TECHNISCHE UNIVERSITÄT CHEMNITZ

Fakultät für Informatik

Professur Rechnernetze und Verteilte Systeme

Studienarbeit

Dynamisches Bandbreitenmanagement im Chemnitzer StudentenNetz

Markus Schade

Chemnitz, den 30. Mai 2006

Betreuer: Dr. Markus Borschbach
Dr. Jörg Anders

Danksagung

Ich möchte zunächst Jan Horbach, dem „Erfinder“ des DynShapers, für seine Ideen zur dynamischen Bandbreitenbeschränkung danken. Ohne die in seiner Arbeit gelegten Grundlagen hätte das CSN kein Shaping. Mein besonderer Dank gilt auch Norbert Scheibner, Thomas Dotschuweit, den Gebrüdern Parthey und allen anderen, die an der ersten Überarbeitung mitgewirkt haben. Natürlich bin ich auch allen früheren und jetzigen Mitarbeitern des CSN zu Dank verpflichtet. Es war und ist eine große Leistung ein solches Netz aufzubauen, aber noch eine viel größere, es so zu betreiben, daß Entwicklungen, wie die des DynShapers, möglich waren und es auch in Zukunft sind. Schließlich möchte ich Herrn Dr. Borschbach dafür danken, daß er die Betreuung dieser Arbeit auch nach seinem Weggang aus Chemnitz fortgesetzt hat.

Aufgabenstellung

Das Chemnitzer StudentenNetz (CSN) setzt seit mehreren Jahren ein System zur automatischen Regelung der Bandbreite („DynShaper“) basierend auf den Grundlagen der Diplomarbeit von Jan Horbach „Dynamische Bandbreitenbeschränkung mit QoS“ [4] ein. Aufgrund der weiterhin bestehenden Notwendigkeit zum Einsatz dieses Systems, ist es nötig die DynShaper-Software zu überarbeiten, um ihre Implementierung an die Standards des CSN anzupassen und die unvollständige Dokumentation zu ergänzen. Hauptaugenmerk liegt dabei auf der Integration des Systems in die bestehende Softwarearchitektur des CSN und der Schaffung einer modularen Implementierung zur Evaluierung anderer Berechnungsverfahren.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Accounting Erfassung des Datenvolumens

BNC Bayonet Nut Connector

bzw. beziehungsweise

CBQ Class Based Queuing

CSN Chemnitzer StudentenNetz

d.h. das heißt

DFN Deutsches Forschungsnetz

DHCP Dynamic Host Configuration Protocol

DNS Domain Name Service

DoS Denial of Service

DSL Digital Subscriber Line

dt. deutsch

engl. englisch

FIFO First In First Out

GWIN Gigabit-WIN

HTB Hierarchical Token Bucket

MB Megabyte

MTU Maximum Transfer Unit

IP Internet Protocol

ISO International Standards Organization

LAN Local Area Network

qdisc queueing discipline

P2P Peer-to-Peer

PPP Point-to-Point Protokoll

PPPoE Point-to-Point Protokoll over Ethernet

QoS Quality of Service

SFQ Stochastic Fair Queueing

StuWe Studentenwerk Chemnitz-Zwickau

TCP Transmission Control Protocol

TP Twisted Pair

Traffic Datenvolumen

UDP User Datagram Protocol

URZ Universitätsrechenzentrum

VPN Virtual Private Network

WIN Wissenschaftsinternet

WLAN Wireless LAN

WWW World Wide Web

XWIN 10-Gigabit-WIN

z.B. zum Beispiel

ZIN Zertifikat Internet-Nutzung

1 Einleitung

Die Nutzung des Internets ist zu einem alltäglichen Vorgang geworden. Nur Wenige können sich noch an die Zeiten erinnern, in denen Breitbandanschlüsse unerschwinglich waren und selbst langsame Modemverbindungen horrende Kosten verursachten. Mit der ungebremst ansteigenden Nutzung tauchten auch neue Probleme auf. Zum einen konnten die Kapazitäten der Leitungen dem erhöhten Bedarf an Bandbreite nicht gerecht werden, zum anderen erlautben die begrenzten Budgets der meisten Einrichtungen kein unbegrenztes Datenvolumen (Traffic).

Im Zuge dieser Entwicklung war es auch für das CSN notwendig, eine Lösung für diese Probleme zu finden. Dazu entwickelte Jan Horbach in seiner Diplomarbeit [4] ein Konzept zur automatischen Regelung der Bandbreite basierend auf dem verbrauchten Traffic jedes Nutzers. Die daraus enstandene Software, genannt „Dynamic Traffic Shaper“ oder kurz „DynShaper“, wurde über eigens dafür geschaffene Schnittstellen an die vorhandene CSN-Software angebunden. Schon kurz nach der Einführung des Systems hagelte es Proteste wütender Nutzer über einige Unzulänglichkeiten des Systems. Allein in der Newsgruppe des CSN sammelten sich mehrere hundert Postings an. Nicht zuletzt weil es dem Berechnungsmodell an Transparenz mangelte, wurde innerhalb eines Jahres ein neues System entworfen, welches fast alle Kritikpunkte beseitigte.

Die Änderungen wurden damals an der ursprünglichen DynShaper-Software vorgenommen. Zu diesem Zeitpunkt war es dem CSN leider personell nicht möglich, die Gelegenheit zu nutzen, die Implementierung an bestehende CSN-Standards anzupassen. Im Laufe der Zeit wurden weitere Änderungen, wie die Umstellung auf *iptables* MARK für die Zuordnung der Nutzer, eingebracht. Die daraus resultierende Ansammlung aus undokumentiertem und schlecht zu pflegendem Code erfüllte zwar noch ihre Funktion, blieb aber separiert von der übrigen CSN-Software.

Im Rahmen dieser Studienarbeit soll die Integration und Dokumentation der aktueller Software erfolgen und die Grundlage für ein modulares Shaping-System geschaffen werden, in das Änderungen und Alternativen leichter eingepflegt werden können.

1.1 Chemnitzer StudentenNetz

Das CSN bietet seit über zehn Jahren Studenten die Möglichkeit sich auch im Wohnheim an das Campusnetz der TU Chemnitz anzuschließen. Es ist eine Initiative von Studenten für Studenten, die ausschließlich aus ehrenamtlichen Mitarbeitern besteht. Diese kümmern sich um die Verwaltung der Technik und unterstützen ihre Kommilitonen beim Anschluß. Aus dem Kreis der Mitarbeiter wird vom StudentenRat der TU Chemnitz das CSN-Team gewählt, welches für die administrativen und organisatorischen Belange des Netzbetriebes verantwortlich ist.
Derzeit sind mehr als 1500 Nutzer und etwa 1700 Rechner an das CSN angeschlossen. Jeder Mieter der Chemnitzer Wohnheime des Studentenwerkes Chemnitz-Zwickau (StuWe) hat die Möglichkeit einen CSN-Anschluß zu beantragen. Neben einem solchen Mietvertrag sind weitere Voraussetzungen ein gültiges Nutzerkennzeichen im Universitätsrechenzentrum (URZ) und für den CSN-Standard-Anschluß ein Zertifikat Internet-Nutzung (ZIN). Nutzer ohne ZIN erhalten den eingeschränkten CSN-Light-Anschluß.

1.1.1 Netzwerkinfrastruktur

Die Infrastruktur des Netzwerks hat sich in den letzten zehn Jahren von einem frei verkabelten Bayonet Nut Connector (BNC)-Netz mit Leihgeräten des URZ zu einem heterogenen Twisted Pair (TP)/BNC-Netz mit einem nahezu vollautomatisierten Netzmanagement entwickelt.

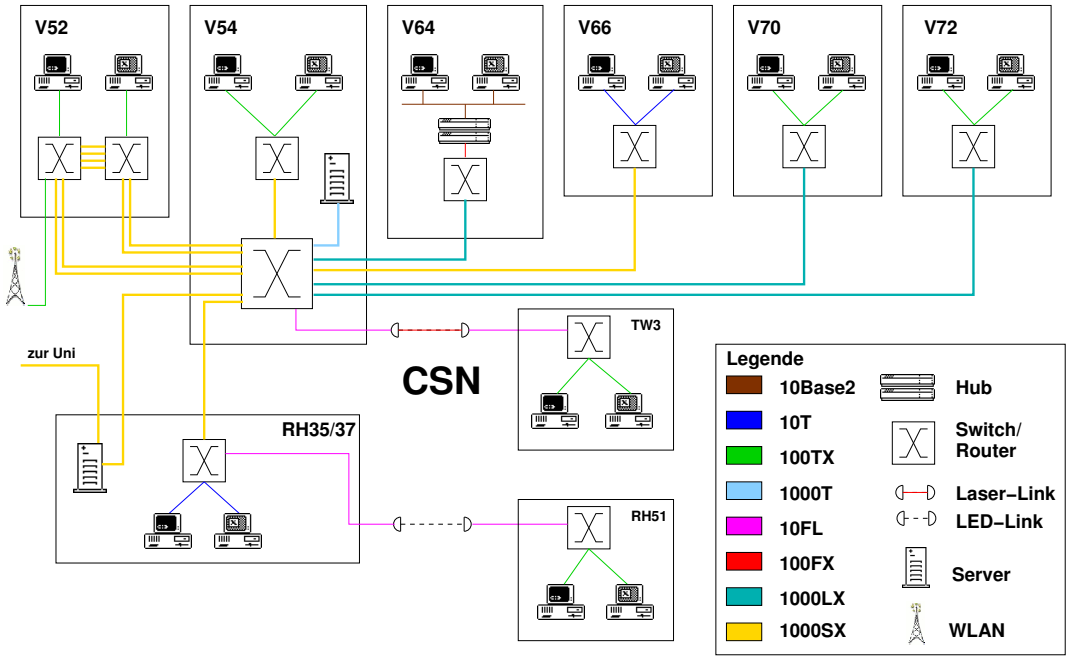

Abbildung 1.1: Netzstruktur des CSN (Stand April 2006)

Neue Technik wurde in der Vergangenheit zum Großteil durch das Studentenwerk Chemnitz-Zwickau (StuWe) finanziert. Durch die Reduzierung von Mitteln muß das CSN Neu- und Folgeinvestitionen in Zukunft selbst tragen. Abbildung 1.1 zeigt die Netzwerkstruktur des CSN im April 2006.

Eine detaillierte Beschreibung der Infrastruktur wurde von mir [9] 2004 veröffentlicht. Einzelne darin enthaltene Ideen zur Weiterentwicklung des CSN wurden bereits 2005 ganz oder teilweise realisiert. Sebastian Junge hat in seiner Studienarbeit [8] eine Lösung implementiert, um verschiedene Zugangmöglichkeiten, Netzklassen genannt, zu realisieren. Parallel dazu entwickelten Daniel Schreiber und ich ein neues Firewallkonzept [10], welches die Idee der Netzklasse auch in der CSN-Firewall umsetzt. Eine zusätzliche Zugangsmöglichkeit wurde durch die Diplomarbeit [3] von Alexander Glöckner geschaffen. Ein Wireless LAN (WLAN) bietet den Nutzern seit November 2005 einen drahtlosen Zugang zum CSN an.

1.1.2 Netzwerkmanagement

Für die Verwaltung von über 100 aktive Netzkomponenten und weit mehr als 2400 Anschlußports, ist ein leistungsfähiges Netzwerkmanagement erforderlich. Die CSN-Datenbank bildet die Grundlage für eine Vielzahl von Automatismen, die in regelmäßigen Abständen das Netz mit der Datenbasis abgleichen. Die einzelnen Skripte modifizieren die Konfiguration von Switches, Erzeugen und Löschen Firewall-Regeln, generieren Konfigurationsdateien oder lösen Neustarts von verschiedenen Diensten aus. Für den DynShaper bedeutet dies: Umordnung der Nutzer und Neuberechnung der Bandbreiten respektive eines Bandbreitenfaktors.

1.1.3 Erfassung des Datenvolumens

Zur Kontrolle und Einhaltung der Begrenzung muß der Datenverkehr erfasst, aufgezeichnet und ausgewertet werden. Zur Erfassung wird die NetFlow-Funktionalität des zentralen CSN-Routers (Cisco 6500) genutzt. Dieser exportiert Verbindungsdaten, keine Inhalte, die ein Server zwischengespeichert und anschließend ausgewertet. Hierbei wird lediglich die Datenmenge pro Internet Protocol (IP)-Adresse in einem Report zusammengefasst. Im nächsten Schritt erfolgt die Übermittlung der Reports an den Web- und Datenbankserver des CSN. Die Verbindungsdaten werden nach ihrer Auswertung zum frühestmöglichen Zeitpunkt gelöscht. Das Einfügen des Reports in die CSN-Datenbank und die Zuordnung des Datenvolumens zu einem Nutzer beendet die Erfassung des Datenvolumens (Accounting). Zur Selbstkontrolle des Nutzers dient eine World Wide Web (WWW)-basierte Schnittstelle, über welche dem Nutzer sein bisher verbrauchtes Datenvolumen angezeigt wird, sowie die Information, welche Bandbreite ihm zur Verfügung steht.

1.2 Historische Entwicklungen

1.2.1 Einführung der Kontingentierung

Das Deutsches Forschungsnetz (DFN) bietet den Universitäten mit seinem „DFN-Internet“-Anschluß Zugang zum Wissenschaftsinternet (WIN) und damit auch zum weltweiten Internet an. Zu Beginn des Sommers 2000 gab das DFN bekannt, daß ab 2001 zusammen mit der Freigabe der nächsten Ausbaustufe des Netzes, dem Gigabit-WIN (GWIN), eine Kontingentierung des Datenvolumens einführt wird. Dies bedeutete im Vergleich zur vorher unbeschränkten Nutzung für viele Einrichtungen eine gewaltige Umstellung, weil das monatliche Volumen nur einmal überschritten werden durfte. Danach erhielt die betroffene Einrichtung eine Verwarnung und die Empfehlung in eine höhere Tarifklasse zu wechseln. Im Wiederholungsfall drosselte das DFN die Anschlüsse auf eine Bandbreite, die ein Überschreiten unmöglich machte. Die Universitäten sahen sich daher unter Druck gesetzt, ihren externen (von außerhalb der Einrichtung) Traffic zu kontrollieren und die Studentennetze befürchteten eine Abschaltung ihrer Zugänge. Nach einer Empfehlung des DFN sollte dies das Mittel der Wahl sein, um etwaige Probleme mit der Einhaltung des Kontingentes in den Griff zu bekommen.

Auch in Chemnitz sah man sich mit dem Problem der Kontingentierung konfrontiert. Doch im Gegensatz zu den Kurzschlußreaktionen andernorts, wurde das Problem zwischen URZ und CSN konstruktiv besprochen. Man einigte sich auf ein festes Teilvolumen, welches auf die Nutzer aufgeteilt werden durfte.

1.2.2 Feste Grenzen und Sperrungen

Für die Aufteilung mußte ein bis dato nicht existentes Accounting-System entwickelt werden. Einen sanften Einstieg bot zunächst die Auswertung der Logdaten des WWW-Proxys. Später wurde das Datenvolumen vollständig durch die Regeln der *ipchains*-basierten Firewall erfasst. Der nächste Schritt bestand in der Festlegung einer täglichen Obergrenze pro Nutzer. Die Auswertung der Accountingwerte ergab einen durchschnittlichen Tagesverbrauch von 33 MB externem Traffic pro Nutzer. In einer Mischkalkulation wurden 250 MB pro Tag als Obergrenze angesetzt. Diese Grenzen wurden im Oktober 2000 eingeführt und sorgten dafür, daß sowohl URZ als auch CSN gelassen dem Beginn der Kontingentierung im Januar 2001 entgegenblicken konnten.

Leider gab es auch weniger einsichtige Nutzer, die keine Veranlassung sahen, das Netz rücksichtsvoll und ressourcenschonend zu nutzen. Die Sanktionen für eine Überschreitung des zulässigen Datenvolumens sahen daher eine einmalige Verwarnung und bei nochmaliger Übertretung eine Sperrung des Zugangs vor. Parallel dazu gab es noch eine weitere Grenze für den insgesamt von einem Rechner

verursachten Traffic, welche dazu diente, tagsüber auch in den Wohnheimen mit BNC-Technik ein interaktives Arbeiten zu ermöglichen.

1.2.3 Der Weg zum Shaping

Die Auswertung der Daten und die manuelle Sperrung von Übertretungen verursachten einen so beträchtlichen Arbeitsaufwand, daß dafür ein eigener Mitarbeiter erforderlich war. Jan Horbach nahm sich in seiner Diplomarbeit dieser Probleme an. Maßgäblich war auch das Ziel, ein System zu schaffen, in dem ein verantwortungsvoller Umgang mit den Ressourcen belohnt und ihr Mißbrauch geahndet wird. Mit der Umstellung auf ein Berechnungsmodell basierend auf dem Durchschnittsverbrauch des Nutzers gewann auch die Fairness innerhalb des Systems an Bedeutung.

1.2.4 Stand 2006

Im Januar 2006 hat das DFN eine weitere Ausbaustufe des WIN fertiggestellt. Das 10-Gigabit-WIN (XWIN) beendet gleichzeitig die Ära der Kontigentierung. Für das CSN hat sich dadurch nichts verändert. Im beiderseitigen Einvernehmen mit dem URZ bleibt das Shaping auch in Zukunft aktiv, da es nach wie vor erforderlich ist, positiv auf die Studierenden einzuwirken und sie zu einem verantwortungsvollen Umgang mit den ihnen zur Verfügung gestellten Ressourcen zu ermutigen.

2 Shaping à la Horbach

2.1 Arbeitsweise

Bei der Betrachtung des Horbachschen Systems werde ich insbesondere auf die Probleme eingehen, welche zu seiner Ablösung geführt haben. Keinesfalls soll an dieser Stelle der Wert von Jan Horbachs Arbeit gemindert werden, vielmehr möchte ich seine Pionierleistung würdigen und ihm für diese Arbeit danken. Ihm können seine Fehleinschätzungen nicht zur Last gelegt werden, da er, ohne die Möglichkeit zum praktischen Einsatz, das Verhalten der Nutzer nicht hat erahnen können. Für die Zukunft bedeutet es, daß jedes neue Modell unter realen Einsatzbedingungen geprüft und verifiziert werden muß.

2.1.1 Berechnung Nutzereinordnung

Das von Jan Horbach entworfene System [4] arbeitet mit fünf Klassen, welchen feste Grenzen[1] zugeordnet sind (Tabelle 2.1). Die Auswertung und Neueinordnung anhand des verbrauchten Datenvolumens wird einmal täglich zu einem konfigurierbaren Zeitpunkt, in der Regel am Ende eines Tages, vorgenommen. Überschreitet der Nutzer die Grenze für Klasse fünf im Laufe des Tages, erfolgt die sofortige Neueinordnung innerhalb der nächsten Stunde.

Klasse	Grenze	Bandbreite	Verweildauer	sofortige Einordnung
0	-	unbeschränkt	keine	-
1	100 MB/Tag	10 MBit/s	2 Tage	nein
2	200 MB/Tag	5 MBit/s	4 Tage	nein
3	300 MB/Tag	3 MBit/s	6 Tage	nein
4	400 MB/Tag	500 kBit/s	8 Tage	nein
5	500 MB/Tag	100 kBit/s	10 Tage	ja

Tabelle 2.1: Limits im Horbachschen Shaping

Desweiteren erhält jede Klasse eine Verweildauer, um einen zusätzlichen Anreiz zur Einsparung von Traffic zu schaffen. Die Verweildauer gibt die Zeit in Tagen an, die in einer Klasse verblieben werden muß, bis ein Aufstieg in die nächsthöhere möglich ist. Bedingung für einen Aufstieg ist das dauerhafte Unterschreiten der Grenze der aktuellen Klasse. Für den Fall, daß der Nutzer während der Verweildauer

[1] Die Werte entstammen der ersten Anpassung des Systems. Sie entsprechend nicht denen der Diplomarbeit.

das Limit einer Klasse mit niedriger Bandbreite überschreitet, wird eine Strafeinordnung vorgenommen. Die aktuelle Klasse geht mit 25 % in die Neueinordnung ein (Gleichung 2.1).

$$\text{Klasse } 2 \cdot 25\,\% = 0,5 \approx 1 + 3 \text{ (für Klasse 3)} = 4 \tag{2.1}$$

Das bedeutet, daß einem Nutzer nach dem ersten Verstoß durch Überschreiten eines höheren Limits eine schlechtere Klasse zugewiesen wird, als einem nicht eingeordneten Nutzer. So steigt beispielsweise ein Nutzer nach einer Einordnung in Klasse drei nach sechs Tagen in Klasse zwei auf. Bei einer erneuten Überschreitung der Grenze von Klasse drei innerhalb der vier Tage Verweildauer in Klasse zwei wird der Nutzer nun in Klasse vier statt Klasse drei eingeordnet.
Jan Horbach führt in seiner Diplomarbeit [4] als Grund für die Strafeinordnung an, daß ein Nutzer, der zweimal 100 MB verbraucht, genauso eingeordnet werden sollte, wie ein Nutzer, der nur einmal 200 MB verbraucht hat. Das dies nicht ganz stimmt, soll hier nicht weiter betrachtet werden.

2.1.2 Beispiele

Zur Verdeutlichung des oben genannten Problems zwei Beispiele:

Beispiel 2.1 *Es wird angenommen, daß Nutzer A an Tag X in Klasse 0 ist und 650 MB durch den Download einer CD verbraucht hat. In den nächsten Tagen wird er nur noch 75 MB täglich verbrauchen. Am Tag X wird er sofort in Klasse 5 eingeordnet. Obwohl er an Tag X+1 nur 75 MB verbraucht, verbleibt er in Klasse 5. Nach 10 ganzen Tagen in Klasse 5, am Tag X+11, steigt der Nutzer in Klasse 4 auf, weil er weniger als 500 MB verbraucht hat. Dort wird er weitere 8 Tage verbringen, bis er am Tag X+19 in Klasse 3 aufsteigt. Diese Entwicklung setzt sich fort, bis der Nutzer nach insgesamt 30 Tagen wieder in Klasse 0 ist. In diesen 31 Tagen hat er 2900 MB verbraucht, was im Durchschnitt etwa 93 MB pro Tag entspricht.*

Beispiel 2.2 *Hier soll nun ausgehend von Klasse 0 jeden Tag 199 MB verbraucht werden.*
Am Tag X+1 wird Nutzer B in Klasse 1 eingeordnet, wo er auch die gesamte Zeit verbleibt. Am Tag X+30 senkt er den Verbrauch auf 99 MB pro Tag, was dazu führt, daß er an Tag X+31 ebenfalls wieder in Klasse 0 ist. Im Vergleich dazu hat er aber 6096 MB verbraucht. Hier liegt der Durchschnitt bei 196 MB pro Tag.

Die Beispiele machen ein Mißverhältnis bei der Regelung deutlich. Ein Nutzer, der nur einmalig ein hohes Datenaufkommen hat, wird mehr bestraft als jemand, der täglich wesentlich mehr verbraucht.

2.1.3 Globalregelung

Zur Einhaltung des Regelziels, dem monatlichen Kontingent, müssen die Bandbreiten der einzelnen Klassen je nach Auslastung modifiziert werden. Die Anpassung der Bandbreiten erfolgt mittels eines

Bandbreitenfaktors, mit dem die Initialbandbreiten multipliziert werden. Einmal am Tag wird eine Neuberechnung des Bandbreitenfaktors vorgenommen, in dem zunächst das seit Monatsbeginn verbrauchte Volumen auf den gesamten Monat hochgerechnet wird.

$$t_m = t \cdot \frac{d_m}{d} \qquad \begin{array}{lcl} t_m & - & \text{errechneter Monatsverbrauch} \\ t & - & \text{bisheriger Verbrauch im Monat} \\ d_m & - & \text{Anzahl der Tage im Monat} \\ d & - & \text{bereits verstrichene Tage} \end{array} \tag{2.2}$$

Liegt eine Überschreitung des vorgegebenen Regelziels vor, errechnet sich der neue Bandbreitenfaktor wie folgt:

$$b_{neu} = b_{alt} \cdot \frac{t_{limit}}{t_m} \qquad \begin{array}{lcl} b_{neu} & - & \text{neuer Bandbreitenfaktor} \\ t_{limit} & - & \text{Monatslimit} \\ b_{alt} & - & \text{alter Bandbreitenfaktor} \\ t_m & - & \text{errechneter Monatsverbrauch} \end{array} \tag{2.3}$$

Zu Beginn des Monats wird der Faktor mit eins initialisiert. Sollte die Hochrechnung des Monatsverbrauchs ergeben, daß noch Kapazitäten verfügbar sind, wird der Bandbreitenfaktor nach oben korrigiert, jedoch in abgeschwächter Form, um ein Schwingen der Bandbreiten zu verhindern.

$$b_{neu} = b_{alt} \cdot \sqrt{\frac{t_{limit}}{t_m}} \tag{2.4}$$

Daraus ergibt sich eine lineare Regelung. In den ersten 5 Tagen ist die Regelung zunächst noch deaktiviert, da die Hochrechnung in dieser Zeit noch starken Schwankungen unterworfen ist. In Abbildung 2.1 ist deutlich der Einfluß des Shapings auf den Verbrauch zu sehen. Durch den Anstieg des Bandbreitenfaktors über 1, und der damit verbundenen Erhöhung der Bandbreite aller Klassen, ist an den Tagen 7 bis 15 ein gestiegener Verbrauch sichtbar. Gegen Ende des Monats sinkt der Faktor auf etwa 0,8 und vermindert den Tagesverbrauch leicht.

Wenige Monate später (Mai 2002) ergibt sich ein völlig anderer Verlauf, der die Schwäche der Regelung zutage treten läßt (Abbildung 2.2). Der Mai 2002 stellte für den DynShaper den ersten ernstzunehmenden Test seiner Leistungsfähigkeit zur Regelung im Grenzbereich dar. Im Vergleich zum Januar ist eine deutliche Steigerung des Tagesverbrauchs und der wochenendbedingten Schwankungen festzustellen. In den ersten zwei Wochen des Monats ist kaum ein Einfluß auf den Verbrauch sichtbar und auch im weiteren Verlauf kann der sinkende Bandbreitenfaktor nicht die gewünschte Wirkung erzielen. Es wird vermutet, daß die Fertigkeiten einer signifikanten Anzahl Nutzer zur Auslastung der Klassen bis in den Grenzbereich so weit fortgeschritten war, daß die Regelung nicht mehr effektiv greifen konnte. Dazu

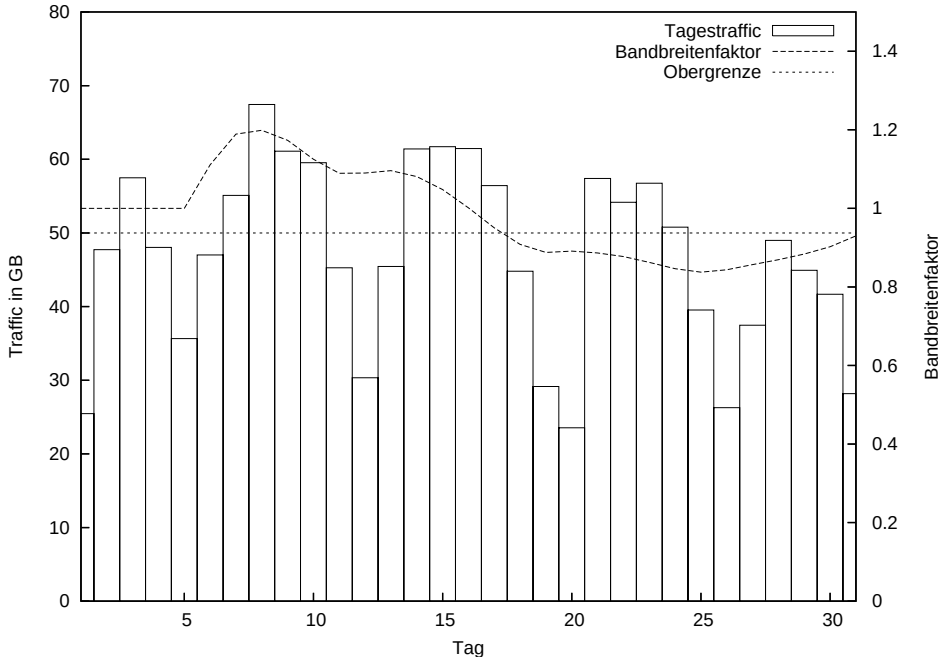

Abbildung 2.1: Shapingverlauf im Januar 2002

zählt auch das Phänomen, daß ein geringer Prozentsatz technisch versierter Nutzer (schätzungsweise 100-150 oder 10 %) durch kontinuierliches Ausschöpfen der Grenzen der Klassen 2 und 3 fast die Hälfte des Kontingents verbraucht, während andere Nutzer nach einmaligem Überschreiten des Limits die größere Hälfte des Monats in den Klassen 4 und 5 verbringen. An dieser Stelle sollte erwähnt werden, daß die Regelung im Mai nicht in der Lage war, das Limit einzuhalten. Als Gründe für das Scheitern wurden die lange Reaktionszeit, Schlupflöcher in der Regelung, technische Probleme und konzeptionelle Fehler im Klassendesign ermittelt. Auf diese Fehler soll im folgenden genauer eingangen werden.

2.2 Änderungen am Horbachschen System

Für die Realisierung des Shapings wird im CSN die Quality of Service (QoS) Funktionalität des Linux-Kernels verwendet. Im Kernel 2.6.x kann diese unter *Networking → Networking Options → QoS and/or fair queueing* aktiviert werden. Alle hier gemachten Angaben beziehen sich auf diese Implementierung. Andere Betriebssysteme (z. B. BSD) bieten eine ähnliche Funktionalität, sodaß, nach einer entsprechenden Anpassung, einem Einsatz des DynShapers nichts im Wege steht.

Schon früh wurden Probleme mit der initialen Version aus der Diplomarbeit sichtbar. Das eingesetzte Verfahren Class Based Queuing (CBQ) arbeitet nicht immer nachvollziehbar und sogar teilweise entgegen dem beabsichtigten Verhalten. Ein weiteres Problem bestand in der Art und Weise der Paket-

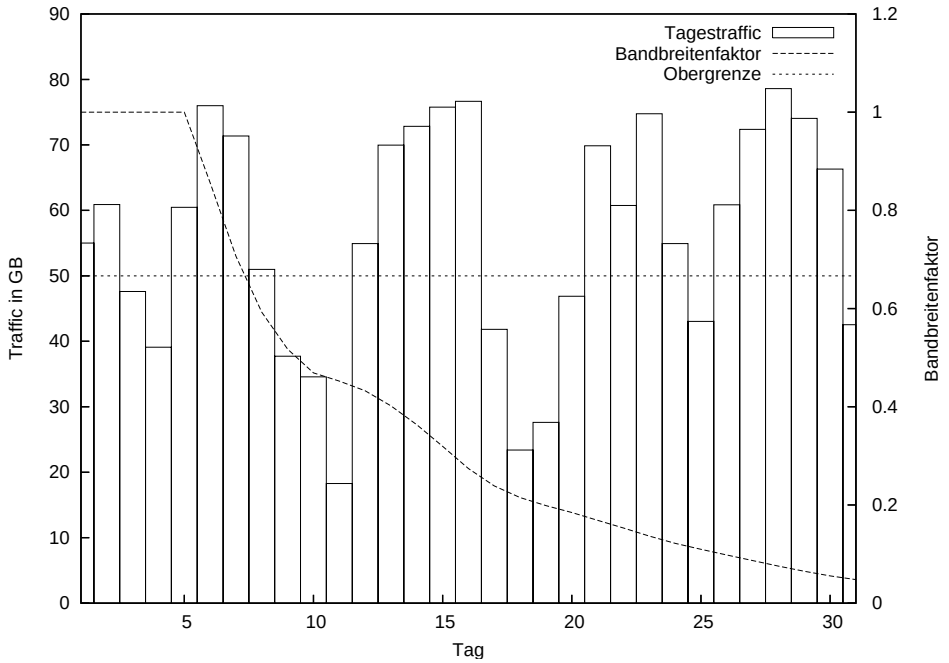

Abbildung 2.2: Shapingverlauf im Mai 2002

verarbeitung des Kernels. Durch das First In First Out (FIFO) Scheduling innerhalb der Klassen werden Nutzer mit mehr als einer IP-Adresse oder vielen Verbindungen bevorteilt.

2.2.1 Class Based Queueing

Das CBQ-Verfahren beruht auf der Berechnung des Leerlaufs eines Links basierend auf der mittleren Paketgröße und der Bandbreite des zu begrenzenden Links. Wenn also ein Link mit 10 MBit/s auf 1 MBit/s gedrosselt werden soll, dann muß der Link zu 90 % unbenutzt sein. Hierzu werden im Kernel komplexe Schritte durchgeführt, um die durchschnittliche Leerlaufrate zu berechnen. Was bei echten Ethernetanschlüssen theoretisch noch funktioniert, führt bei Point-to-Point Protokoll over Ethernet (PPPoE)- oder Virtual Private Network (VPN)-Verbindungen zu erheblichen Diskrepanzen. Fehlende und fehlerhafte Dokumentation einer nahezu unüberschaubaren Vielzahl an Parametern fördern Fehler und Probleme beim Verständnis und bei der Konfiguration. Desweiteren wird auch in der Literatur [6] zu *iproute2* und *tc*, dem Tool zum Erstellen der Regeln, darauf hingewiesen, daß CBQ in bestimmten Fällen nicht so funktioniert, wie man es erwarten würde. Es wird weiterhin empfohlen stattdessen HTB für reproduzierbare Ergebnisse zu verwenden.

2.2.2 Hierarchical Token Bucket

Da sich auch das CSN mit diesem Problem konfrontiert sah, wurde das System auf ein neues Verfahren umgestellt. Zum damaligen Zeitpunkt, war HTB noch ein Recht neues Modul im Kernel (mittels Patch ab 2.4.18, offiziell ab 2.4.20). In dem von Martin Devera entwickelten Verfahren [1] wird im Gegensatz zum CBQ die Bandbreite durch eine Hierarchie von Token Bucket Filtern geregelt. Dieses Verfahren implementiert ähnlich wie CBQ eine Variante des „formal sharing“, welches von Floyd [2] definiert wurde. Das Token Bucket Modell beschreibt einen Behälter (engl. Bucket), der mit einer konstanten Rate Tokens, vergleichbar mit Münzen, gefüllt wird. Die Füllrate (engl. rate) ist dabei die zugesicherte Bandbreite. Die Füllung wird unterbrochen, wenn der maximale Füllstand erreicht ist. Soll ein Paket gesendet werden, so kostet dies ein Token aus dem Bucket. Wenn der Bucket leer ist, wird das Paket in die Warteschlange gestellt, bis wieder Tokens verfügbar sind. Der Bucket hält eine gewisse Menge an Tokens vorrätig und erlaubt so auch kurzzeitig höhere Datenraten (Bursts). Im Falle einer andauernden Überlastsituation werden neue Packete verworfen, wenn die Warteschlange voll besetzt ist.

Im entgegengesetzten Fall, einer Unterlastsituation, kann die überschüssige Bandbreite an Kindklassen abgegeben werden. Dies wird Borgen (engl. borrowing) genannt. Die Kindklassen borgen sich dabei Bandbreite von ihrer Elternklasse, wenn ihr Bedarf höher als die zugesicherte Bandbreite ist. Wieviel und ob sich eine Klasse borgen darf, wird durch die Spitzenrate (engl. ceil rate) angegeben. Wenn die Spitzenrate gleich der zugesicherten Rate ist, darf keine Bandbreite geborgt werden. Ein sogenanntes Quantum wird benutzt, um anzugeben, in welcher Granularität die Klassen Bandbreite von der Elternklasse borgen können. Für Ethernetpakete wird in der Regel eine Maximum Transfer Unit (MTU) von 1500 Bytes verwendet. Daher sollte das Quantum mindestens so groß wie die MTU gewählt werden, damit mindestens ein Paket gesendet werden kann. Die Größe des Quantums kann entweder direkt angegeben werden oder wird automatisch durch Division durch den r2q (Rate to Quantum)-Faktor bestimmt. Dieser Faktor hat den Standardwert 10, der für Raten zwischen 120 kBit/s und 10 MBit/s ausgelegt ist.

2.2.2.1 Konstruktion eines HTB Klassenbaum

Bei der Konstruktion einer Klassenhierarchie ist darauf zu achten, daß die Summe der zugesicherten Bandbreite (engl. assured rate) der Kindklassen die Bandbreite der Elternklasse nicht übersteigt. Im Listing 2.1 wird eine einfache Klassenhierarchie mit einer Elternklasse und drei Kindklassen angelegt. Bevor jedoch mit dem Anlegen von Klassen begonnen werden kann, muß dem Kernel mitgeteilt werden, daß HTB als Warteschlangenregel (engl. queueing discipline) verwendet werden soll. Eine queueing discipline (qdisc) kommt immer dann zum Tragen, wenn ein Paket versendet werden soll. Der Kernel stellt es dazu in die Warteschlange der Netzwerkschnittstelle und versucht anschließend so-

viele Pakete wie möglich aus der Warteschlange zu entfernen, um sie an den Treiber der Netzwerkkarte zu übergeben. Ist keine andere qdisc definiert, wird *pfifo_fast* benutzt.

```
1 tc qdisc add dev eth0 root handle 1: htb default 2
2 tc class add dev eth0 parent 1: classid 1:1 \
3     htb rate 1mbit
4 tc class add dev eth0 parent 1:1 classid 1:2 \
5     htb rate 64kbit ceil 1mbit
6 tc class add dev eth0 parent 1:1 classid 1:3 \
7     htb rate 128kbit ceil 1mbit
8 tc class add dev eth0 parent 1:1 classid 1:4 \
9     htb rate 56kbit
```

Listing 2.1: Kommandos zur Erstellung einer HTB-Beispielhierarchie

Das Anlegen einer Klassenhierarchie hat in der Regel zunächst keine Auswirkungen, da es dem Kernel an Filtern mangelt, anhand derer er Pakete in Klassen einordnen könnte. Aus diesem Grund werden ohne Filter alle Pakete in die „default“-Klasse eingeordnet, im Beispiel Klasse 1:2, welche beim Festlegen der qdisc angeben werden kann.

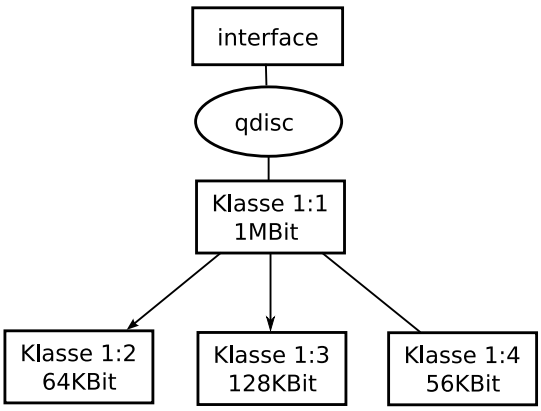

Abbildung 2.3: Klassenhierarchie eines HTB Beispielbaumes

2.2.2.2 Klassenhierarchie

Der reale Klassenbaum (Abbildung 2.4) enthält, neben den Nutzerklassen im rechten Teilbaum, einen zusätzlichen Zweig, der für Ausnahmen reserviert ist. Als mögliche Ausnahmen kommen bestimmte Dienste (DNS, E-Mail, etc.) oder Hosts in Frage, die von den Auswirkungen des Shapings ausgenommen werden sollen. Es ist so auch möglich, unerwünschte Dienste auf eine geringe Bandbreite zu begrenzen.

Es wurde bereits erwähnt, daß bei dieser Baumstruktur eine Verdrängung durch Nutzer mit vielen Verbindungen und/oder mehreren IP-Adressen möglich ist. Durch eine zusätzliche Unterteilung jeder Klasse, hätte dieses Problem gelösen werden können. Allerdings hatten die Simulationen von Jan Horbach gezeigt, daß bei mehr als 1000 Klassen der Durchsatz rapide einbricht und schon bei 1500 Klassen um mehr als die Hälfte gesunken ist. Allein das Anlegen der Klassen würde mehr als 10 Minuten dauern. Dank des Einsatzes von HTB konnte, wie Jan Horbach in [5] schreibt, eine deutliche Verbesserung erreicht werden. Die dadurch freiwerdende Systemleistung wurde jedoch sofort durch das Routing beansprucht. So konnte dieses Problem erst nach dem Kauf eines leistungsfähigeren Rechners zufriedenstellend gelöst werden. Auf den geänderten Klassenbaum wird in Abschnitt 3.3 gesondert eingegangen.

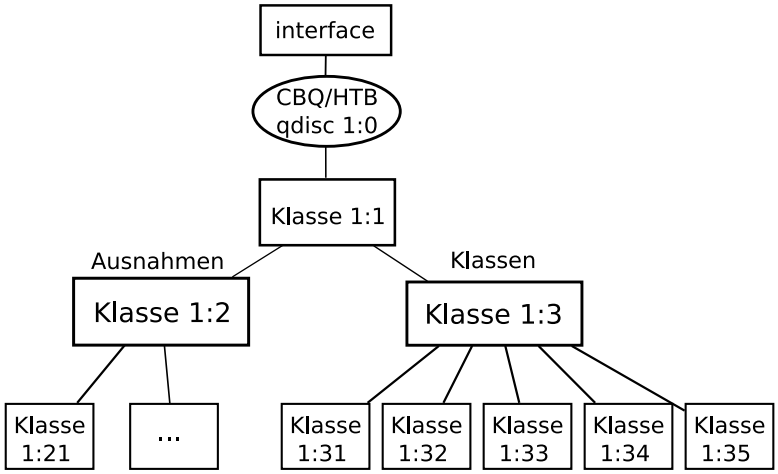

Abbildung 2.4: Klassenbaum des Horbachschen Modells

2.2.2.3 Paketzuordnung

Mittels Filterregeln kann nun festlegt werden, welche Pakete einer Klasse zugeordnet werden. Listing 2.2 zeigt die Anwendung von verschiedenen Filtertypen.

```
1 tc filter add dev eth0 parent 1: protocol ip prio 10 \
2     u32 match ip dport 4662 0xffff flowid 1:4
3 tc filter add dev eth0 parent 1: protocol ip prio 9 \
4     handle 42 fw flowid 1:3
5 tc filter add dev eth0 parent 1: protocol ip prio 10 \
6     u32 match ip dst 192.168.1.42 action drop
```

Listing 2.2: Beispiele *tc filter*

Die erste Filterregel ordnet Pakete mit dem Zielport 4662 der Klasse 1:4 zu. Dazu wird der u32-Filter benutzt, der beliebige Bitmuster des IP-Kopfes auswerten kann. Die zweite Regel benutzt den fw-Filter, welcher die durch *iptables* gesetzten Markierungen auswertet. In diesem Fall werden Pakete mit der Markierung 42 der Klasse 1:3 zugeordnet. Die letzte Filterregel stellt eine Neuerung aus dem Jahr 2005 dar. Sie nutzt ebenfalls den u32-Filter, um Pakete mit dem Ziel 192.168.1.42 zu selektieren, aber statt diese einer Klasse zuzuordnen wird eine Aktion, hier *drop* also Wegwerfen, ausgelöst. Die Parameter *parent* und *prio* geben an, an welchem Knoten des Baumes die Regel anzulegen ist und welche Priorität sie besitzt, wobei numerisch kleinere Zahlen eine höhere Priorität bedeuten. Diese wenigen Beispiele können nur andeuten, welche latente Mächtigkeit allein die Filterregeln besitzen. So lassen sich in neueren Kernels (etwa ab 2.6.9) weitere Filtertypen wie *ematch* für erweiterte Mustererkennung mit logischen Operatoren finden.

Sowohl beim Horbachschen Modell als auch bei der ersten Realisierung des späteren Durchschnittsmodells wurden die Nutzer mittels u32-Filter anhand ihrer IP-Adresse einer Klasse zugeordnet. Leider hat dieser Filter keinen Negationsoperator. Neben der schlechten Benutzbarkeit des Kommandozeilentools *tc*, wie z. B. die hexadezimale Anzeige des Filtermusters relativ zum Beginn des IP-Kopfes, war dies der entscheidende Grund die Selektion stattdessen mittels des fw-Filters vorzunehmen. Die Pakete werden hier anhand einfacher 32 Bit-Zahlen (fwmarks) eingeordnet. Die Markierung, welche nur innerhalb des Kernels sichtbar ist und das Paket in keiner Weise verändert, erfolgt mit den wesentlich komfortableren *iptables* Regeln.

2.3 Bewertung des Horbachschen Modells

Die Einführung des ersten Shapingmodells hat deutlich gezeigt, daß Bandbreitenbegrenzung auch ein psychologisches Problem ist. Schon die zuvor eingesetzten Grenzen führten dazu, daß einige Nutzer sich in ihrer Freiheit beschränkt fühlten. Andere wiederum empfanden die nun aktive Regelung als Einführung eines Klassensystems, welches die Menschen in gut und böse kategorisiert. Im Zuge dessen wird seitdem vom CSN nur noch der Term Gruppen statt Klassen verwendet. (Dennoch soll hier der technisch korrekte Term „Klasse“ verwendet werden.) Als besonders unfair wurden jedoch die Anomalien bewertet, die in den Beispielen im Abschnitt 2.1.2 angeführt wurden.

Im Gegensatz zu den deutlich sichtbaren Mißständen in der Einordnung der Nutzer, blieben die Probleme in der globalen Regelung weitestgehend unbemerkt. Umso deutlicher wurden sie für die Administratoren des CSN-Teams. Das Horbachsche Modell nutzt zur Globalregelung ein Monatskontingent ohne Nutzung von Vorwissen. Symptomatisch für diese Regelung ist der rapide Abfall der Gesamtbandbreite unter Last gegen Ende jedes Monats (Abbildung 2.2) und die nicht garantierbare Einhaltung des Monatslimits. Warum dies nicht garantiert werden konnte, zeigt Abbildung 2.5.

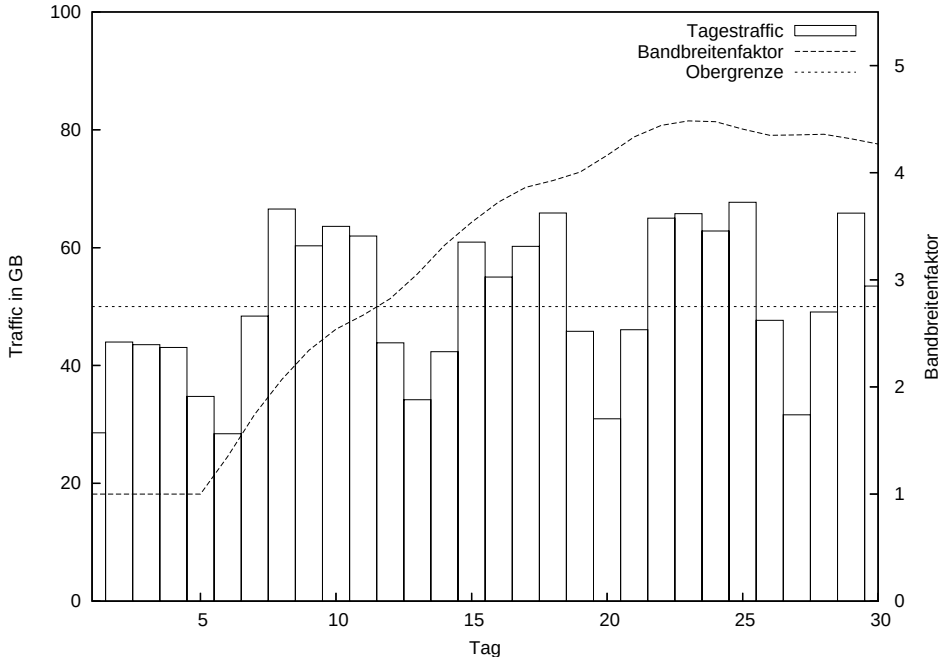

Abbildung 2.5: Shapingverlauf im April 2002

Der Monat beginnt programmgemäß mit dem Bandbreitenfaktor eins. Bedingt, beispielsweise, durch den Semesterbeginn bleibt der Verbrauch in den ersten fünf Tagen unter dem theoretischen Tageslimit. Doch schon in der folgenden Woche steigt der Verbrauch überdurchschnittlich an und übersteigt an mehrenden Tagen in Folge das Limit. Die Hochrechnungen können aber diesen Verlauf nicht wiederspiegeln und der Bandbreitenfaktor steigt. Auch im übrigen Monat ergeben die Prognosen ein positives Gesamtergebnis. Der akkumulierte Verbrauch des gesamten Monats liegt nur geringfügig über dem zulässigen Kontingent. Es ist leicht nachvollziehbar, daß bei etwas ungünstigerer Verteilung der Wochentage (ein Tag mit hohem anstatt niedrigem Verbrauch) ein deutliches Überschreiten ohne weiteres möglich ist.

In einem Worst-Case-Szenario könnte es dadurch zu einer Überschreitung des Monatslimits um 30 % oder mehr kommen. In diesem hypothetischen Szenario liegt die erste Hälfte des Monats in der vorlesungsfreien Zeit und die zweite außerhalb davon. Es wird weiterhin angenommen, daß in den ersten 14 Tagen der Verbrauch nah dem Limit liegt ohne es zu überschreiten. Danach erfolgt eine schrittweise Erhöhung des Verbrauchs durch die zurückkehrenden Studierenden. Wie Abbildung 2.6 zeigt, würde der Bandbreitenfaktor in diesem Zeitraum zunächst über den Startwert von eins steigen. In der zweiten Phase kommt es bedingt durch den erhöhten Verbrauch zum schnellstmöglichen Sinken des Faktors. Er kann aber durch die Linearität der Regelung nicht schnell genug fallen, um das Ergebnis noch zu beeinflußen. Die Folge ist eine Überschreitung des Kontingentes um ein Drittel.

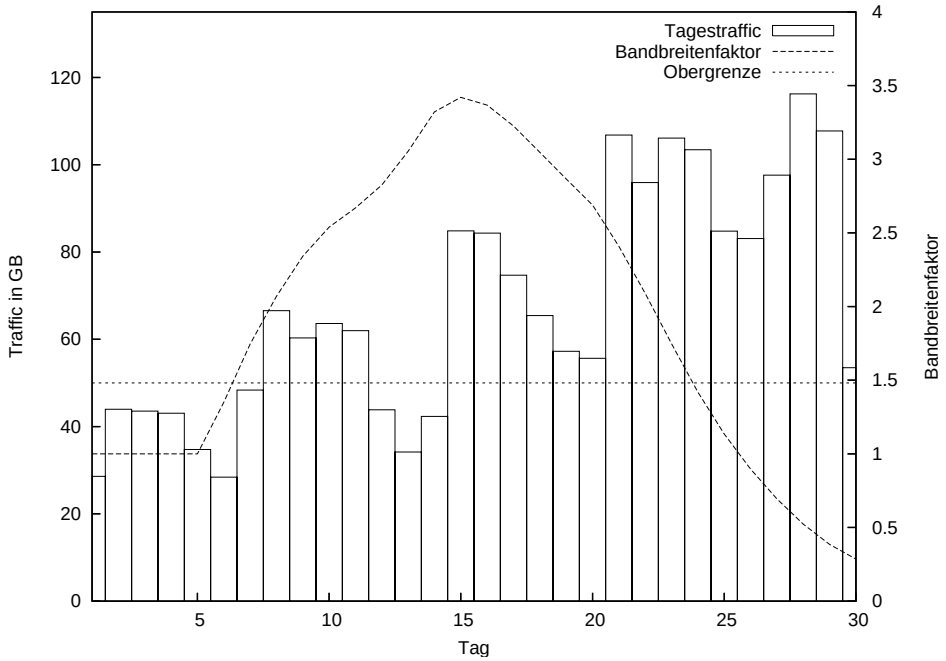

Abbildung 2.6: Hypothetisches Worst-Case-Szenario

Glücklicherweise kam es während der gesamten Einsatzdauer des Horbachschen Systems nur zu zwei solchen Fällen (April und Mai 2002). Die Diskrepanzen lagen jeweils unter 10 %. Trotz aller Probleme, die bei jeder Einführung einer neuen Software auftreten, war die Einführung einer automatischen Bandbreitenregelung ein Erfolg und ein wesentlicher Fortschritt auf wissenschaftlichem Gebiet.

3 Durchschnittsverfahren

Die Probleme des gerade erst eingeführten Shapings waren offensichtlich, dennoch blieb das System in Betrieb, da die Rückkehr zu Verwarnungen und Sperrungen ein noch größerer Rückschritt gewesen wäre. Um auch den Nutzern die Möglichkeit zu geben, ihre Ideen zur Verbesserung des Systems einzubringen, fand im November 2001 eine Diskussionsrunde zu diesem Thema statt. Hierbei wurde unter anderem auch die Idee eines Durchschnittsmodells eingebracht. In diesem sollte im Vordergrund das Prinzip stehen, daß der Nutzer mit dem höchsten Verbrauch auch die geringste Bandbreite erhält. In den folgenden Monaten erarbeitete Norbert Scheibner auf Grundlage von Simulationen [11] ein mögliches Konzept für ein durchschnittsbasiertes Verfahren. In vielen Gesprächen reifte das Konzept zu einem einsatzfähigen Prototypen heran. Ein Jahr nach der Einführung des Shapings im CSN wurde das Modell aus der Diplomarbeit im Oktober 2002 durch das Durchschnittsverfahren abgelöst.

3.1 Nutzereinordnung

Um ein Maximum an Fairness zu erreichen wurde die Nutzereinordnung vom Tagesverbrauch in einen Durchschnittswert umgewandelt. Hierbei kommt der durchschnittliche Verbrauch der letzten 14 Tage zur Anwendung. Bei der Auswahl des Zeitraumes wurde berücksichtigt, daß ein zu kurzes Intervall eine geringere Aussagekraft über das typische Nutzerverhalten hat und ein zu langes Intervall den Nutzer unnötig lang bestraft. Die Auswahl des Intervalls von 14 Tagen erfolgte daher eher aus subjektivem Empfinden, als aus wissenschaftlichen belegten Daten. Die Erfahrungen im Einsatz haben gezeigt, daß dieses Intervall seinen Zweck gut erfüllt, jedoch existieren keine Meßwerte oder Forschungsbelege, die seine Optimalität belegen. So wäre es durchaus denkbar, daß ein Intervall von 10 oder 15 Tagen besser geeignet wäre. Nicht zuletzt wurden 14 Tage auch aus dem Grund gewählt, daß sie dem Nutzer einfach zu erklären sind.

3.1.1 Erhöhung der Granularität

Durch das bereits bestehende Accounting war es einfach die Ermittlung des Verbrauchs eines Nutzer vom Tageswert auf den Durchschnittswert anzupassen. Problematisch jedoch gestaltete sich die Zuordnung von Verbrauch zu Bandbreite, da nun ein bestimmter Wert nicht automatisch einer bestimmte Klasse entsprach. Zunächst sollte jedem Nutzer eine individuelle Bandbreite entsprechend seines Verbrauchs zugeordnet werden. Es zeigte sich aber, daß dies nicht zu realisieren war, da man den Verbrauch

nicht direkt in eine Bandbreite umrechnen kann. (Natürlich ist dies möglich, aber die Verwendbarkeit des Ergebnisses ist fraglich.) Besonders die Frage, wieviel mehr beziehungsweise weniger Bandbreite pro verbrauchten Megabyte (MB) zuzuteilen ist, führte zum Verwerfen der Idee. Theoretisch stellt es einen Unterschied dar, ob dem Nutzer 1789 kBit/s oder 1812 kBit/s zur Verfügung stehen. Praktisch ist für ihn kein Unterschied feststellbar und die Auswirkungen auf seinen Verbrauch sind eher stochastischer Natur. Letzendlich läßt der Durchschnittswert keine Rückschlüsse darüber zu, ob das Volumen am Stück oder abschnittsweise verbraucht wurde.
Nachdem es sich als unpraktikabel herausgestellt hat, individuelle Nutzerbandbreiten zu berechnen, mußte auf andere Möglichkeiten ausgewichen werden. Die Klassenanzahl wurde vorsichtig auf zehn erhöht. Dies verbesserte die Granularität der Regelung und begrenzte die zusätzliche Last auf den damals recht schwach bestückten Shapingrechner (Pentium III mit 400 MHz).

3.1.2 Berechnung der Nutzereinordnung

Um eine Zuordnung der Nutzer auf die nun 10 Klassen zu realisieren, hatte Mirko Parthey die Idee, daß jede Klasse den gleichen Anteil am Gesamtverbrauchs enthält. Dazu werden die Durchschnittswerte aller Nutzer summiert und durch die Anzahl an Klassen geteilt (Gleichung 3.1).

$$t_k = \frac{t_{14}}{n} \qquad \begin{array}{lcl} t_k & - & \text{Verbrauch pro Klasse} \\ t_{14} & - & \text{Traffic aller Nutzer in den letzten 14 Tagen} \\ n & - & \text{Anzahl Klassen} \end{array} \tag{3.1}$$

Anschließend werden die Nutzer in absteigender Sortierung ihres Verbrauchs den Klassen, beginnend mit der langsamsten, zugeordnet. Überschreitet der akkumulierte Verbrauch der Nutzer in dieser Klasse die berechnete Grenze des Verbrauchs pro Klasse, wird mit der nächstschnelleren Klasse fortgesetzt. Dies wird bis zur zweitschnellsten Klasse wiederholt. Die verbleibenden Nutzer werden der Klasse 1 zugeordnet. Diese Vereinfachung ergibt sich aus der Tatsache, daß den bereits bearbeiteten Nutzer (n-1)/n-tel des Verbrauchs zugeordnet wurden und die verbleibenden Nutzer daher nur 1/n-tel des Verbrauchs verursacht haben können (Abbildung 3.1).

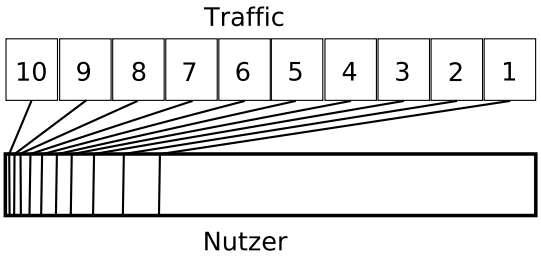

Abbildung 3.1: Zuordnung Nutzer - Klasse

3.1.2.1 Auswirkung der Änderung an der Nutzereinordnung

Die Änderung an der Nutzereinordnung wurde unter der Prämisse der Fairness gemacht. Aus Statistiken und Erfahrungen im Nutzersupport war bekannt, daß einige Nutzer das bisherige Shapingsystem entweder zu ihrem Vorteil ausnutzten oder dadurch ungerechterweise bestraft wurden. Aus den laufenden Accounting-Werten konnte ermittelt werden, welche Auswirkung die Änderung haben würde. Wie in Tabelle 3.1 ersichtlich ist, kann man von den Nutzern oberhalb der Hauptdiagonale sagen, daß für sie durch die Änderung eine Verbesserung erreicht wurde. Es handelt sich dabei um die Gruppe der Nutzer, die nur einmalig ein erhöhtes Datenaufkommen zu verzeichnen hatten. Bei den Nutzern unterhalb der Hauptdiagonale ist das Gegenteil der Fall. Diese werden erst durch die Änderung ihrem Traffic entsprechend eingeordnet und können nicht länger durch geschicktes Handeln das System überlisten.

			vorher Klasse					
neue Klasse	Nutzer	Prozentsatz	0	1	2	3	4	5
1	949	55,4	925	11	7	4	2	0
2	222	13,0	193	18	7	2	1	1
3	151	8,8	113	23	5	8	1	1
4	111	6,5	53	24	18	8	8	0
5	82	4,8	27	12	33	6	3	1
6	64	3,7	5	10	25	11	8	5
7	49	2,9	4	4	15	14	4	8
8	39	2,3	0	0	9	16	9	5
9	30	1,8	1	1	4	14	5	5
10	16	0,9	0	0	1	2	11	2

Tabelle 3.1: Verschiebung der Nutzer nach Änderung der Nutzerzuordnung

Wenn man ein Monatsvolumen von einem Gigabyte pro Nutzer als Ausgangswert annimmt, welches komplett ausgeschöpft wird, so läßt sich der Durchschnittsverbrauch pro Nutzer pro Tag errechnen. Gegeben sind die Nutzerzahlen in Tabelle 3.1 und die Nutzereinordnung, welche garantiert, daß von jeder Klasse nur 1/n-tel des Kontingents verbraucht wird. Der Mittelwert der Ergebnisse in Tabelle 3.2 liegt bei 34 MB pro Nutzer pro Tag. Die Durchschnitte lassen jedoch keinen Rückschluß darauf zu, wie und wann der Traffic entstanden ist. So ist es durchaus möglich, daß ein Nutzer in Klasse 2 ein CD-Image heruntergeladen hat und den Rest des Monats sich wie ein Nutzer in Klasse 1 verhalten hat.

3.2 Globalregelung

Neben der Änderung der Nutzereinordnung wurde auch die Neuberechnung des Bandbreitenfaktors verändert. Um das Problem der gegen Monatsende stark sinkenden Bandbreite zu lösen, muß schnel-

Klasse	Nutzer	Verbrauch pro Tag in MB	Gesamtverbrauch pro Nutzer in MB
1	949	6,16	184,84
2	222	26,34	790,14
3	151	38,72	1161,66
4	111	52,68	1580,28
5	82	71,31	2139,16
6	64	91,36	2740,80
7	49	119,33	3579,82
8	39	149,92	4497,72
9	30	194,90	5847,04
10	16	365,44	10963,20

Tabelle 3.2: Durchschnittsverbrauch pro Nutzer

ler auf Abweichungen vom Regelverbrauch reagiert werden. Von zwei vorgeschlagenen Ansätzen [5] wurde die Berechnung von Norbert Scheibner gewählt.

3.2.1 Berechnung der Initialbandbreiten

Für die Regelung ist es unerlässlich eine initiale Bandbreite für jede Klasse festzulegen, die später mit dem Bandbreitenfaktor beaufschlagt werden kann. Während diese im Horbachschen System noch willkürlich gewählt waren, werden sie nun durch Berechnung ermittelt. Auch hier kommt wieder die Idee zum tragen, daß jede Klasse exakt 1/n-tel des Kontingentes verbrauchen soll. In den folgenden Berechnungen wird exemplarisch ein Monatsvolumen von 1000 GB angenommen.

$$
\begin{aligned}
t_{cls} &= \frac{t_{limit}}{n} \\
t_{cls} &= \frac{1000\,\mathrm{GB}}{10} \\
&= \underline{\underline{100\,\mathrm{GB}}}
\end{aligned}
\qquad
\begin{aligned}
t_{cls} &\quad - \quad \text{Limit pro Klasse} \\
t_{limit} &\quad - \quad \text{Monatslimit}
\end{aligned}
\tag{3.2}
$$

Die langsamste Klasse erhält eine Bandbreite, bei welcher nicht mehr als 1/n-tel, selbst bei maximaler Auslastung, verbraucht werden kann. Diese Voraussetzung wurde unter der Annahme getroffen, daß die Nutzer dieser Klasse, in der Regel sogenannte Power-User, trotz der Einschränkungen ihre verbleibende Bandbreite weiterhin vollständig nutzen. Sie wurde durch den Fakt gestützt, daß diese Nutzergruppe nach der Umstellung der Nutzereinordnung 10 % des Gesamtverbrauchs für sich beansprucht. Die Berechnung erfolgt anhand Gleichung 3.3.

$$
\begin{aligned}
bw_n &= \frac{t_{cls} \cdot 8\,\text{Bit}}{30\,\text{d} \cdot 86400\,\text{s}} \\
bw_{10} &= \frac{100\,\text{GB} \cdot 1024^2 \cdot 8\,\text{Bit}}{30\,\text{d} \cdot 86400\,\text{s}} \\
&\approx \underline{\underline{324\,\text{kBit/s}}}
\end{aligned}
\qquad
\begin{aligned}
n \quad &- \text{Anzahl Klassen} \\
bw_n \quad &- \text{Bandbreite n-te Klasse} \\
bw_{10} \quad &- \text{Bandbreite Klasse 10}
\end{aligned}
\tag{3.3}
$$

Für die Bandbreite der schnellsten Klasse wird der Einfachheit halber die Bandbreite des Uplinks herangezogen. Den Nutzern in Klasse eins werden somit anfänglich keine Beschränkungen auferlegt, da ihr Verbrauch vergleichsweise gering ausfällt. Dies folgt aus der Nutzerzahl nach Verteilung auf die Klassen (siehe Abschnitt 3.1.2). Für die übrigen Klassen wurde daraus geschlossen, daß ihre Auslastung zwischen diesen beiden Extremen liegen muß. Um die Initialbandbreiten zu erhalten, wurde eine Korrelation zwischen der Nutzerzahl jeder Klasse und ihrer Auslastung hergestellt. Trotz gleichen Verbrauchs (jeweils 1/n-tel) steigt die Nutzerzahl stetig an (siehe Tabelle 3.2). Daraus wurde abgeleitet, daß sich die Auslastung umgekehrt proportional zur Anzahl der Nutzer in der Klasse verhält. Anders ausgedrückt: Je höher (numerisch kleiner) die Klasse, desto geringer die Auslastung. Ein Näherungswert dieses variablen Proportionalitätsfaktors wurde mittels einer geometrische Folge errechnet.

$$
bw_{n-i} = bw_n \cdot q^i \quad \text{für } 1 \leq i < n
\qquad
\begin{aligned}
q \quad &- \text{Verhältnis} \\
n \quad &- \text{Anzahl Klassen} \\
bw_n \quad &- \text{Initialbandbreite n-te Klasse}
\end{aligned}
\tag{3.4}
$$

umgestellt nach q

$$
q = \sqrt[i]{\frac{bw_{n-i}}{bw_n}} \tag{3.5}
$$

für $n = 10$ und $i = 9$ ergibt sich

$$
\begin{aligned}
q &= \sqrt[9]{\frac{bw_1}{bw_{10}}} \\
&= \sqrt[9]{\frac{100\,\text{MBit/s}}{0,324\,\text{MBit/s}}} \\
&\approx \underline{\underline{1,9}}
\end{aligned}
$$

Die Klassen werden ohne die Möglichkeit zu Borgen angelegt, da die Wirksamkeit der Regelung sonst nicht gegeben wäre. Bei theoretischer Vollauslastung aller Klassen würde die aggregierte Bandbreite die Interfacebandbreite um etwa das Doppelte überschreiten. Hier wäre es denkbar, die Initialbandbrei-

Klasse	Bandbreite	Verhältnis q	Bandbreite Klasse n-1
10	324 KBit/s	1,9	612 KBit/s
9	612 KBit/s	1,9	1157 KBit/s
8	1157 KBit/s	1,9	2,2 MBit/s
7	2,2 MBit/s	1,9	4 MBit/s
6	4 MBit/s	1,9	8 MBit/s
5	8 MBit/s	1,9	15 MBit/s
4	15 MBit/s	1,9	28 MBit/s
3	28 MBit/s	1,9	53 MBit/s
2	53 MBit/s	1,9	100 Mbit/s
1	100 MBit/s	-	-

Tabelle 3.3: Initialbandbreiten bei 10 Klassen, 1000 GB Monatslimit und 100 MBit/s Uplink

te zu halbieren, um eine Überlastsituation bei Vollast zu verhindern. Praktisch ist ein solches Vorhaben zwecklos, da nach der Beaufschlagung der Klassen mit dem Bandbreitenfaktor das Ergebnis ad absurdum geführt würde. Desweiteren war es in der Vergangenheit ausreichend weniger als 20 % der Nutzer (entspricht etwa den Klassen 8 bis 10) einzuschränken, um die Einhaltung des Monatslimits zu gewährleisten.

3.2.2 Berechnung Bandbreitenfaktor

Das wichtigste Element der Regelung und Stellgröße des gesamten Regelkreislaufs ist der Bandbreitenfaktor. Während man die Nutzereinordnung als Mikromanagement betrachten kann, wird das Makromanagement über den Bandbreitenfaktor realisiert. Die geänderte Regelung basiert auf einer Simulation von Norbert Scheibner [11]. Ideen und Ratschläge zur Verbesserung stammen von Jan Horbach, Thomas Dotschuweit und den Gebrüdern Parthey. Im Vergleich zum Horbachschen System unterscheidet sich die Regelung vor allem dadurch, daß keine monatsweise Betrachtung mehr vorgenommen wird. Stattdessen wird das Kontingent auf ein Tagesvolumen umgerechnet, welches die Regelung einzuhalten versucht.
Das zulässige Tagesvolumen wird trivial berechnet, in dem das Monatskontingent durch die Anzahl der Tage im aktuellen Monat dividiert wird.

$$t_d = \frac{t_{limit}}{d_m} \qquad \begin{array}{lcl} t_d & - & \text{zulässiges Tagesvolumen} \\ t_{limit} & - & \text{Monatslimit} \\ d_m & - & \text{Anzahl der Tage im Monat} \end{array} \tag{3.6}$$

Aus den Daten des Accountings werden durch Summierung des Verbrauchs aller Nutzer der Durchschnittsverbrauch der letzten 14 Tage ermittelt. Hierbei wird nur das von extern verursachte Datenvo-

lumen (Download) zur Berechnung herangezogen.

$$k = \frac{t_d}{t_{avg_{14}}} \qquad \begin{array}{lcl} k & - & \text{Korrekturfaktor} \\ t_d & - & \text{zulässiges Tagesvolumen} \\ t_{avg_{14}} & - & \text{14-Tage-Durchschnittsverbrauch} \end{array} \tag{3.7}$$

Der Korrekturfaktor wird gemäß Gleichung (3.7) berechnet. Durch Multiplikation des Korrekturfaktors mit dem Bandbreitenfaktor des Vortages erhält man den neuen Bandbreitenfaktor, der mit der Initialbandbreite jeder Klasse multipliziert wird. Daraus resultiert die neue Bandbreite für jede Klasse.

$$b_{neu} = b_{alt} \cdot k \qquad \begin{array}{lcl} b_{neu} & - & \text{neuer Bandbreitenfaktor} \\ b_{alt} & - & \text{Bandbreitenfaktor des Vortages} \\ k & - & \text{Korrekturfaktor} \end{array} \tag{3.8}$$

Um auf starke Anstiege im Verbrauch zu reagieren, ist die lineare Regelung allerdings zu schwach. Da dem Durchschnittsmodell keine Vorhersagewerte zur Verfügung stehen, muß bei Bedarf vorsorglich die Bandbreite stärker eingeschränkt werden. Vergeht zuviel Zeit zwischen Anstieg und Reaktion, kann es passieren, daß trotz weiterer Abregelung eine Überschreitung des Monatslimit nicht mehr zu verhindern ist. Um ein stärkere Regelung zu erreichen, wird der Anstieg mit einem Hebel kombiniert. Zur Erkennung eines ansteigenden Verbrauches muß ein kürzerer Zeitraum betrachtet werden. Hierfür wurden sieben Tage als geeigneter Zeitrum gewählt, um eine enstsprechende Tendenz zu erkennen. Der Anstieg, als Differenzierglied bezeichnet, errechnet sich aus dem Quotient der Werte für den 7- und 14-Tage-Durchschnitt.

$$g = \frac{t_{avg_7}}{t_{avg_{14}}} \qquad \begin{array}{lcl} g & - & \text{Differenzierglied (Anstieg)} \\ t_{avg_7} & - & \text{7-Tage-Durchschnittsverbrauch} \\ t_{avg_{14}} & - & \text{14-Tage-Durchschnittsverbrauch} \end{array} \tag{3.9}$$

Der Einsatz des verstärkten Korrekturfaktors ist nur dann nötig, wenn der Anstieg über einem Schwellwert liegt und der aktuelle Verbrauch sich in der Nähe des zulässigen Tagesvolumens bewegt. Aus diesen Vorgaben resultiert folgende Bildungsvorschrift:

Wenn das zulässige Tagesvolumen in den vorangegangenen sieben Tagen (ausgehend vom aktuellen Tag) mindestens einmal überschritten wurde und das Differenzierglied $g \geq 1,01$, dann

$$k_f = \frac{k}{g^4} \qquad \begin{array}{lcl} k_f & - & \text{verstärkter Korrekturfaktor} \\ k & - & \text{Korrekturfaktor} \\ g & - & \text{Differenzierglied (Anstieg)} \end{array} \tag{3.10}$$

Bei einem positiven Anstieg verkleinert sich dadurch der Korrekturfaktor überproportional und erfüllt somit den Zweck einer vorsorglichen Abregelung bei plötzlichen Veränderungen im Nutzerverhalten.

3.3 Klassenhierarchie

Die Regelung geht von der Voraussetzung aus, daß ihre Berechnungen durch die QoS Implementierung auch in der Realität umgesetzt werden kann und eine Umgehung nicht möglich ist. Im Abschnitt 2.2.2.2 wurde festgestellt, daß die verwendete Klassenstruktur nicht in der Lage ist, dies zu garantieren. Wie dort ebenfalls dargelegt wurde, konnte dieses Problem aufgrund des Leistungsmangels des Shaping-rechners nicht behoben werden. Der Wechsel der technischen Implementation von CBQ auf HTB konnte bereits wesentliche Verbesserungen der Geschwindigkeit erreichen. Der nächste logische Schritt bestand darin, allen Nutzer die gleichen Voraussetzungen zu bieten und die noch vorhanden Schlupflöcher zu schließen. Um dies zu erreichen, mußte der bestehende Klassenbaum erweitert werden.

3.3.1 Paketscheduling

Die technische Realisierung mittels HTB erfordert die Angabe einer zugesicherte Rate für jede Klasse. Beim Senden eines Paketes wird zunächst geprüft, ob das Paket noch mit der zugesicherten Rate gesendet werden kann. Würde diese überstiegen, wird geprüft, ob die angeforderte Rate kleiner als die Spitzenrate ist. Falls ja, wird versucht von der Elternklasse Bandbreite zu borgen, andernfalls wird das Paket in die Warteschlange eingeordnet.

Um eine Gleichheit zwischen den Nutzern zu schaffen, ist es wünschenswert, daß jeder Nutzer nach der Round-Robin-Methode bedient wird. Eine Näherung dieses Verfahrens könnte mittels Stochastic Fair Queueing (SFQ) erreicht werden. Die von diesem Verfahren verwendet Hashbildung über Quell- und Zieladresse, sowie Quell- und Zielport, kann keine absolute Gleichheit schaffen, da nur 127 Warteschlangen existieren und Nutzer mit mehreren IP-Adressen weiterhin bevorteilt würden. Die Lösung dieses Problems verbirgt sich in der Realisierung des Borgens im HTB. Die Zuteilung von Bandbreite an die Kindklassen wird durch Round-Robin vorgenommen. Wie schon im Abschnitt 2.2.2 beschrieben, wird die Granularität des Borgens durch das Quantum bestimmt. Die Größe des Quantums wird, wenn nicht explizit angegeben, vom Kernel automatisch durch den r2q-Faktor bestimmt. Da das Senden paketweise erfolgt, muß das Quantum größer oder gleich der MTU, also 1500 Bytes bei Ethernet, sein. Andernfalls führen die Diskrepanzen zu Rechenfehlern in der Regelung, wenn beispielsweise 1500 Bytes gesendet wurden, obwohl das Quantum nur 1000 Bytes beträgt.

3.3.2 Zusätzliche Nutzerklassen

Wie bereits angesprochen, kann durch die Einführung einer zusätzlichen Klasse pro Nutzer die gewünschte Fairness erreicht werden (Abbildung 3.2). Damit das Round-Robin-Verhalten überhaupt angewendet wird, muß die Klasse jedes Nutzers mit so wenig Bandbreite ausgestattet werden, daß eine Nutzung effektiv nur durch Borgen möglich ist. Die Klassen werden daher mit einer Rate von einem kBit/s und der Bandbreite der eigentlichen Klasse als Spitzenrate angelegt. Damit ist gewährleistet, daß das Gesamtverhalten durch den Algorithmus des Borgens dominiert wird. Gleichzeitig garantiert das Scheduling, daß jeder Nutzer unabhängig vom Verhalten der anderen Nutzer einen fairen Anteil an der Bandbreite der Elternklasse erhält.

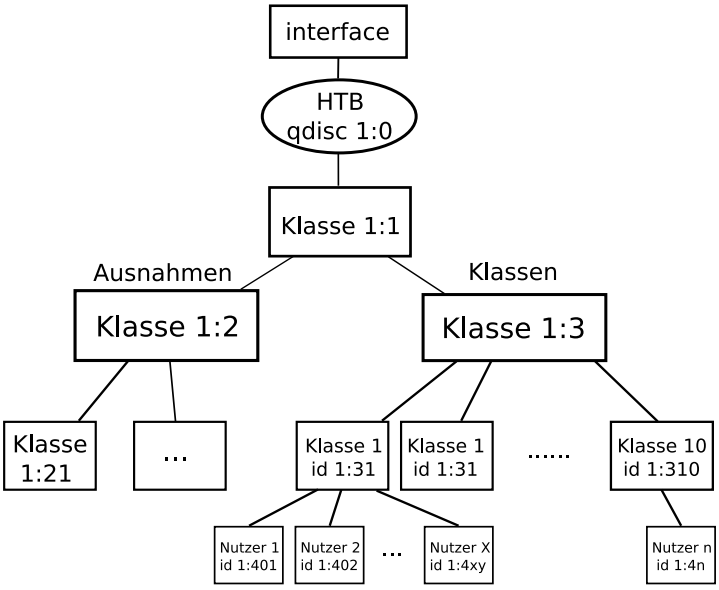

Abbildung 3.2: HTB-Baum des Durchschnittsmodells

Aufgrund der großen Unterschiede zwischen den Klassen eins bis zehn, sowie der Nutzerklassen, kann der r2q-Faktor nicht für jede Klasse ein passendes Quantum berechnen. Der r2q-Faktor von zehn errechnet ein Quantum von 1500 Bytes erst bei einer Rate von 120 kBit/s, die für das Shapingsystem viel zu hoch ist, da die Summe der zugesicherten Bandbreite der Nutzerklassen von Klasse zehn die Bandbreite der Elternklasse überstiege und somit das Shaping wirkungslos machen würde. Daher muß das Quantum explizit als Parameter angegeben werden, wenn die zugesicherte Rate von einem kBit/s verwendet wird.

3.3.3 Ausregelgrenzen für die Globalregelung

Wie im Abschnitt 3.2.2 erläutert, wird der berechnete Korrekturfaktor mit dem Bandbreitenfaktor multipliziert. Ohne Begrenzung ist es theoretisch möglich, daß der Bandbreitenfaktor soweit sinkt, daß er

durch Rundungsfehler zu Null wird. Während der vorlesungsfreien Zeit kann sich der Faktor wiederum soweit erhöhen,daß die Regelung nicht schnell genug reagieren kann, wenn der Verbrauch wieder ansteigt. Es müssen daher Grenzen festgelegt werden, die den Bandbreitenfaktor beidseitig beschränken. Die untere Grenze muß so berechnet werden, daß die Funktionalität der Regelung nicht beeinträchtig wird. Der angepasste Bandbreite einer Klasse muß daher größer oder gleich der Summe der Bandbreiten ihrer Kindklassen sein (vergleiche Abschnitt 3.3.2).

$$
\begin{aligned}
b_{min} &= \frac{bw_{min} \cdot u_n}{bw_n} \\
b_{min} &= \frac{1\,\text{kBit} \cdot 16}{324\,\text{kBit}} \\
&\approx \underline{\underline{0,0494}}
\end{aligned}
\qquad
\begin{aligned}
b_{min} &- \text{minimaler Bandbreitenfaktor} \\
bw_{min} &- \text{Minimumbandbreite} \\
u_n &- \text{Anzahl Nutzer in Klasse n} \\
bw_n &- \text{Initialbandbreite Klasse n}
\end{aligned}
\tag{3.11}
$$

Obwohl der minimale Bandbreitenfaktor nur anhand der Anzahl der Nutzer in Klasse n (hier Klasse zehn) ermittelt wird, kann und muß er auch als untere Schranke für die übrigen Klassen verwendet werden. Der Grund dafür ist, daß alle Klassen mit dem gleichen Bandbreitenfaktor multipliziert werden. Dies stellt kein Problem dar, da die Initialbandbreiten stärker (exponentiell) ansteigen als die Nutzerzahlen.

Die Obergrenze des Bandbreitenfaktors wird analog zur Untergrenze ermittelt. Ausschlaggebend ist an dieser Stelle die Bandbreite, bis zu welcher eine Regelung zweckmäßig erscheint. Im CSN existieren 10 und 100 MBit-Anschlüsse, wobei die Anschlüsse mit 10 MBit/s die Mehrheit bilden. Übersteigt die Bandbreite einer Klasse die Marke von 10 MBit/s, so wirkt die Beschränkung nur noch bei Vollauslastung durch mehrere Nutzer. Es wird aber davon ausgegangen, daß dies selbst unter optimalen Bedingungen nicht beziehungsweise nur teilweise möglich ist, z. B. wegen des Protokolloverheads von TCP/IP. Aufgrund dieser Überlegungen wurde ein Wert von 15 MBit/s als Maximumbandbreite gewählt. Die Obergrenze für den Bandbreitenfaktor ergibt sich wie folgt:

$$
\begin{aligned}
b_{max} &= \frac{bw_{max}}{bw_n} \\
b_{max} &= \frac{15\,\text{MBit}}{0,324\,\text{MBit}} \\
&\approx \underline{\underline{46,296}}
\end{aligned}
\qquad
\begin{aligned}
b_{max} &- \text{maximaler Bandbreitenfaktor} \\
bw_{max} &- \text{Maximumbandbreite} \\
bw_n &- \text{Initialbandbreite Klasse n}
\end{aligned}
\tag{3.12}
$$

Da eine Begrenzung der Klasse bei Überschreiten der Maximumbandbreite als nicht mehr zweckmäßig angesehen wird, ergibt sich daraus die Vereinfachung, daß auch keine Regeln mehr mittels *tc* angelegt werden müssen. Dank dieser Vereinfachung konnte die Last auf dem Shapingrechner weiter vermindert werden.

3.4 Zusammenfassung

Ziel der Verbesserungen, war neben einer erhöhten Variabilität und der Entwicklung eines Systems, welches lediglich von wenigen Kennwerten abhängig ist, das Erreichen einer höheren Nutzerakzeptanz. Über eine Webseite erfährt der Nutzer nur noch seinen Durchschnittsverbrauch und eine grobe Information über die ihm zur Verfügung stehende Bandbreite („Deine Bandbreite wird nicht beschränkt.“, „größer 10 MBit“, „kleiner 10 MBit“, „kleiner 5 MBit“ und „kleiner 1 MBit“). Kaum einem Nutzer ist bekannt, daß es überhaupt Klassen gibt. Durch die hohe Variabilität der Einordnung ist diese Information auch von geringer Aussagekraft, da jeder Nutzer relativ zum Verbrauch der anderen Nutzer eingeordnet wird. Dies läßt sich am ehesten mit einem Bewertungssystem, ähnlich denen, welche eBay oder Amazon einsetzen, vergleichen (1423 Nutzer haben weniger als Nutzer A verbraucht, 112 mehr als Nutzer A). Ein typisches Phänomen der geänderten Einordnung ist beispielsweise, daß ein Nutzer zu Beginn eines Tages in Klasse zehn ist und im Laufe des Tages bis in Klasse 1 aufsteigt, weil durch den gleitenden 14-Tage-Durchschnitt der Verbrauch, der Grund für die Einordung war, aus dem betrachteten Zeitfenster herausfällt.

Aus eigenem Erleben kann ich sagen, daß die Erfahrungen mit dem Durchschnittsmodell sehr gut waren und es auch immer noch sind. Die Anfragen an das CSN sind soweit zurückgegangen, daß sie nur noch technischer Natur sind und meist von anderen Einrichtungen stammen, die ein ähnliches System einsetzen möchten.
Die wesentlichen Vorteile gegenüber dem Horbachschen Modell sind:

- Wegfall der festen Limits
- faire Verteilung der Nutzer auf die Klassen
- keine durch Nutzer ausnutzbare Schwachstellen (weder durch viele Verbindungen, noch durch Ausreizung eines Limits)
- geringe Reaktionszeit bei Änderungen im Gesamtverbrauch
- kontinuierliche Regelung
- verkürztes Einordnungsintervall (stündlich statt täglich)
- Wegfall der Strafeinordnung (keine Bestrafung bei Mehrfachverstößen durch Wegfall der Limits)

Betrachtet man die technische Realisierung, so wurden die Abhängigkeiten, die durch die Verweildauer entstanden, eliminiert. Es war nun möglich die Neueinordnung bei Fehlern ohne Rückstellen der Datenbasis zu wiederholen. Die Ausregelgrenzen begrenzen die Regelung innerhalb eines Fensters. Sie verhindern somit ein Überschwingen, sorgen für eine schnellere Abregelung im Bedarfsfall und

entlasten den Shapingrechner, weil Shapingregeln nur noch unterhalb der Maximumbandbreite erstellt werden.
Die Nachteile des Durchschnittsverfahrens liegen in seinen Vorteilen. Weil dem Nutzer nur noch Näherungswerte über seine Bandbreite mitgeteilt werden, um Rückschlüsse auf das Gesamtlimit zu verhindern, verliert das System an Transparenz. Für die Wirksamkeit der Regelung mag dies bedeutungslos sein, für die Akzeptanz kann es eine entscheidende Rolle spielen. Dem CSN sind jedoch daraus bisher keine Problem erwachsen. Während des mehr als vierjährigen Einsatzes wurden einige Anomalien entdeckt, die nur unter bestimmten Bedingungen auftreten. Eine davon ist im Dezember beobachtbar, wenn der Verbrauch, für das System unerwartet, kurz vor Weihnachten schlagartig sinkt. Weil die Regelung kein Wissen über Feiertage besitzt, kann das Monatslimit nicht optimal genutzt werden. Eine Weiterentwicklung könnte das System um dieses Wissen erweitern. Abschließend läßt sich sagen, daß sich das System sehr gut bewährt hat.

4 Reimplementation

Der praktische Teil der vorliegenden Studienarbeit beschäftigte sich mit der Reimplementation der DynShaper-Software, um eine bessere Integration in die übrige CSN-Software zu erreichen und die erforderlichen Schnittstellen für zukünftige Weiterentwicklungen bereitzustellen. Für die Implementation wurde die Skriptsprache Perl verwendet, in der nahezu die gesamte CSN-Software geschrieben wurde.

4.1 Architektur des Systems

Bei der Analyse der bestehenden DynShaper-Sofware wurde festgestellt, daß alle Komponenten (Manager, Evaluator, Configurator und DynShaper) zu 95 % die gleichen Daten verarbeiteten, diese jedoch unabhängig voneinander aus der Datenbasis (Datenbank, Konfigurationsdatei) bezogen. Im schlimmsten Fall wurden die Daten bis zu dreimal initialisiert. Bei diesem Design waren Redundanzen nicht zu vermeiden, welche besonders fehlerträchtig bei Veränderungen waren, da es leicht passieren konnte, daß die Modifikationen bei einem oder gar mehreren Subsystemen vergessen wurden. Es ist leicht vorstellbar, daß tiefgreifende Änderungen an der Implementierung dadurch sehr fehleranfällig waren.

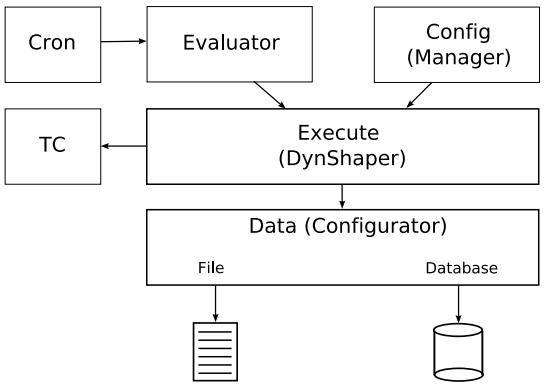

Abbildung 4.1: Architektur des neuen Systems

Die Reimplementierung mußte deshalb gewährleisten, daß die Software frei von redundanten Funktionen ist und die Schnittstellen unabhängig von der internen Realisierung sind. Perl bietet zwei Möglichkeiten, um Funktionalität zu kapseln und eine Software zu strukturieren: Unterprogramme und Methoden, Module und Klassen bzw. imperativ oder objektorientiert. Prinzipiell sind beide Ansätze

möglich, jedoch erfordert der imperative Ansatz die Einbindung (Import) jedes Moduls, dessen Funktion genutzt werden soll, während ein Objekt potentiell auf alle Methoden zurückgreifen kann, die seine Klasse und deren Elternklassen geerbt haben. Durch Objektorientierung ist es auch sehr einfach möglich, die Implementierung einer Funktion oder einer ganzen Klasse durch Überschreiben zu verändern. Das Ablegen der zu verarbeitenden Daten in den Attributen des Objekts befreit von der Notwendigkeit diese beim Methodenaufruf zu übergeben. Aufgrund dieser Vorteile und der dadurch gegebenen Erfüllbarkeit der genannten Anforderungen fiel meine Wahl auf den objektorientierten Ansatz.

4.1.1 Hierarchie der Klassen

Die Wahl einer geeigneten Vererbungshierarchie der Klassen bestimmt, auf welche Methoden das Objekt und die Methoden selbst zugreifen können. Für jede Klasse gilt, daß für alle Methoden nur die Methoden aufrufbar sind, die in der gleichen Klasse definiert sind oder ererbt wurden, jedoch nicht die Methoden ihrer Kindklassen. Die Vererbung muß daher so konstruiert werden, daß alle benötigten Funktionen innerhalb des Sichtbarkeitsbereiches (engl. scope) liegen.

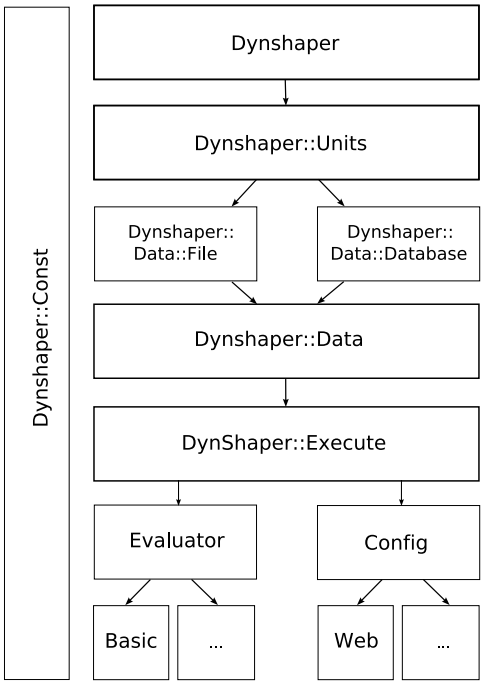

Abbildung 4.2: Hierarchie der Perlklasssen

Grundfunktionen, welche Lese- und Schreiboperation auf interne Datenstrukturen abstrahieren oder Einstellungen am Operationsmodus der Sofware vornehmen, müssen daher an der Spitze der Vererbung liegen. Speziellere Methoden, wie das Berechnen des Bandbreitenfaktors, können hingegegen in Blattklassen verlagert werden, wenn keine andere Methode von ihnen abhängig ist. Angewendet

auf alle Methoden ergibt sich eine Abhängigkeitskette, aus welcher, nach Gruppierung der funktional zusammengehörigen Methoden, eine Klassenhierarchie abgeleitet wurde (Abbildung 4.2).

4.1.2 Datenstruktur

Die Skriptsprache Perl erlaubt es, beinahe alles (Skalare, Arrays, Hashes, Funktionen, reguläre Ausdrücke) zu einem Objekt zu „weihen“ (engl. bless). Der Unterschied zwischen einem einfachen Datentyp und einem Objekt besteht lediglich darin, daß ein Objekt einen Vermerk über die Klassenzugehörigkeit enthält. Die einfachste und flexibelste Variante eines Perlobjektes stellen Hashes dar. Sie erlauben den wahlfreien Zugriff auf alle Schlüssel-Wert-Paare, sowie eine automatische Erweiterung der Datenstruktur bei Bedarf. Der Preis dieser Flexibilität ist ein erhöhter Speicherbedarf für die interne Verwaltung (Überlauflisten), der aber in Zeiten von mehreren Gigabyte großen Hauptspeichern vernachlässigbar ist. Arrays hingegen haben einen Geschwindigkeitsvorteils gegenüber Hashes, setzen jedoch das Wissen über den entsprechenden Index für einen wahlfreien Zugriff voraus. Für die Strukturierung der internen Attribute des Objektes wurden deshalb Hashes genutzt, weil ein Schlüsselwort sich leichter merken läßt als eine Zahl und gleichzeitig zu einer erhöhten Lesbarkeit und einem besseren Verständnis des Quelltextes beiträgt.
Neben der Wahl eines geeigneten Datentyps wurde Wert darauf gelegt, die Objektattribute möglichst so zu organisieren, daß sich ein Austausch zwischen Datenstruktur und Datenbasis, inbesondere einer Datenbank, so einfach wie möglich gestaltet. Die Adressierung einer einzelnen Information innerhalb einer Datenbank erfordert mindestens drei Parameter: Name der Tabelle, Primärschlüssel und Spaltenname. Eine äquivalente Abbildung wurde in der Implementierung durch einen dreistufigen Hash realisiert.

4.2 Beschreibung der Komponenten

4.2.1 Evaluator

Das Herzstück des DynShaper-Systems war und ist der Evaluator. Er berechnet den Bandbreitenfaktor und ordnet die Nutzer anhand ihres Durchschnittsverbrauches in Klassen ein. An der zugrundeliegenden Arbeitsweise wurden keine Modifikationen vorgenommen. Die sichtbaren Änderungen, neben der Portierung, sind daher lediglich struktureller Natur.

4.2.2 Data

Der eigenständige Configurator der früheren Software wurde zugunsten einer gemeinsam genutzten Abstraktionsschicht aufgegeben. Die *Dynshaper::Data*-Klasse bildet die Daten unabhängig von der

genutzen Darstellung (Datenbank, Konfigurationsdatei) in einer für alle Subsysteme zugänglichen Datenstruktur ab. Der direkte Zugriff auf die privaten Attribute ist nicht erwünscht und wird über entsprechende Methoden abstrahiert. Somit wird ermöglicht, in Zukunft auch andere Bereitstellungsformen zu verwenden. Um eine zuverlässige Initialisierung unter allen Umständen zu gewährleisten, wurde ein Sicherungsmechanismus eingebaut, welcher zunächst versucht die Datenbank zu kontaktieren und im Falle eines Fehlschlages auf die Konfigurationsdatei zurückgreift. In der Regel ist dies nur nach einem Neustart des Shapingrechners notwenig, weil die Datenbank erst zu einem späteren Zeitpunkt gestartet wird. Werden in dieser Phase Methoden aufgerufen, welche auf die Verfügbarkeit der Datenbank angewiesen sind, wird das Programm mit einer entsprechenden Fehlermeldung beendet. Sollte es im Regelbetrieb zu einem Datenbankausfall kommen, bleibt so eine Basisfunktionalität erhalten, bis der Schaden behoben ist.

4.2.3 Execute

Bisher setzte ein Shellskript die Parameter aus einer Konfigurationsdatei in *tc*-Befehle um. Nach der Überarbeitung übernimmt dies die Dynshaper::Execute-Klasse, welche analog das Anlegen und Löschen der Shapingklassen und -regeln realisiert. Ein Vorteil der neuen Lösung ist, daß das Neuanlegen der Klassen sofort nach dem Evaluator-Aufruf ohne Neuinitialisieren der Daten erfolgt.

4.2.4 Manager

Trotz der starken Vereinfachungen der Programmierschnittstellen ist es nicht praktikabel Parameteränderung über einen Shellzugang vorzunehmen. Die Überwachung und Konfiguration durch eine Webanwendung bietet ein wesentlich höheres Maß an Nutzerfreundlichkeit und Benutzbarkeit. Die Funktionalität des Manager wurde insofern beibehalten und lediglich an den CSN-Standard angepasst. Eine Erweiterung um eine graphische Visualisierung ist für die Zukunft angedacht.

5 Abschließende Bemerkungen und Zusammenfassung

Diese Arbeit beschäftigte sich mit den Weiterentwicklungen der DynShaper-Software, welche im CSN zur Einhaltung des Datenkontingentes eingesetzt wird. Die Software nutzt dazu weiterhin die Möglichkeiten des Linux-Kernels hinsichtlich Quality of Service. Der konsequente Einsatz von OpenSource Software in der Entwicklung von CSN-eigenen Lösungen garantiert Anpassungsfähigkeit und Erweiterbarkeit für zukunftige Arbeiten, welche bei proprietärer Software nicht gegeben wäre. Das Horbachsche Modell konnte nur so durch ein besseres und leistungsfähigeres Verfahren ersetzt werden.

Obwohl das Durchschnittsverfahren zunächst auf ungeprüften Annahmen basierte, haben sich diese in der Praxis bewährt. Als besonders sinnvoll hat sich die relative Nutzereinordung (verstecktes Ranking) erwiesen. Ein vergleichbares Verfahren konnnte bisher nicht gefunden werden, da üblicherweise[1] [2] [3] [4] jeder Nutzer getrennt betrachtet und reglementiert wird. Kommerzielle Anwendungen wie Packeteer [7] arbeiten hingegen meist im Bereich der Liveüberwachung von QoS-Parametern, um sicherzustellen, daß unternehmenskritische Anwendungen genügend Bandbreite erhalten. Im überarbeiteten Berechnungsmodell der DynShaper-Software soll jedoch ein Kontingent möglichst fair auf alle Nutzer verteilt werden. Damit ist der Bandbreitenbedarf einzelner Anwendungen irrelevant. Auch der Verbrauch des Einzelnen ist von sekundärer Bedeutung. Einzig das Erreichen des Regelziels zählt, für welches die Bandbreite des einzelnen Nutzers in Abhängigkeit seines Verbrauchs beschränkt wird. Diese Sichtweise macht die Shapinglösung des CSN nach wie vor zu einem Unikat.

Im praktischen Teil wurde die Implementierung vollständig überarbeitet und schrittweise ersetzt. Seit Dezember 2005 ist der Evaluator im aktiven Betrieb. Im April 2006 wurden die übrigen Teilsysteme abgelöst. Kleinere Probleme konnten Dank des Entwicklungsmodus und verbesserter Protokollierung schnell diagnositiziert und behoben werden. Das neue System arbeitet bisher sehr zuverlässig und fügt sich nun nahtlos in die übrige CSN-Software ein.

[1] http://www.stunet.tu-freiberg.de/?do=traffic
[2] http://www.mcn.htwm.de/mcn/regeln.htm
[3] http://www.hh.fh-stralsund.de/Home/HHNet/Transfervolumen
[4] http://www.fh-oow.de/projekte/StuWoNet/netzwerk/teilnahme.html

Ein Ziel der Neuimplementierung war, eine Grundlage für zukünftige Arbeiten zu schaffen. Hauptsächlich sollten sich diese damit befassen, die Anomalien, welche in Abschnitt 3.4 beschrieben wurden, zu beheben und weitere Optimierungsmöglichkeiten (Verhältnis der Initialbandbreiten) zu untersuchen.

Literaturverzeichnis

[1] Martin Devera: *Hierarchical Token Bucket Theory*, 2002.
URL `http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm`

[2] Sally Floyd und Van Jacobson: *Link-Sharing and Resource Management Models for Packet Networks*, *IEEE/ACM Transactions on Networking*, Bd. 3(4):S. 365–386, 1995.

[3] Alexander Glöckner: *Wireless-LAN im StudentenNetzwerk (CSN)*, 2005.
URL `http://archiv.tu-chemnitz.de/pub/2006/0040/`

[4] Jan Horbach: *Dynamische Bandbreitenbeschränkung mit QoS*, 2001.
URL `http://archiv.tu-chemnitz.de/pub/2001/0100/`

[5] Jan Horbach: *Aktuelle Entwicklungen des DynShapers*, 2002.
URL `http://archiv.tu-chemnitz.de/pub/2002/0040/`

[6] Bert Hubert *et al.*: *Linux Advanced Routing and Traffic Control HOWTO, Chapter 9.5.5*, 2004.
URL `http://www.lartc.org/howto/lartc.qdisc.classful.html#AEN1072`

[7] Packeteer Inc.: *Packet Shaper Family Datasheet*, 2006.

[8] Sebastian Junge: *Klassenbasierter Anschluss im Chemnitzer Studenten Netz*, 2005.
URL `http://archiv.tu-chemnitz.de/pub/2005/0168/`

[9] Markus Schade: *Netzentwicklung im CSN*, 2004.
URL `http://archiv.tu-chemnitz.de/pub/2004/0135/`

[10] Markus Schade und Daniel Schreiber: *A Packet filtering Concept based on iptables MARK's*, 2005.
URL `https://www.csn.tu-chemnitz.de/~marks/firewall.pdf`

[11] Norbert Scheibner: *Simulationen zur Verbesserung des DynShapers*, unveröffentlichte Berechnungen mittels Tabellenkalkulation.

Selbstständigkeitserklärung

Hiermit erkläre ich, daß ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Chemnitz, den 30. Mai 2006

Markus Schade

Anhang

A Programmdokumentation

A.1 Datenstruktur

```
'ds_gruppen_v2' => {
    'changed' => 0, # Tabelle geändert?
    'entries' => {
        '10' => { # Gruppe 10
            'params' => {
                'conf_rate' => {
                    'changed' => 0, # Attribut geändert?
                    'value' => '2982912'
                    },
                'conf_in' => {
                    'changed' => 0,
                    'value' => 'on'
                },

    [ ... ]

                'conf_factor' => {
                    'changed' => 0,
                    'value' => '0.1138'
                },
            },
    'changed' => 0 # Eintrag (Klasse/Gruppe) geändert?
}
```

Listing A.1: Auszug Datenstruktur

In den dreistufigen Hash wurden zusätzliche Ebenen zur leichteren Verarbeitung eingefügt. In jeder Stufe protokolliert ein Flag (`changed`), ob der überwachte Schlüssel geändert wurde. Änderungen in einer tieferen Ebene werden nach oben progagiert, das heißt nach einer Änderung in Ebene drei

werden auch die `changed`-Flags in Ebene eins und zwei gesetzt. Dadurch ist es effizient möglich, nur die tatsächlichen Änderungen zu extrahieren, ohne alle Schlüssel zu durchsuchen.
Für die Bereitstellung, Initialisierung und Speichern der Daten ist die *Dynshaper::Data*-Klasse zuständig. Für das Objekt wird der Zugriff auf Datenbank oder Konfigurationsdatei transparent über eine identischen Datenstruktur dargestellt. Der direkte Zugriff auf die Objektattribute ist außerhalb der *Dynshaper::Data*-Klasse nicht erwünscht. Um Lese- und Schreiboperationen an der Datenstruktur vorzunehmen, sind entsprechende Methoden (append, change, get, get_ptr, set) vorhanden. Diese werden aus der *Dynshaper*-Klasse geerbt.

A.2 Synopsis

A.2.1 Evaluator

```
1 #!/usr/bin/perl
2 use Dynshaper::Evaluator;
3
4 my $shaper = new Dynshaper::Evaluator('Basic');
5
6 # Protokollierung in Datei
7 $shaper->use_logfile(1);
8
9 # Nutzereinordnung und globale Berechnungen durchführen
10 $shaper->reclassify();
11
12 # Speichern der Änderungen
13 $shaper->commit();
```

Listing A.2: Evaluator Skript

A.2.2 Temporäre Änderung eines Parameters

```
1 #!/usr/bin/perl
2 use Dynshaper::Execute;
3
4 my $shaper = new Dynshaper::Execute;
5
6 # Umrechnung von kBit/s in Bit/s
7 my $bandwidth = $shaper->convert(100, 'kbit', 'bps');
8
```

```
9  # Setzen der Initialbandbreite von Klasse 10 auf 100 KBit/s
10 $shaper->change('groups',10,'conf_rate',$bandwidth);
11
12 # Klassen neu anlegen
13 $shaper->restart();
```

Listing A.3: temporäre Änderung der Bandbreite von Klasse 10

A.2.3 Entwicklungsmodus

```
1  #!/usr/bin/perl
2  use Dynshaper::Evaluator;
3
4  my $shaper = new Dynshaper::Evaluator('Basic');
5
6  # Entwicklungsmodus aktivieren
7  $shaper->devel(1);
8
9  # Trafficdaten in Datei zwischenspeichern
10 $shaper->use_traffic_dump(1);
11
12 # Neuberechnung
13 $shaper->reclassify()
14
15 # Speichern der Änderungen (Nur Protokollierung)
16 $shaper->commit();
```

Listing A.4: Entwicklungsmodus (keine Veränderung der Datenbank)

Jedes Subsystem ist einzeln nutzbar. Die Erzeugung eines Objektes ist durch die *new*-Methode in den Klassen *Dynshaper::Execute*, *Dynshaper::Config* und *Dynshaper::Evaluator* möglich[1].
Bei der Erzeugung eines Objekts einer dieser Klassen wird die von *Dynshaper::Data* ererbte Methode **load_data()** aufgerufen, um die interne Datenstruktur zu initialisieren. Die Methode prüft zunächst, ob ein Datenbankzugriff (*Dynshaper::Data:Database*) möglich ist und greift auf die Konfigurationsdatei (*Dynshaper::Data::File*) zurück, falls keine Datenbankverbindung hergestellt werden kann. Die

[1] Durch weitere Vererbung auch in Kindklassen

jeweilige Klasse wird anschließend in die Vererbungsstruktur eingefügt. Zur eigentlichen Initialisierung wird abschließend die **load_config()**-Methode der eben eingefügten Klasse aufgerufen.

A.3 Überschreibbare Methoden

Obwohl die Objektorierung prinzpipiell das Überschreiben beliebiger ererbter Methoden erlaubt, sind hier nur die gemeint, für die es sinnvoll erscheint. Abbildung 4.2 zeigt mehrere Kindklassen der Evaluator- und Config-Klasse. Diese überschreiben die Methoden ihrer Elternklassen, um das gewünschte Berechnungsmodell oder die Darstellung für ein bestimmtes Anzeigemedium zu realisieren. Welche Kindklasse benutzt werden soll, muß bei der Erzeugung des Objekts angeben werden. Dem Objekt wird dann die angegebene Kindklasse anstelle der Elternklasse zugewiesen. So ist ein Wechsel zwischen den Verfahren möglich.

A.3.1 Evaluator

reclassify_users()

Nimmt die Einordnung der Nutzer vor

reclassify_global()

Berechnet die globalen Parameter (Bandbreitenfaktor, Korrekturfaktor).

reclassify()

Ruft **reclassify_users()** und danach **reclassify_global()** auf.

A.3.2 Config

display()

Gibt mediumspezifische Anzeigedaten zurück (z. B. HTML)

A.4 Bezugsquellen

DynShaper

`http://rnvs.cs.tu-chemnitz.de/twiki/bin/view/Dynshaper/`

CSN OpenSource Software

`https://www.csn.tu-chemnitz.de/OpenSource/twiki/index.html`