Bei normalverteilter Grundgesamtheit mit unbekannter Varianz wird das Konfidenzintervall für den Erwartungswert angegeben als[19]:

$$\left[\bar{x} - t_{\left(1-\frac{\alpha}{2};n-1\right)}\frac{s}{\sqrt{n}}\ ;\ \bar{x} + t_{\left(1-\frac{\alpha}{2};n-1\right)}\frac{s}{\sqrt{n}}\right] \quad [19]$$

Dabei ist z(1-α/2, n-1) das 1−α/2-Quantil der Standardnormalverteilung.

Cumulative distribution function (CDF)

In der Wahrscheinlichkeitstheorie beschreibt die CDF die Wahrscheinlichkeit, die eine Zufallsvariable X mit einer bestimmten Wahrscheinlichkeitsverteilung den Werte weniger oder gleich X hat. Die CDF Funktion erhält man als Integral über die Dichtefunktion:

$$F(x) = \int_{-\infty}^{x} f(t)\,\mathrm{d}t \quad [20]$$

Fairnessbetrachtung

Im White Paper namens *Next Generation Mobile Networks Radio Access Performance Evaluation Methodology* von *Next Generation Mobile Networks* (NGMN) ist die Metrik der Fairness so festgelegt:

Im CDF Graph ist eine Grenzlinie definiert. Die Kurve, die den Durchsatz veranschaulicht, muss auf der rechten Seite der Grenzlinien liegen, damit die Verteilung fair ist[18].

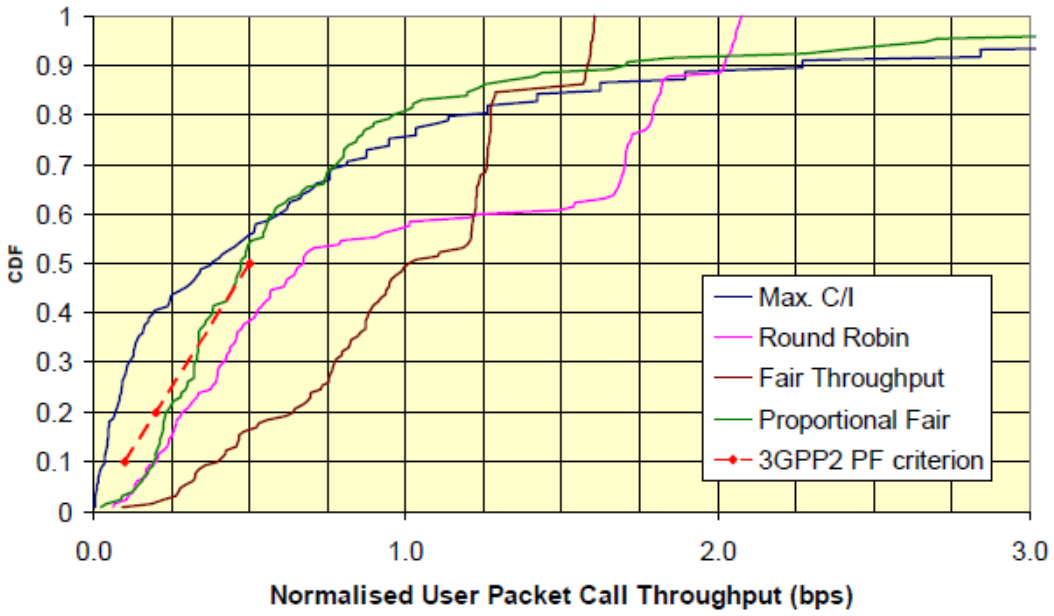

Abbildung 4.2 Beispiel für *Cumulative Distribution Function* (CDF) auf normalisierten Durchsätze

Die Grenzlinien werden aus folgenden Punkten definiert:

Normalized Throughput	CDF
0.1	0.1
0.2	0.2
0.5	0.5

Tabelle 4.1 Grenzlinien des Durchsatzes [18]

Jain's fairness index

Jain's fairness Index ist eine Methode zur Fairnessbewertung. Die Funktion ist:

$$\mathcal{J}(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2}$$ [21]

Diese Funktion stellt die Fairness mit n Nutzern dar. Daher steht x_i für den Durchsatz für die i-te Verbindung. Das Ergebnis liegt zwischen 1/n (schlechtester Fall) und 1(bester Fall). Die Fairness wird mit 1 bewertet, nur wenn alle Nutzer gleiche Ressourcen bekommen haben. [21]

Anhand der obengenannten Messgrößen werden in dem nächsten Kapitel die Simulationsergebnisse analysiert und die Messgrößen aus verschiedenen Algorithmen verglichen.

5 Simulationsergebnisse

In diesem Kapital werden alle Simulationsergebnisse der obengenannten Algorithmen analysiert und miteinander verglichen. Daher werden der Systemdurchsatz, die Fairness zwischen verschiedenen Klassen und die Durchsätze aller Klassen berücksichtigt. Der Unterschied zwischen verschiedenen Algorithmen wird ebenfalls analysiert.

5.1 Ergebnisse aus Statischer Priorisierung

5.1.1 Szenario

Die Simulation ist mit folgenden Parametern durchgeführt:

	Anzahl der Nutzer
Prioritätsklasse 0	10
Prioritätsklasse 1	10
Prioritätsklasse 2	10

Tabelle 5.1 Parameter der statischer Priorisierung

Das System hat insgesamt 30 Nutzer. Die Nutzer sind in drei Gruppen unterteilt. Jede Gruppe hat genau 10 Nutzer. Außerdem ist die Anzahl der Frequenzen auf 50 festgesetzt. Die Paketgröße ist 1000 Bit. Und die Ankunftsfrequenz der Pakete wird mit dem Parameter *Mean* gesteuert. In der Simulation werden die Werte der Ankunftsfrequenz von 0.0001 bis 1 gleichmäßig im SimTree Parameterfile vorgegeben d.h. die Simulation wurde zwischen der minimalen Systemlast von 10^3 Bit/Sekunde und der maximalen Systemlast von 10^7 Bit/Sekunde durchgeführt.

5.1.2 Durchsatz

Die Abbildung 5.1 zeigt die Durchsätze für jede Klasse. Die schwarze Kurve beschreibt den gesamten Durchsatz des Systems. Die X-Achse ist 1/IAT, nämlich 1/Mean. Die Werte in X-Achse stellen genau dar, wie viele Pakete pro Sekunde zur Übertragung angekommen sind. (Jedes Paket hat die Größe von 10^3 Bit.) Die Y-Achse ist der Durchsatz in Bit/Sekunde.

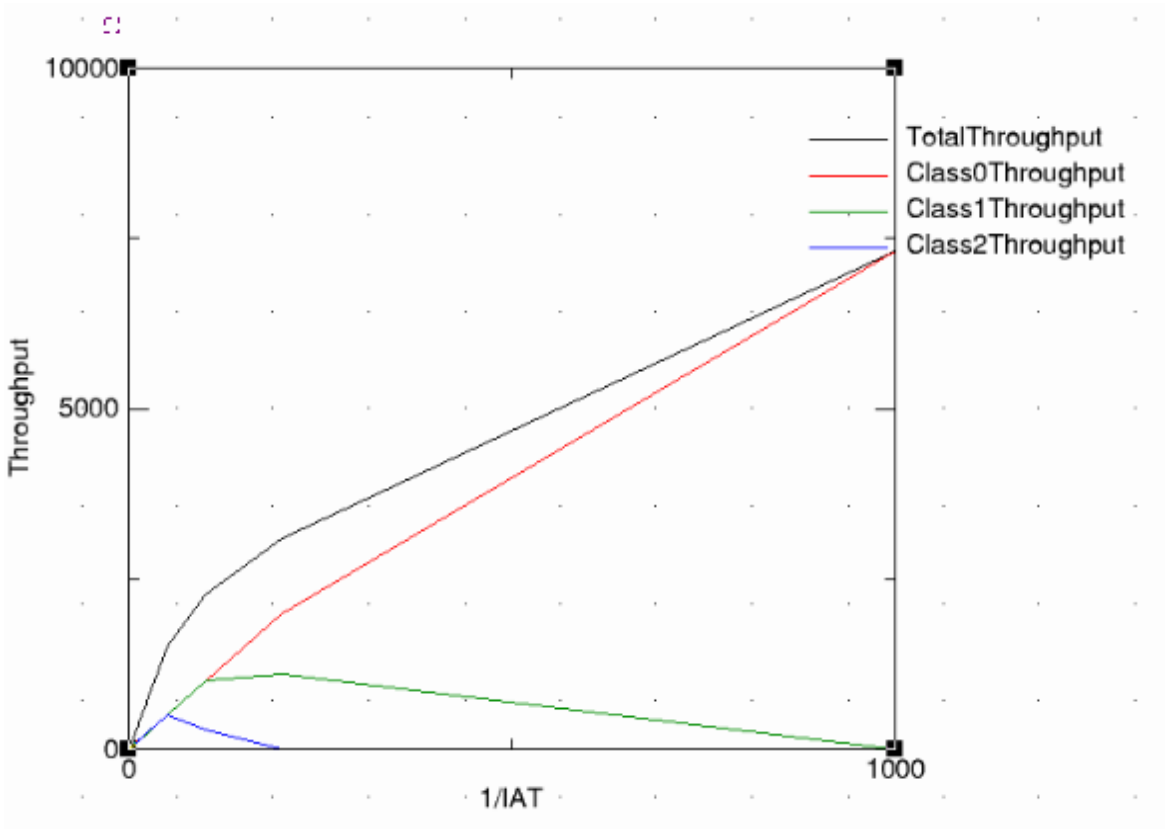

Abbildung 5.1 Durchsatz des Algorithmus zur statischen Priorisierung bis 1/IAT=1000

Aus der Abbildung 5.1 wird verdeutlicht, dass am Anfang alle Klassen einen gleichen Anteil an Ressourcen bekommen und deswegen gleich gut funktionieren. Ab einem bestimmten 1/IAT sinkt der Durchsatz der untersten Klasse und entfernt sich von den anderen beiden. Bei weiter zunehmender Systemlast verlieren alle Klassen, die keine höchste Priorität haben, langsam ihren Durchsatz. Ab dem Zeitpunkt, an dem die vorletzte Prioritätsklasse keine Ressourcen bekommt (1/IAT=1000), sind alle Ressourcen von der obersten Klasse besetzt. In der Abbildung 5.1 ist dies auch gezeigt. Ab 1/IAT=1000 überlappen sich der gesamte Durchsatz (schwarze Kurve) und der Durchsatz in Klasse 0 (rote Kurve).

Der Grund für das Abnehmen der Durchsätze liegt in der höher werdenden Systemlast. Mit zunehmender Systemlast beanspruchen die höheren Klassen immer der Gesamtbandbreite, so dass für die unteren Klassen immer weniger freie Ressourcen übrigbleiben. Ab einem bestimmten Grenzwert beträgt der Durchsatz jeder Klasse, die nicht die erste ist, 0 - die Übertragung wird eingestellt. Dieser Grenzwert liegt für die unterste Klasse bei ca. 1/IAT=250.

Die Abbildung 5.2 zeigt ebenfalls die Durchsätze in der statischen Priorisierung. In dieser Abbildung wurden die Durchsätze für jede Klasse und der gesamte Durchsatz mit höherer Systemlast simuliert. Die Achsen haben die gleiche Bedeutung wie in Abbildung 5.1.

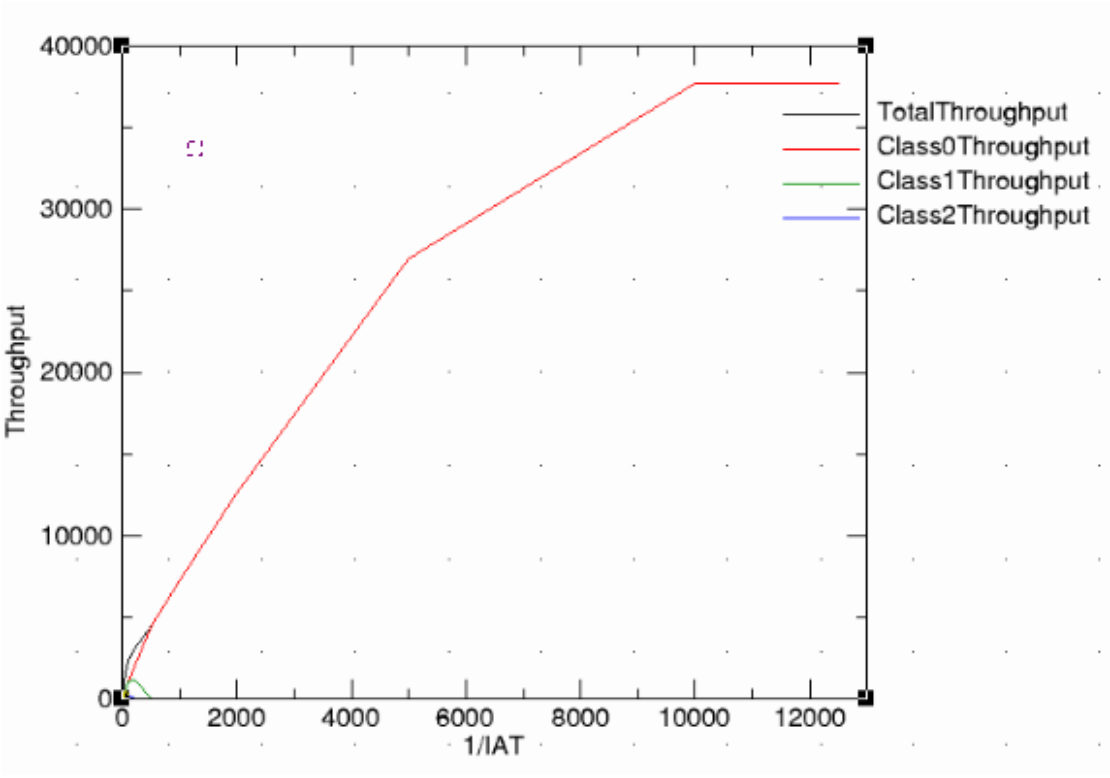

Abbildung 5.2 Durchsatz des Algorithmus zur statischen Priorisierung bis 1/IAT=13000

Ab 1/IAT=12000 stagniert der Gesamtdurchsatz auf dem Wert von ca. 38000 Bit/Sekunde, weil das System seine maximale Auslastung erreicht hat.

5.1.3 Fairness

Die folgende Abbildung zeigt die Kurve in verschiedenen IATs mit *Cumulative distribution function* (CDF).

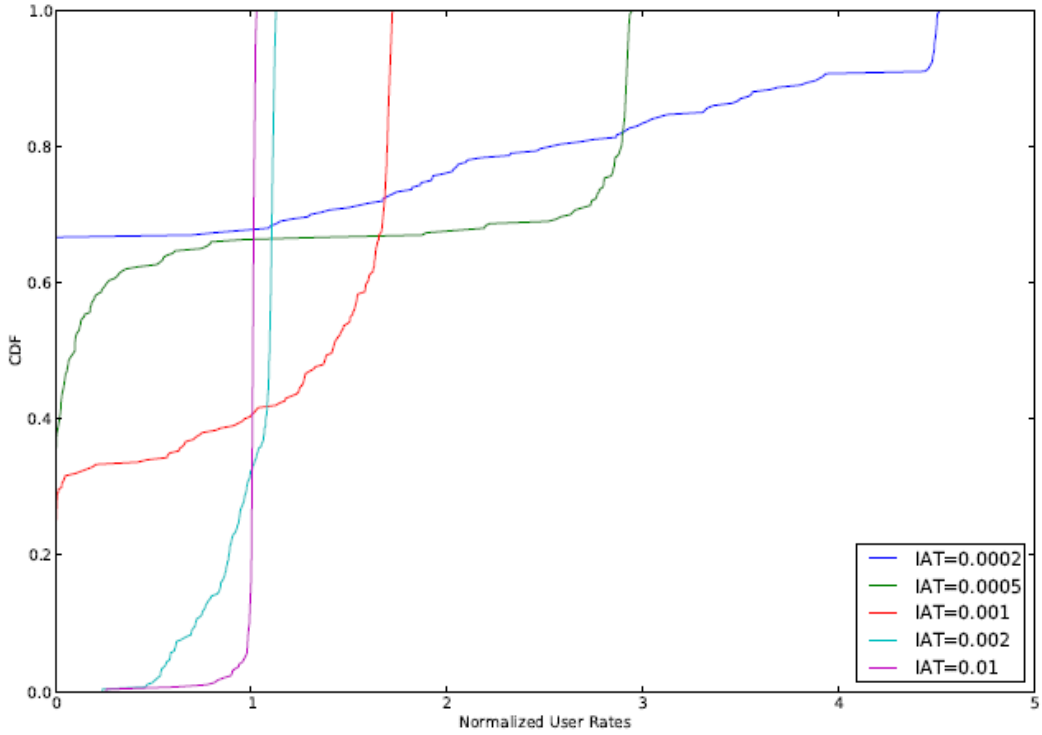

Abbildung 5.3 CDF des Algorithmus zur statischen Priorisierung Algorithmus

In diesem Modell sind Nutzer in drei Klassen geteilt. Jede Klasse beinhaltet genau 1/3 aller Nutzer. In der Y-Achse entsprechen die Werte zwischen 0 und 0.33 den Nutzern in der untersten Klasse. 0.33 bis 0.66 markiert die Nutzer aus der mittleren Klasse. Ab 0.66 sind die Nutzer mit der höchsten Priorität gezeigt. Die X-Achse ist der normalisierte Durchsatz.

Bei IAT=0.01 und IAT=0.002 ist das System nur leicht belastet. Und wie in Abbildung 5.3 dargestellt, sind die violette und die hellblaue Kurvengerade senkrecht nach oben gestiegen. d.h unter diesen IATs haben alle Nutzer gleiche Ressourcen bekommen. Bei IAT=0.001 fängt die rote Kurve an der Y-Achse mit dem Wert von ca. 0.3 an. Dies zeigt, dass ungefähr 1/3 aller Nutzer keine Ressourcen in diesem IAT bekommen haben. Aus Abbildung 5.3 ist weiterhin ersichtlich, dass der Durchsatz der Klasse 2 ab 1/IAT=1000 null geworden ist. Die grüne und blaue Kurve stellt die Ergebnisse mit höchster Systemlast dar. Obwohl die grüne Kurve mit einem Wert von 0.33 anfängt, ist der Durchsatz der zweiten Klasse sehr gering, fast alle Ressourcen sind von erster Klasse besetzt. Und die blaue Kurve ist von Anfang an nur im Bereich der Klasse 0. Dies zeigt, dass ab diesem IAT nur die erste Klasse von dem Scheduler bedient wird.

Als nächstes wird die Fairness zwischen den Prioritätsklassen betrachtet. Im statischen Priorisierung Algorithmus ist die Fairness nur schlecht erfüllt. Bei großer Last haben einige Klasse keine Ressourcen mehr. In der QoS-Differenzierung kann nur ein Datenmodell bei großer Last bedient werden. Die anderen Daten sind in diesem Fall verhungert.

In folgende Tabelle ist der Jain's fairness index. Dieser kann die Fairness in unterschiedlichen IATs mathematisch darstellen.

IAT	Jain's fairness index
0.0002	0.17
0.0005	0.24
0.001	0.37
0.002	0,64
0.01	0.99

Tabelle 5.2 Jain's fairness index der statischer Priorisierung

5.2 Ergebnisse aus gewichteter Priorisierung

5.2.1 Szenario

Die Simulation ist mit folgenden Parametern durchgeführt:

	Anzahl der Nutzer	Gewichtsfaktor
Prioritätsklasse 0	10	10
Prioritätsklasse 1	10	5
Prioritätsklasse 2	10	2

Tabelle 5.3 Parameter der gewichteter Priorisierung

Alle 30 Nutzer sind in drei Klassen geteilt. Jede Klasse hat 10 Nutzer. Die erste Klasse hat den größten Gewichtsfaktor 10. Der Gewichtsfaktor in Klasse 1 ist nur halb so groß wie in Klasse 0. Und die Klasse 3 hat die niedrigste Priorität mit einem Gewichtsfaktor von 2. Die Bandbreite ist auf 50 PRBs festgelegt. Die Paketgröße liegt bei 1000 Bit. Es werden die Ergebnisse zwischen IAT=0.0001 und IAT=1 analysiert.

5.2.2 Durchsatz

Die folgende Abbildung beschreibt den Durchsatz bei gewichteter Priorisierung. Die Y-Achse ist die Durchsatzachse. Die X-Achse bezeichnet 1/IAT. Die schwarze Kurve stellt den gesamten Durchsatz dar. Die anderen farbigen Kurven entsprechen den jeweiligen Klassen.

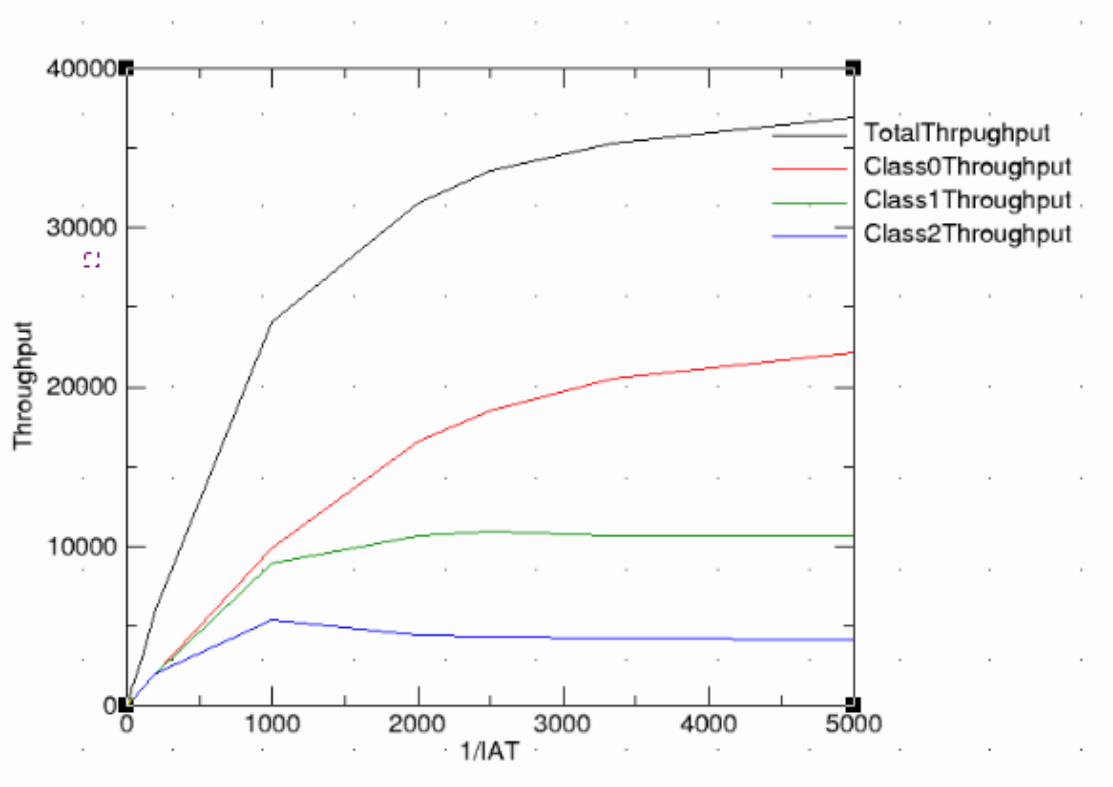

Abbildung 5.4 Durchsatz des Algorithmus zur gewichteten Priorisierung

Bei sehr wenig Verkehr haben alle Klassen einen gleichen Anteil an Ressourcen bekommen. Zwischen den Werten 0 bis ca. 200 auf der X-Achse überlappen sich die frei farbigen Kurven. Wenn die Systemlast weiter hoch geht, spielt der Gewichtsfaktor eine Rolle. Ab 1/IAT=200 bekommt die erste Klasse immer mehr Ressourcen. Die Klasse 3 hat bei 1/IAT=1000 ihre Spitzenwert erreicht und fällt dann im Durchsatz ab. Mit weiter steigender Last bleibt die Klassenverteilung nahezu konstant. In diesem Fall sind die Ressourcen allen Klassen zugeteilt. d.h. jede Klasse bekommt garantiert eine bestimmte Leistung, die durch den Gewichtsfaktor festgelegt wurde. Bei starker Last beträgt der Durchsatz der ersten Klasse ca. 20000 Bit/Sekunde. Die zweite Klasse ist bei ca. 10000 Bit/Sekunde Durchsatz geblieben. Und die letzte Klasse bekommt maximal ca. 400 Bit/Sekunde. Das Verhältnis des Durchsatzes jeder Klassen stimmt genau mit dem Verhältnis ihres Gewichtsfaktors überein.

5.2.3 Fairness

Die unter stehende Abbildung plottet die Kurve in verschiedene IATs mit *Cumulative distribution function* (CDF).

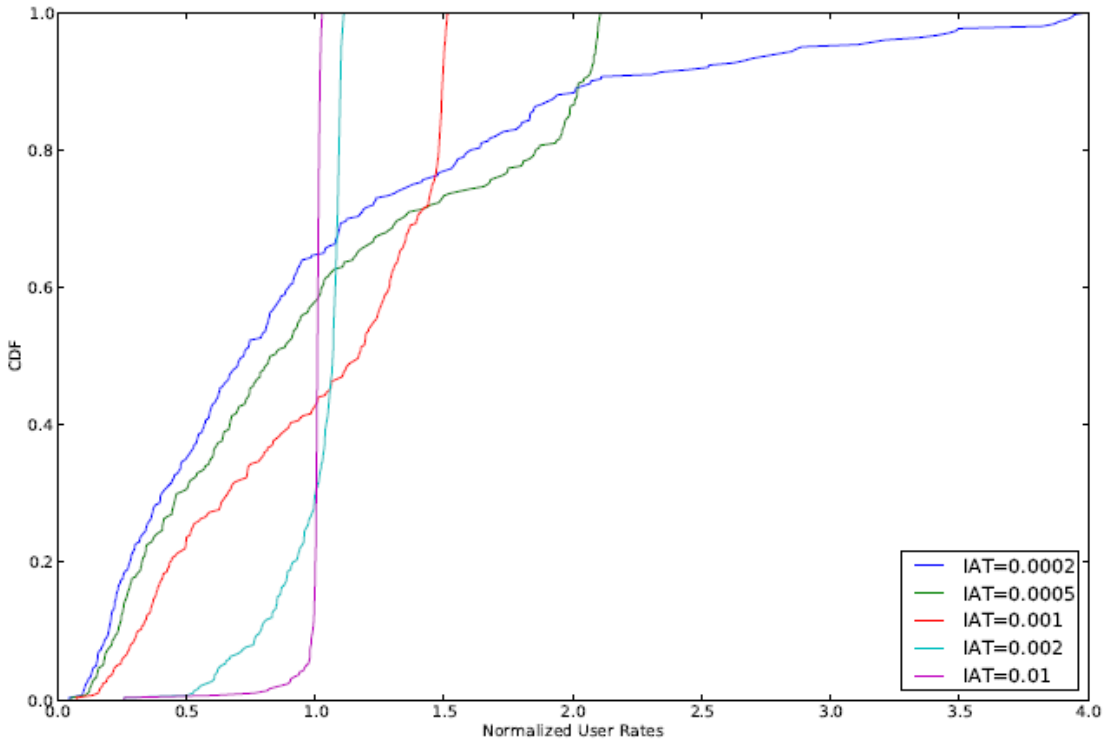

Abbildung 5.5 CDF des Algorithmus zur gewichteten Priorisierung

In diesem Modell sind Nutzer in drei Klassen geteilt. Jede Klasse beinhaltet genau 1/3 aller Nutzer. In der Y-Achse entsprechen die Werte zwischen 0 und 0.33 den Nutzern in der untersten Klasse. 0.33 bis 0.66 markiert die Nutzer aus der mittleren Klasse. Ab 0.66 sind die Nutzer mit der höchsten Priorität gezeigt. Die X-Achse ist der normalisierte Durchsatz.

Bei sehr wenig Verkehr (IAT=0.01, und 0.002) sind die violette Kurve und die hellblaue Kurve gerade senkrecht nach oben gestiegen. d.h Die Ressourcen sind fast gleichmäßig zu allen Klassen zugeteilt. Bei IAT= 0.001 und 0.0005 ist die Systemlast größer geworden. In diesem Fall sind die Durchsätze der zwei unteren Klassen weiter gesunken. Im Vergleich zur statischen Priorisierung bekommen die unteren Klassen einiges an Bandbreite. Bei extrem größer Last (IAT=0.00002) ist der maximale Systemdurchsatz erreicht. Die blaue Kurve fängt wieder vom Nullpunkt an. Alle Klassen bekommen Ressourcen.

In Tabelle 5.4 ist der Jain's fairness index. Dieser kann die Fairness in unterschiedlichen IATs mathematisch darstellen.

IAT	Jain's fairness index
0.0002	0.62
0.0005	0.67
0.001	0.71
0.002	0,82
0.01	0.99

Tabelle 5.4 Jain's fairness index der gewichteter Priorisierung

5.3 Ergebnisse aus Bandbreitenreservierung pro Klasse

5.3.1 Szenario

Die Simulationsparameter für KBP sehen wie folgt aus:

	Anzahl der Nutzer	Bandbreitenanteil
Prioritätsklasse 0	10	0.75
Prioritätsklasse 1	10	0.2
Prioritätsklasse 2	10	0.05

Tabelle 5.5 Parameter der Algorithmus zur KBP

Alle 30 Nutzer sind in drei Klassen geteilt. Jede Klasse hat 10 Nutzer. Es steht 50 PRBs zur Verfügung. Der Bandbreitenanteil begrenzt, wie viel Prozent von allen Frequenzen diese Klasse besetzten kann. Die erste Klasse hat die höchste Priorität. Diese Klasse darf maximal 75% aller Ressourcen besetzen. Für die Klasse 1 ist die maximale Anzahl der Bandbreite auf 20% begrenzt. Die niedrigste Klasse 2 kann die restlichen 5% übernehmen. Die Pakete haben eine Größe von 1000 Bit. In der Simulation wird eine Systemlast zwischen 10^3 Bit/Sekunde und 10^7 Bit/Sekunde simuliert.

5.3.2 Durchsatz

Die Abbildung 5.6 beschreibt den Durchsatz bei KBP. Die Y-Achse ist die Durchsatzachse. Die X-Achse bezeichnet 1/IAT. Die schwarze Kurve stellt den gesamten Durchsatz dar. Die anderen farbigen Kurven entsprechen den jeweiligen Klassen.

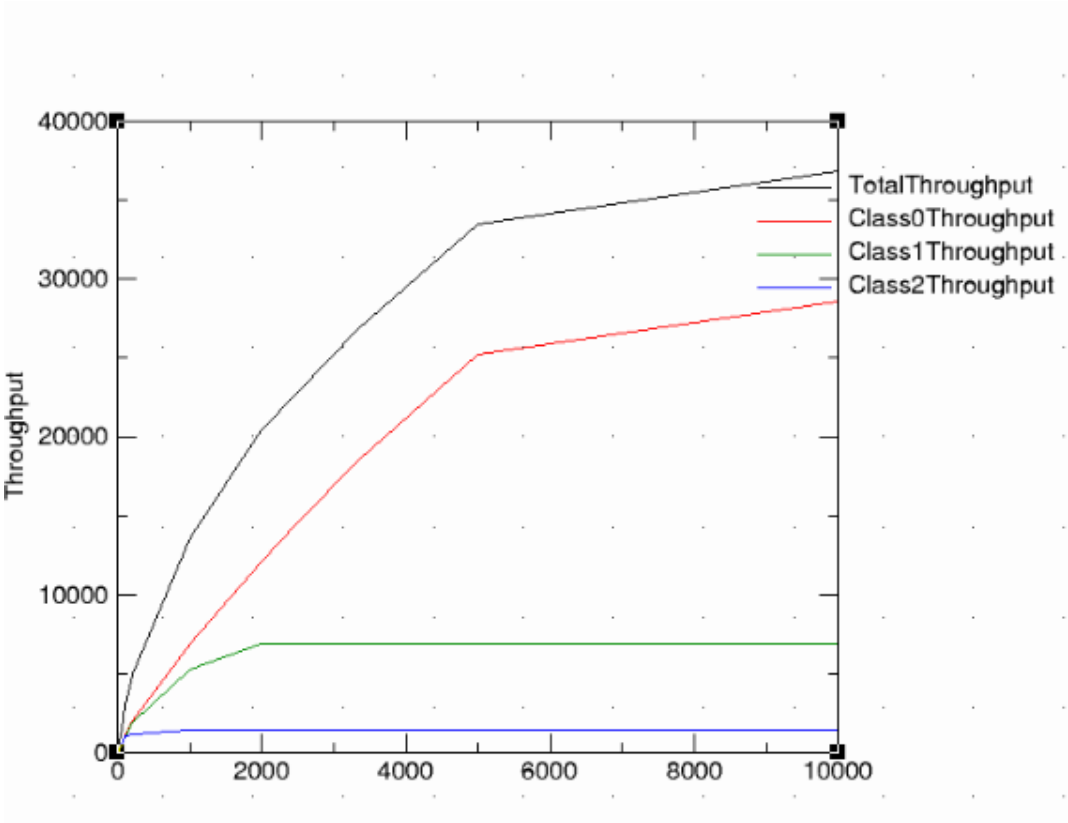

Abbildung 5.6 Durchsatz des Algorithmus zur KBP

Auf X-Achse von 0 bis ca. 150 überlappen sich die drei farbigen Kurven. Dies bedeutet, dass bei sehr geringem Verkehr alle Klassen den gleichen Ressourcenanteil bekommen. Mit steigender Systemlast hat die Klasse 2 zuerst ihren maximalen Ressourcenanteil erreicht. Ab 1/IAT=250 bleibt ihr Durchsatz fast konstant. Die Klasse 1 erreicht ihren maximalen Durchsatz ab 1/IAT=2000. Weil die erste Klasse am meisten Bandbreite besetzen kann, bleibt der gesamte Durchsatz konstant, wenn ihr maximaler Durchsatz erreicht ist. Ab 1/IAT=10000 hat die erste Klasse auch ihren maximalen Durchsatz erreicht. Die Verhältnisse zwischen der tatsächlichen gemessenen Klassebandbreite und den vorher festgelegten Anteilen stimmen überein.

In der KBP sind die Durchsätze aller Klassen durch den festgelegten Bandbreiteanteil garantiert. Jede Klasse bekommt einen garantierten Anteil der Ressourcen. Und das ganze System erreicht sein maximalen Durchsatz ab 1/IAT=10000.

5.3.3 Fairness

Die unter stehende Abbildung zeigt die Kurve mit verschiedenen IATs mit *Cumulative distribution function* (CDF).

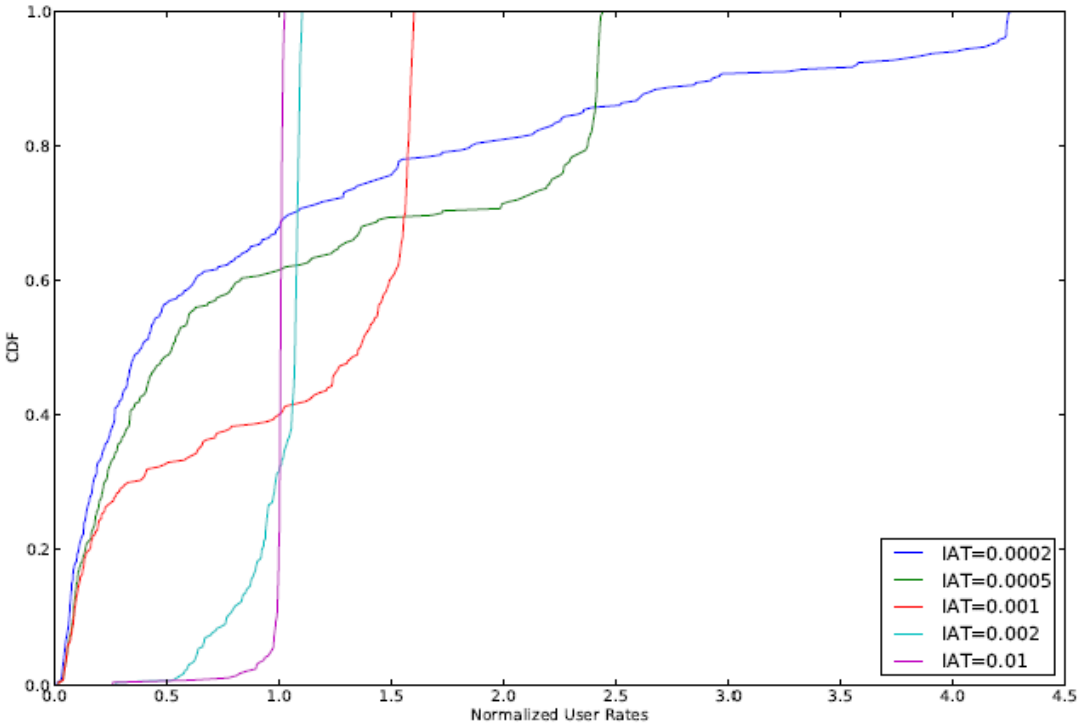

Abbildung 5.7 CDF des Algorithmus zur KBP

In diesem Modell sind Nutzer in drei Klassen geteilt. Jede Klasse beinhaltet genau 1/3 aller Nutzer. In der Y-Achse entsprechen die Werte zwischen 0 und 0.33 den Nutzern in der untersten Klasse. 0.33 bis 0.66 markiert die Nutzer aus der mittleren Klasse. Ab 0.66 sind die Nutzer mit der höchsten Priorität gezeigt. Die X-Achse ist der normalisierte Durchsatz.

Bei sehr geringem Verkehr (IAT=0.01, und 0.002) sind die violette und die hellblaue Kurve gerade senkrecht nach oben gestiegen. Dies bedeutet, dass die Ressourcen fast gleichmäßig allen Klassen zugeteilt wurden. Bei IAT= 0.001 und 0.0005 ist die Systemlast größer geworden. In diesem Fall sind die Durchsätze für die beiden unteren Klassen gesunken. Gleichzeitig ist aber der Durchsatz der ersten Klasse deutlich größer im Vergleich zur geringeren Systemlast. Bei extrem großer Last (IAT=0.00002) ist der maximale Systemdurchsatz erreicht. Die blaue Kurve fängt wieder am Nullpunkt an. Der Durchsatz der obersten Klasse hat sich weiter vergrößert. Im Vergleich zur statischen Priorisierung bekommen alle Klassen Ressourcen.

In der Tabelle 5.6 ist der Jain's fairness index der verschiedenen IATs gezeigt. Der Index kann auch die Fairness in verschiedenen IATs mathematisch beschreiben.

IAT	Jain's fairness index
0.0002	0.44
0.0005	0.56
0.001	0.65
0.002	0,77
0.01	0.99

Tabelle 5.6 Jain's fairness index der KBP

5.4 Ergebnisse aus Algorithmus zur reservierten Nutzerbandbreit Priorisierung

5.4.1 Szenario

Die Simulationsparameter für NBP sehen wie folgt aus:

	Anzahl der Nutzer	Bandbreitenanteil
Prioritätsklasse 0	10	0.75
Prioritätsklasse 1	10	0.2
Prioritätsklasse 2	10	0.05

Tabelle 5.7 Parameter des Algorithmus zur NBP

Alle 30 Nutzer sind in drei Klassen unterteilt. Jede Klasse hat genau 10 Nutzer. In jedem Frame stehen 50 PRBs zur Verfügung. Der Bandbreitenanteil begrenzt, wie viel Prozent der Gesamtbandbreite diese Klasse benutzen darf. Der Bandbreitenanteil begrenzt auch die maximale Bandbreite für die Nutzer innerhalb einer Klasse. Die erste Klasse hat die höchste Priorität. Diese Klasse darf maximal 75% aller Ressourcen besetzen. Für die Klasse 1 ist die maximale Anzahl der Bandbreite auf 20% begrenzt. Die niedrigste Klasse 2 kann den Rest von 5% übernehmen. Die Pakete haben eine Größe von 1000 Bit. In der Simulation wird die Systemlast zwischen 10^3 Bit/Sekunde und 10^7 Bit/Sekunde simuliert. Das System wird mit Padding simuliert.

5.4.2 Durchsatz

Die folgende Abbildung beschreibt den Durchsatz bei gewichteter Priorisierung. Die Y-Achse ist die Durchsatzachse. Die X-Achse bezeichnet 1/IAT. Die schwarze Kurve stellt den gesamten Durchsatz dar. Die anderen farbigen Kurven entsprechen den

jeweiligen Klassen.

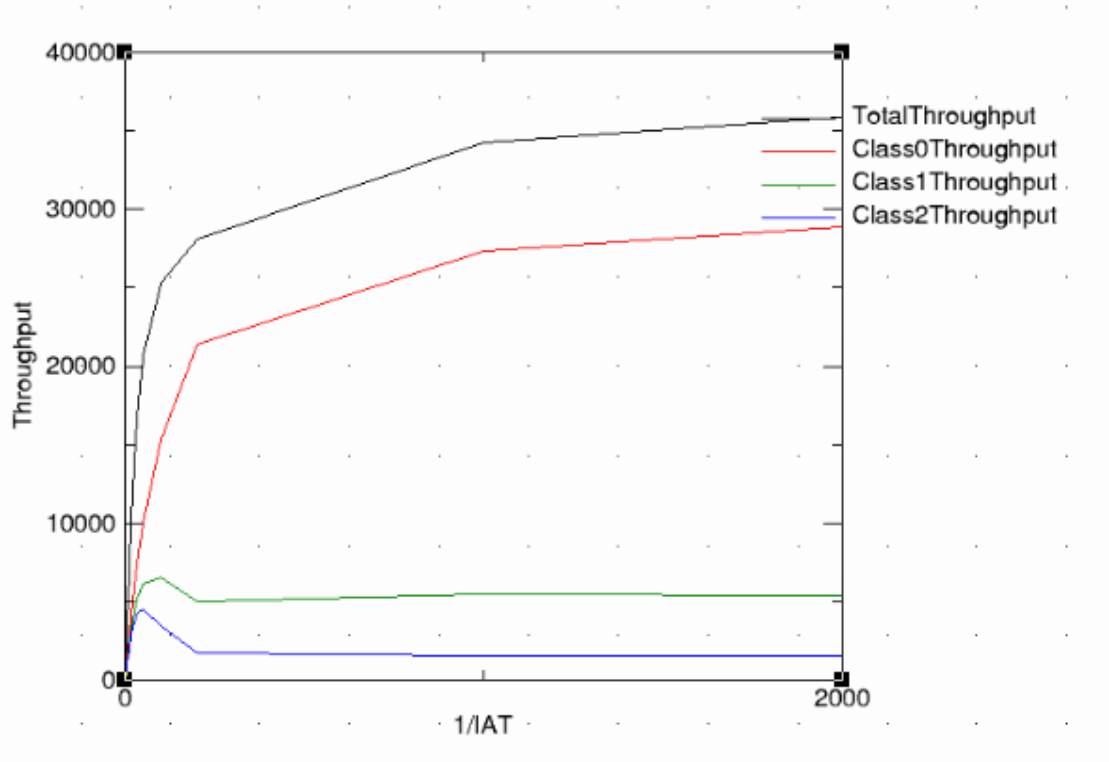

Abbildung 5.8 Durchsatz des Algorithmus zur NBP

Wie in der Abbildung 5.8 zu sehen, bekommen bei sehr geringem Verkehr alle Klassen Ressourcen zugeteilt. Mit steigender Systemlast erreichen Klasse 1 und Klasse 2 ihre Spitzenwert und sinken dann wieder etwas, bis sie ab 1/IAT=300 konstant bleiben. Ab 1/IAT=2000 hat die erste Klasse ihren maximalen Durchsatz erreicht. Die Verhältnisse zwischen der tatsächlichen gemessenen Klassebandbreite und den vorher festgelegten Anteilen stimmen überein.

In der NBP sind die Durchsätze aller Klassen durch den festgelegten Bandbreiteanteil garantiert. Jede Klasse bekommt einen garantierten Anteil der Ressourcen. Und das ganze System erreicht sein maximalen Durchsatz ab 1/IAT=2000.

5.4.3 Fairness

Die folgende Abbildung zeigt die Kurve in verschiedenen IATs mit *Cumulative distribution function* (CDF).

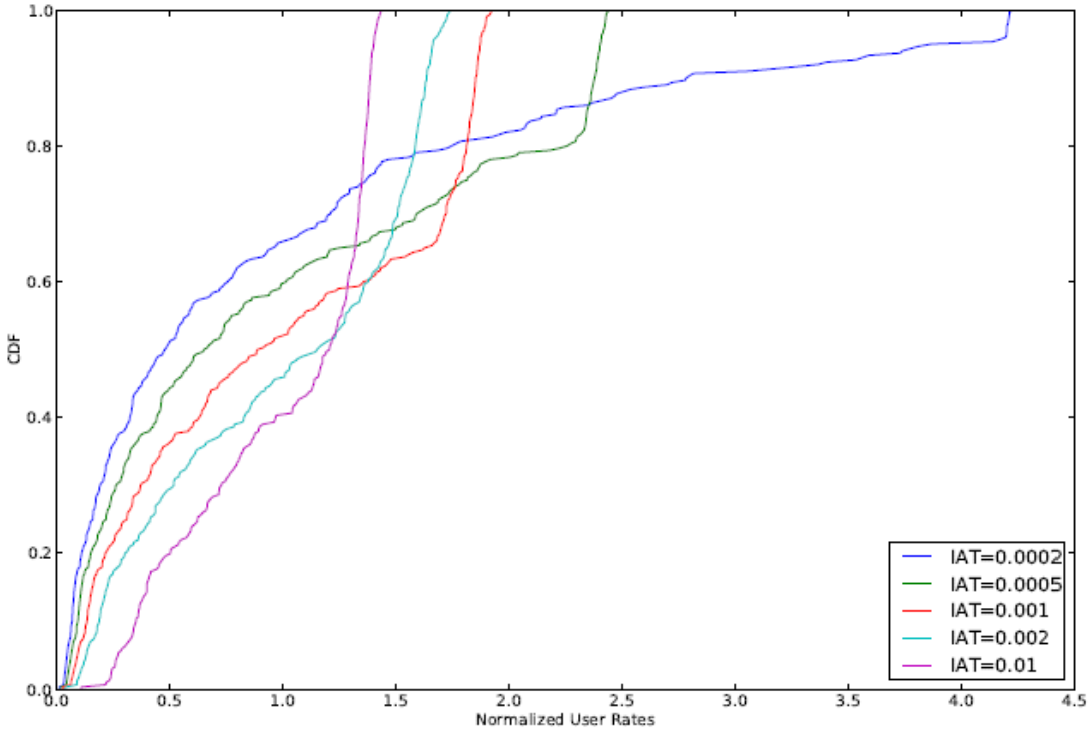

Abbildung 5.9 CDF des Algorithmus zur NBP

In diesem Modell sind Nutzer in drei Klassen geteilt. Jede Klasse beinhaltet genau 1/3 aller Nutzer. In der Y-Achse entsprechen die Werte zwischen 0 und 0.33 den Nutzern in der untersten Klasse. 0.33 bis 0.66 markiert die Nutzer aus der mittleren Klasse. Ab 0.66 sind die Nutzer mit der höchsten Priorität gezeigt. Die X-Achse ist der normalisierte Durchsatz.

Der Unterschied zwischen verschiedenen Systemlasten ist in diesem Algorithmus nicht deutlich. Sogar bei sehr leichter Systemlast sind die Ressourcen nicht gleichmäßig aufgeteilt. Mit steigender Systemlast bekommt die untere Klasse immer weniger Ressourcen. Eine so abrupte Änderung der Klassenzuteilung wie bei anderen Algorithmen ist hier nicht erkennbar. Bei extrem großer Last (IAT=0.00002) hat der Systemdurchsatz den maximalen Wert erreicht. Auch hier fängt die blaue Kurve am Nullpunkt an. Der Durchsatz für die oberste Klasse ist mit der Systemlast immer größer geworden.

In der Tabelle 5.8 ist der Jain's fairness index in verschiedenen IATs gezeigt.

IAT	Jain's fairness index
0.0002	0.41
0.0005	0.53
0.001	0.64
0.002	0,80
0.01	0.99

Tabelle 5.8 Jain's fairness index der NBP

5.5 Die drei Algorithmen zum Vergleich

Im diesen Kapital werden die Algorithmen für die statische Priorisierung, gewichtete Priorisierung und die KBP verglichen. Daher werden der gesamte Systemdurchsatz, die Fairness und der Durchsatz in der obersten Klasse analysiert.

5.5.1 Der gesamte Durchsatz

Die Abbildung 5.10 beschreibt den Durchsatz in den obengenannten Priorisierungen. Die Y-Achse ist die Durchsatzachse. Die X-Achse bezeichnet die 1/IAT. Die Pakete haben eine Größe von 10^3 Bit. Jede Kurve bezeichnet den gesamten Durchsatz eines Algorithmus.

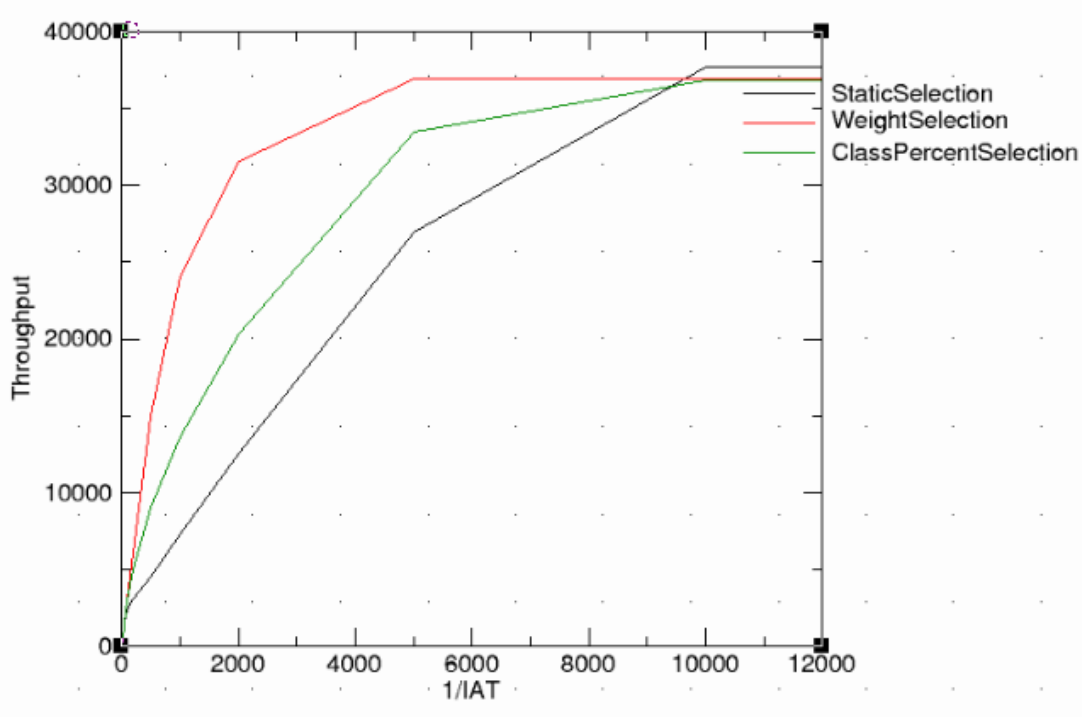

Abbildung 5.10 Gesamter Durchsatz mit verschiedenen Priorisierungen

Aus Abbildung 5.10 kann man sehen, dass bei verschiedener Systemlast die Durchsätze unterschiedlich aussehen. Bei kleinem 1/IAT sind kaum Unterschiede zu sehen. Die statische Priorisierung hat den schlechtesten Durchsatz bei niedriger Systemlast. Mit 1/IAT=5000 hat diese Priorisierung ca. 25000 Bit/Sekunde übertragen. Unter gleicher Systemlast hat die reservierte Klassebandbreit Priorisierung ca. 33000 Bit/Sekunde erreicht. Und die gewichtete Priorisierung verfügt mit ca. 36000 Bit/Sekunde im Vergleich zu den anderen Priorisierungen über den besten Durchsatz. Ab 1/IAT=10000 haben alle Algorithmen den maximalen Durchsatz erreicht. Die statische Priorisierung hat ca.38000B Bit/Sekunde erreicht. Die gewichtete Priorisierung und die KBP können ca. 37000 Bit/Sekunde übertragen.

5.5.2 Fairness

In diesem Kapital werden die drei Algorithmen in verschiedener Systemlast verglichen. Daher werden die Kurven in unterschiedlichen IATs mit CDF geplottet.

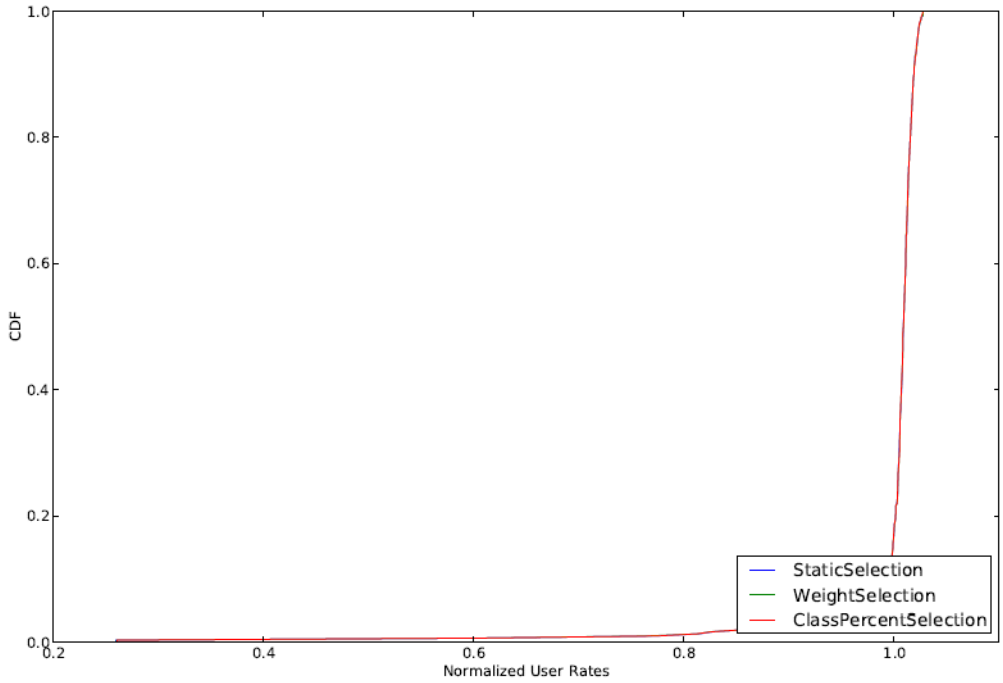

Abbildung 5.11 CDF von drei Algorithmen bei IAT=0.01

Die Abbildung 5.11 stellt die Fairness aller Algorithmen bei IAT=0.01 dar. Die drei Kurven überlappen sich komplett. Also ist, bei sehr geringer Systemlast, kein Unterschied zwischen den Priorisierungen zu sehen.

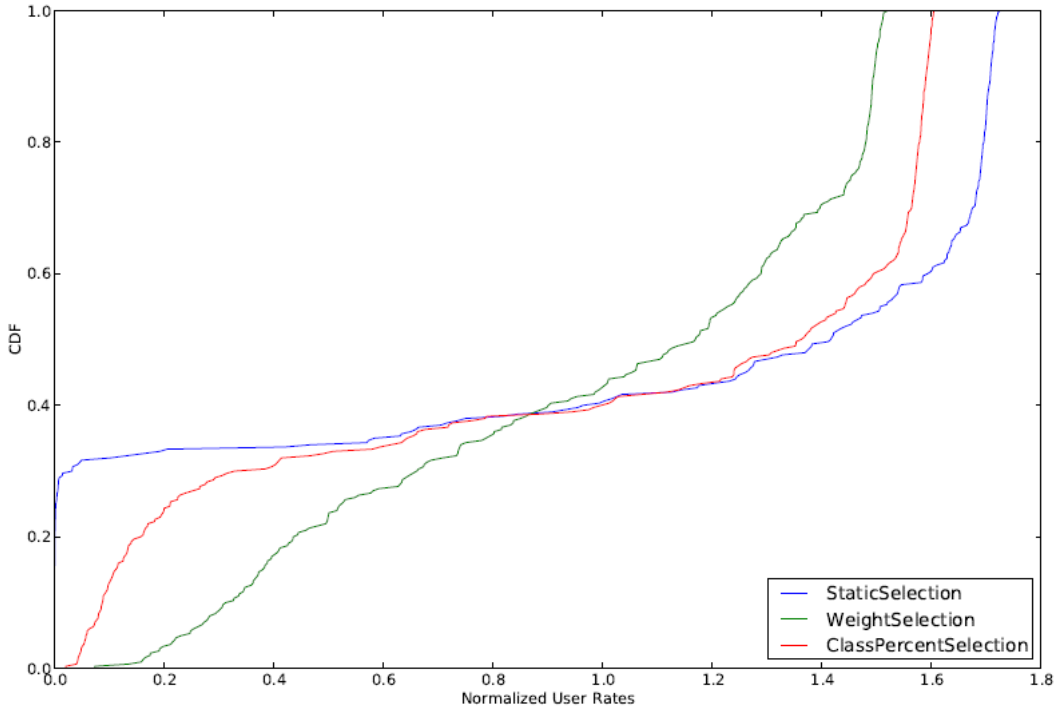

Abbildung 5.12 CDF von drei Algorithmen mit IAT=0.001

In Abbildung 5.12 ist die Fairness aller Algorithmen mit IAT=0.001 zu sehen. Die Systemlast ist um ein zehnfaches größer als in Abbildung 5.12. In dieser Abbildung ist die Stufe bei der statischen Priorisierung (blaue Kurve) bei dem Wert 0.33 auf der Y-Achse deutlich zu sehen. Also werden bei diesem IAT die Bedürfnisse der untersten Klasse bei der statischen Priorisierung nicht berücksichtigt. Die rote Kurve der KBP liegt im Graphen im Bereich der untersten Klasse rechts von der blauen Kurve – ganz rechts befindet sich die grüne Kurve der gewichteten Priorisierung. Die unterste Klasse bekommt bei der gewichteten Priorisierung mehr Ressourcen als bei den anderen zwei Verfahren. Interpretiert man den Kurvenverlauf anhand der NGMN-Fairness, so ist allein die gewichtete Priorisierung rechts von Grenzlinie der Fairnessbetrachtung, die anderen beiden befinden sich links davon, wobei Priorisierung nach Klassen, nicht weit entfernt ist.

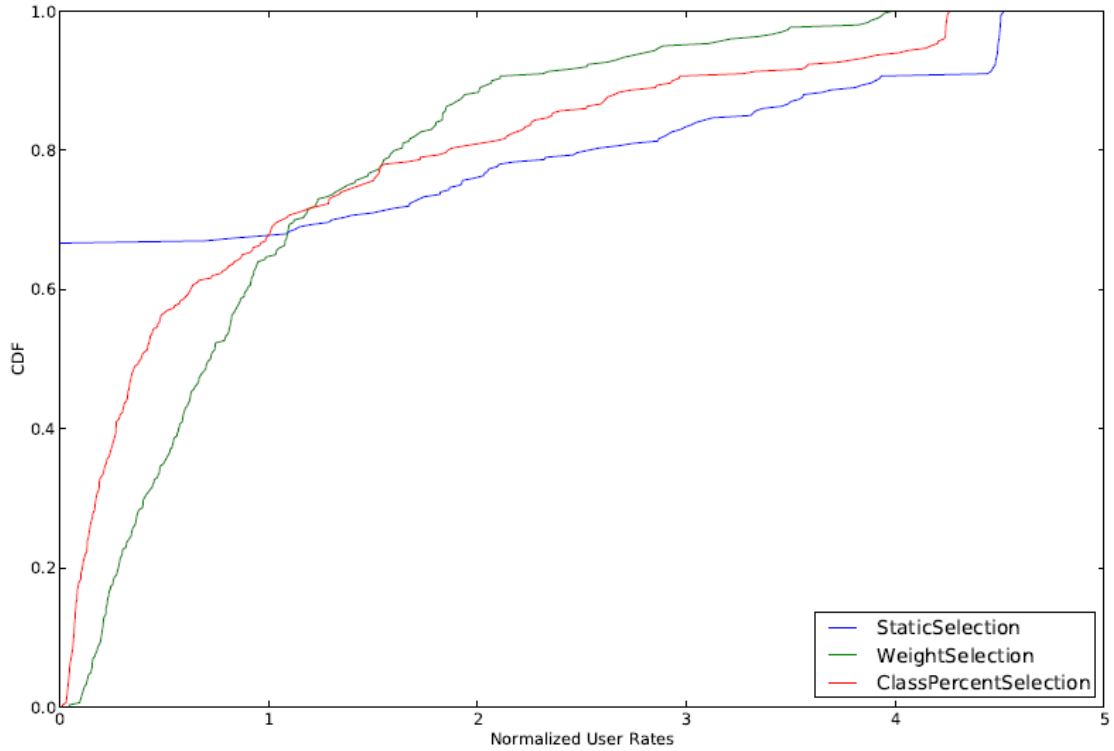

Abbildung 5.13 CDF von drei Algorithmen beim IAT=0.0002

Bei sehr großer Systemlast sieht die Fairness wie in Abbildung 5.13 aus. Bei der statischen Priorisierung (blaue Kurve) wird nur die oberste Klasse vom Scheduler bedient. Die erste Klasse bekommt alle Ressourcen zugewiesen. Bei der KBP (rote Kurve) ist der Durchsatz in erster Klasse geringer geworden, weil einiges an Bandbreite den unteren beiden Klassen zugeteilt wurde. Der Durchsatz, der unteren beiden Klassen ist bei der gewichteten Priorisierung stärker berücksichtigt. Sie erreichen unter dieser Priorisierung die größten Durchsätze. Gleichzeitig sinkt der Durchsatz der ersten Klasse noch weiter. Bei kleinem IAT hat die gewichtete Priorisierung die beste Fairness gezeigt. Die statische Priorisierung ist relativ unfair. Und die Fairness der KBP liegt zwischen die beiden Priorisierungen.

Interpretiert man den Kurvenverlauf anhand der NGMN-Fairness, so ist allein die gewichtete Priorisierung rechts von Grenzlinie der Fairnessbetrachtung, die anderen beiden befinden sich links davon, wobei Priorisierung nach Klassen, nicht weit entfernt ist.

5.5.3 Der Durchsatz in erster Klasse

Die Abbildung 5.14 zeigt den Durchsatz für die erste Klasse in allen drei obengenannten Priorisierungen. Die Y-Achse ist die Durchsatzachse. Die X-Achse bezeichnet 1/IAT. Die Pakete haben die Größe von 10^3 Bit. Jede Kurve bezeichnet den Durchsatz einer Priorisierung.

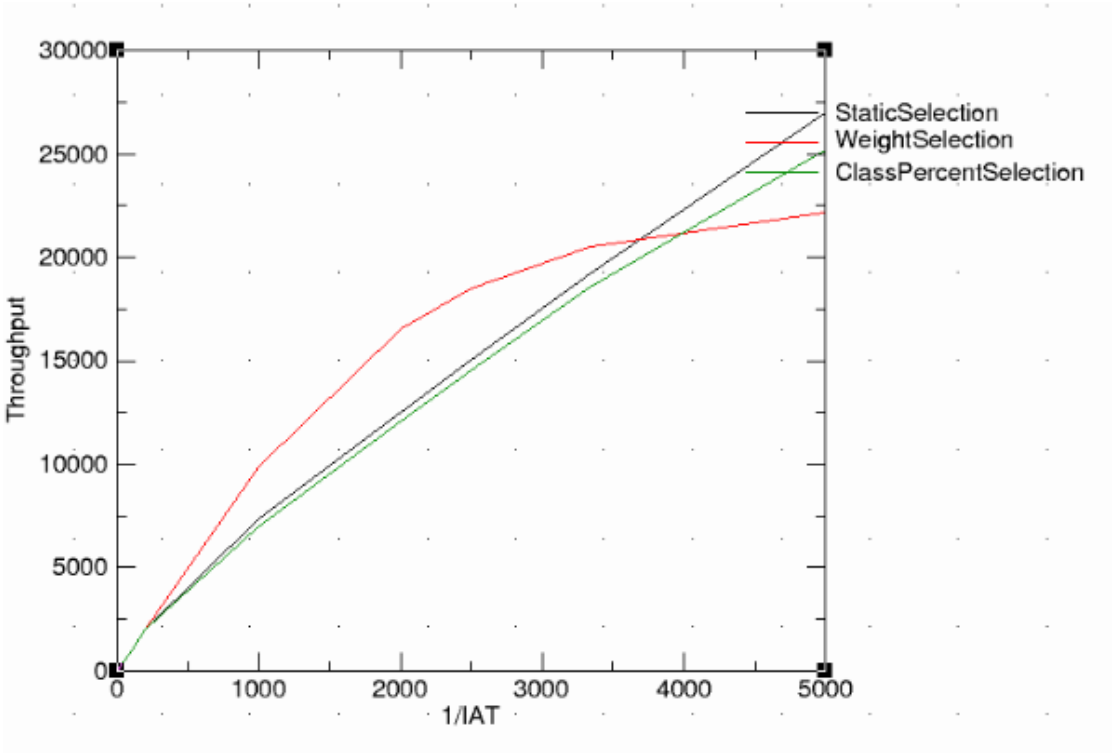

Abbildung 5.14 Der Durchsatz der ersten Klasse mit allen drei Algorithmen

In der Abbildung 5.14 ist zu sehen, das bei sehr geringem Verkehr (1/IAT zwischen 0 und 300) der Durchsatz der ersten Klasse bei allen drei Algorithmen gleich ist. Ab 1/IAT=300 ist der Durchsatz bei gewichteter Priorisierung (rote Kurve) höher als bei den anderen zwei Priorisierungen. Und bei der KBP sieht man den geringsten Durchsatz, der sich aber nicht wesentlich vom statischen Priorisieren unterscheidet. Die beiden Algorithmen haben ein nahezu lineares Wachstum des Durchsatzes. Allerdings schneidet die rote Kurve die anderen Kurven bei 1/IAT=3600. Also ist der Durchsatz der gewichteten Priorisierung ab 1/IAT=3600 geringer als die Durchsätze der anderen beiden Priorisierungen. Ab diesen Zeitpunkt hat die statische Priorisierung den besten Durchsatz.

5.5.4 Zusammenfassung

Die statische Priorisierung hat im Vergleich zu den anderen Algorithmen bis zu einer sehr hohen Last (1/IAT=9500) den geringsten gesamten Durchsatz. Die CDF-Kurven zeigen, dass die Fairness bezüglich der unteren Klassen bei großer Systemlast sehr schlecht ist. Der Jain's fairness index bewertet die statische Priorisierung aber auch bei kleinen 1/IATs schlecht. Der Durchsatz in der ersten Klasse liegt hier auch bei großen IATs unter der gewichteten Priorisierung. Die statische Priorisierung hat größten Durchsatz bei sehr starker Systemlast.

Die gewichtete Priorisierung kann als einer der besten Algorithmen bezeichnet werden. Diese Priorisierung hat den besten Durchsatz in allen IATs. Im Vergleich zu

anderen Priorisierungen hat die gewichtete Priorisierung die beste Fairnessbetrachtung, sowohl mit Jain's fairness index als auch in NGMN. Alle Klassen haben Ressourcen bekommen. Sogar bei kleinen IATs bekommen die unteren zwei Klassen immer noch zuverlässig Frequenzen - ihren Anteilen entsprechend. Deswegen ist die Steigung des Durchsatzes der ersten Klasse nicht konstant, wie bei den anderen beiden Algorithmen, sondern verringert sich bei kleinen IATs.

Die KBP hat einen mittleren gesamten Durchsatz im Vergleich zu den anderen Algorithmen. Bezüglich der Fairness liegt diese Priorisierung auch zwischen den zwei anderen Algorithmen. Bei der Durchsatzbetrachtung der ersten Klasse hat diese Priorisierung den geringsten Durchsatz, allerdings nicht weit vom Durchsatz der statischen Priorisierung entfernt.

5.6 Padding

In diesem Kapital wird das Padding in der statischen Priorisierung analysiert. Es wird mit verschiedenen IATs gemessen. Gleichzeitig werden wir zeigen, wie der Durchsatz mit Padding bei der statischen Priorisierung aussieht.

5.6.1 Szenario

Die Simulation mit Padding wird mit den folgenden Parametern durchgeführt:

	Anzahl der Nutzer
Prioritätsklasse 0	10
Prioritätsklasse 1	10
Prioritätsklasse 2	10

Tabelle 5.9 Parameter der statischer Priorisierung mit Padding

Das System hat insgesamt 30 Nutzer. Die Nutzer sind in drei Gruppen unterteilt. Jede Gruppe hat genau 10 Nutzer. Außerdem ist die Anzahl der Frequenzen auf 50 festgesetzt. Die Paketgröße ist 1000 Bit. Und die Ankunftsfrequenz der Pakete wird mit dem Parameter *Mean* gesteuert. In der Simulation werden die Werte der Ankunftsfrequenz von 0.0001 bis 1 gleichmäßig im SimTree Parameterfile vorgegeben d.h. die Simulation wurde zwischen der minimalen Systemlast von 10^3 Bit/Sekunde und der maximalen Systemlast von 10^7 Bit/Sekunde durchgeführt.

5.6.2 Die Abbildung von Padding

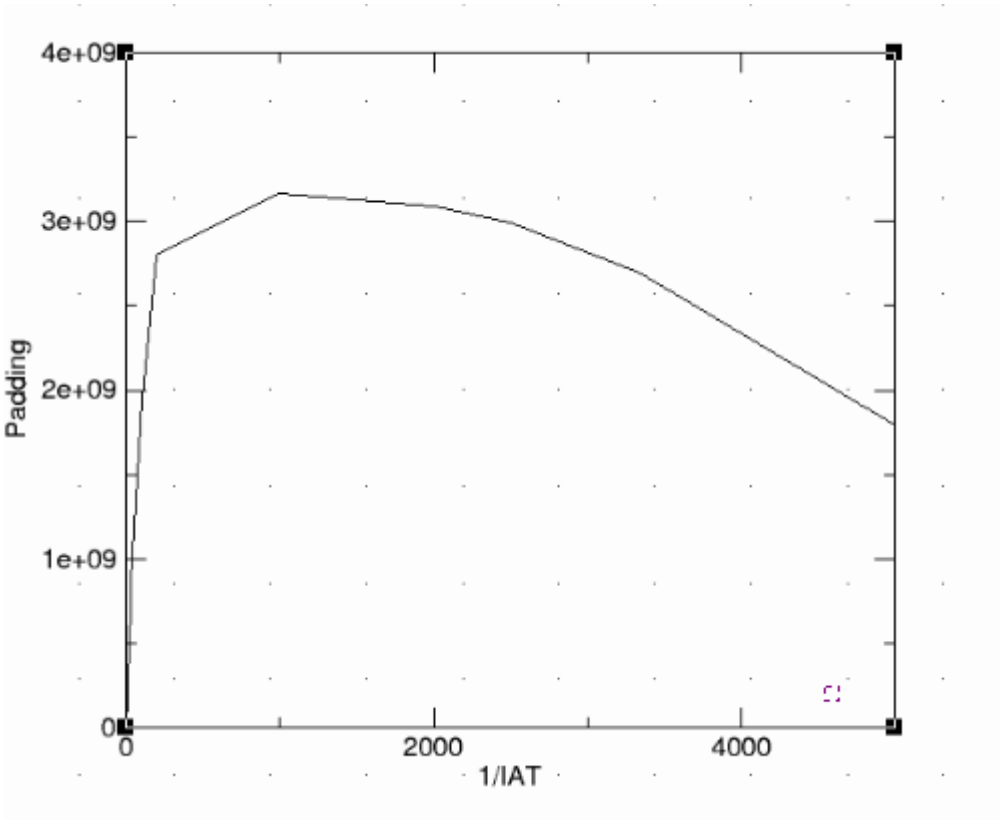

Abbildung 5.15 Padding aus statischer Priorisierung

Die Abbildung 5.15 beschreibt das Padding bei verschiedenen IATs. In diesem Bild kann man sehen, dass das Padding von 0 bis 1/IAT=200 linear gestiegen ist und schnell seinen maximalen Wert erreicht hat. Bei weiter steigender Systemlast sinkt das Padding wieder. Aber der Abstieg ist deutlich langsamer als Anstieg. Bis 1/IAT= 7500 erreicht es den Wert Null.

Am Anfang ist das System unter leichter Last gelaufen. Die Pakete der ersten Klasse wurden in jedem Frame komplett übertragen. Im diesen Fall nimmt das Padding zu, weil es sehr wenige übertragende Nutzer gibt. Der Durchsatz dieser Nutzer ist manchmal auch nicht sehr hoch. Das dadurch entstandene Padding ist deutlich geringer als das Padding bei vielen Nutzern. Mit zunehmender Nutzeranzahl ist das Padding schnell an seinem maximalen Wert angekommen. Mit zunehmender Systemlast haben alle Klassen immer mehr Pakete zu übertragen. Die Ressourcen, die für einen Nutzer zu viel sind, sind immer geringer geworden. Deswegen ist das Padding abgefallen. Das Padding sinkt immer weiter gegen Null. Ab einem bestimmten 1/IAT ist das System so belastet, dass das Padding nicht mehr vorkommt.

5.6.3 Der gesamte Durchsatz

Die folgende Abbildung beschreibt den Durchsatz bei gewichteter Priorisierung. Die

Y-Achse ist die Durchsatzachse. Die X-Achse bezeichnet 1/IAT. Die schwarze Kurve stellt den gesamten Durchsatz dar. Die anderen farbigen Kurven entsprechen den jeweiligen Klassen.

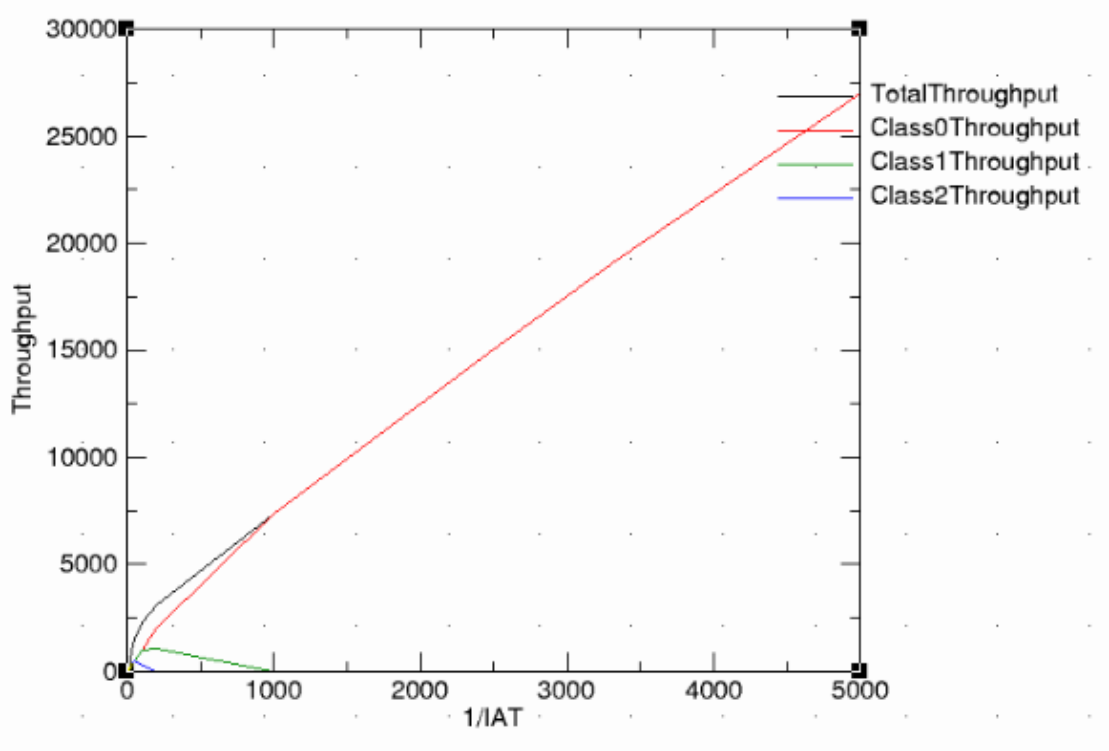

Abbildung 5.16 Der gesamte Durchsatz bei statischer Priorisierung mit Padding

Ab einem bestimmten 1/IAT sinkt der Durchsatz der untersten Klasse und entfernt sich von den anderen beiden. Bei weiter zunehmender Systemlast verlieren alle Klassen, die keine höchste Priorität haben, langsam ihren Durchsatz. Ab dem Zeitpunkt, an dem die vorletzte Prioritätsklasse keine Ressourcen bekommt (1/IAT=1000), sind alle Ressourcen von der obersten Klasse besetzt. In der Abbildung ist dies auch gezeigt. Ab 1/IAT=1000 überlappen sich der gesamte Durchsatz (schwarze Kurve) und der Durchsatz in Klasse 0 (rote Kurve).

6 Zusammenfassung

Ziel dieser Diplomarbeit war es, die verschiedenen Scheduler-Algorithmen in der Basisstation zu implementieren, die Untersuchung und Abwägung zwischen Übertragungseffizienz und Fairness und einer adaptiven Parametrisierung der Fairnesseigenschaften eines Schedulers durchzuführen. Durch die Simulation mussten die Ergebnisse hinsichtlich Fairness, Zelldurchsatz und Erfüllungsgrad der QoS-Anforderungen analysiert und diskutiert werden.

Im Rahmen dieser Arbeit wurden die vier Algorithmen entworfen, die verschiedenen Mechanismen zur QoS Priorisierung realisiert. Die implementierten Algorithmen wurden mit SimLib simuliert.

Nach Auswertung der Simulation hat sich gezeigt, dass es nicht einen besten Scheduler gibt, sondern je nach Szenario unterschiedliche Scheduler die beste Leistung zeigen.

Betrachtet man den Gesamtdurchsatz liegt die gewichtete Priorisierung bei geringer und mittlerer Systemlast klar vorne, während sich bei hoher Last fast keine Unterschiede zeigen.

Auch wenn die garantierte Bandbreite für untere Klassen, also eine hohe Fairness, im Vordergrund steht, ist die gewichtete Priorisierung die beste Wahl.

Und auch, wenn eine möglichst hohe Bandbreite für die erste Klasse garantiert werden soll, ist die gewichtete Priorisierung der beste Kandidat. Erst bei sehr hoher Systemlast wird sie von der statischen Priorisierung überholt.

Das Ergebnis ist also, dass die gewichtete Priorisierung in den meisten Fällen, die beste Wahl ist.

Weiterführende Arbeiten können den Algorithmus zur NBP ohne Padding implementierten. Außerdem können die Scheduling Algorithmen inerhalb einer Klasse statt mit Proportional Fair Scheduling mit anderen Scheduling Algorithmen simuliert werden und dann die Ergebnisse vergleichen werden.

A Literaturverzeichnis

[1] E-commerce Magazen,
http://www.e-commerce-magazin.de/ecm/news
/immer-mehr-deutsche-gehen-mit-dem-handy-ins-internet

[2] Internet Telecommunication Union, http://www.itu.int/ITU-D/ict/statistics/

[3] o-telko. Tele- u. Bürokommunikation,
http://www.o-telko.de/lte.html

[4] Wikipedia, http://de.wikipedia.org/wiki/Long_Term_Evolution

[5] 3GPP TR 25.892; Feasibility Study for Orthogonal Frequency
Division Multiplexing (OFDM) for UTRAN enhancement (Release 6)

[6] Rohde Schwarz,
http://www2.rohde-schwarz.com/file_10948/1MA111_2E.pdf

[7] Wikipedia, http://de.wikipedia.org/wiki/Quality_of_Service

[8] Lupocom,
http://www.lupocom.com/fachbuch/qos/
483-qos-101-quality-of-service-qos-in-der-praxis.html

[9] Wikipedia, http://de.wikipedia.org/wiki/Best_Effort

[10] Alexander Schüll, Fachhochschule Köln Diplomarbeit Untersuchung von
QoS-Mechanismen für VoIP in LAN und WLAN-Umgebungen

[11] Wikipedia, http://de.wikipedia.org/wiki/IntServ

[12] Fiandrino Claudio web page,
http://claudiofiandrino.altervista.org/Master_degree/
Network_management_and_QoS_provisioning/scheduling.pdf

[13] Wikipedia, http://en.wikipedia.org/wiki/Proportionally_fair

[14] Matthew Andrews. Lijun Qian. Alexander Stolyar,
Optimal Utility Based Multi-User Throughput Allocation subject to Throughput Constraints

[15] IKR Simulation Library-Resources,

www.ikr.uni-stuttgart.de/INDSimLib/Resources

[16] Wikipedia, http://de.wikipedia.org/wiki/Durchsatz

[17] Universität Magdeburg,
http://www-e.uni-magdeburg.de/zoellner/wiwi0304/folien15_031203.pdf

[18] A White Paper by the NGMN Alliance,
Next Generation Mobile Networks Radio Access Performance Evaluation Methodology

[19] Wikipedia, http://de.wikipedia.org/wiki/Konfidenzintervall

[20] Wikipedia, http://en.wikipedia.org/wiki/Cumulative_distribution_function

[21] Wikipedia, http://en.wikipedia.org/wiki/Jain%27s_fairness_index

B Abbildungsverzeichnis

C Tabellenverzeichnis

D Erklärung

Mir ist bekannt und ich erkenne an, dass ich an den Ergebnissen meiner Diplomarbeit keine eigenständigen Verwertungsrechte habe. Eine Verwertung der Arbeit einschließlich der Ausarbeitung darf nur nach Zustimmung durch das Institut für Kommunikationsnetze und Rechnersysteme erfolgen.

Ich erkläre weiterhin, dass ich diese Arbeit selbständig verfasst und keine anderen als die in meiner Ausarbeitung angegebenen Hilfsmittel für die Durchführung der Arbeit verwendet habe.

Ludwigsburg, den 23.08.2011

Unterschrift: