DiffServ mit adaptiven Regeln als Management-Instrument

Jan Horbach, Uwe Hübner
Technische Universität Chemnitz

18. März 2009

Zusammenfassung

Dieser Beitrag soll zeigen, dass sich das heute vorhandene IP-QoS-Instrumentarium nicht nur zur Erfüllung der Anforderungen von Audio- oder Videoanwendungen eignet, sondern sich auch gezielt für ganz andere Management-Aufgaben einsetzen lässt. Mit einem intelligenten Bandbreitenmanagement wird ein wirtschaftlicher und fairer Umgang mit Netzressourcen erreicht. Dazu werden die verfügbaren Bandbreiten der Nutzer dynamisch entsprechend ihres Verkehrsaufkommens angepasst, so dass bestimmte Zielparameter wie z.B. Datenvolumen-Begrenzungen eingehalten werden.

1 Einleitung

Die bisher übliche Internet-Technologie leistet einen *Best-Effort*-Service. Bei dieser Gleichbehandlung allen Verkehrs können harte Bandbreiten- und Latenzzeit-Forderungen kaum erfüllt werden. Einen Ausweg sollten verschiedene Entwicklungen in Richtung *Quality of Service (QoS)* bringen (siehe auch [1, 2]).
Nach einer Definition, die 1997 auf dem Next Generation Internet (NGI) Workshop in Virginia geprägt wurde, wird damit die Unterscheidung von verschiedenen Verkehrsklassen und Services sowie deren unterschiedliche Behandlung beschrieben:

„[...] the group came to the conclusion that the best definition it could formulate for QoS was one that defined methods for differentiating traffic and services - a fairly broad brush stroke, and one that may be interpreted differently.“ [3, Seiten xv-xvi]

Es geht also einmal um qualitative Parameter wie z.B. eine verlässliche Datenlieferung, kein Datenverlust, garantierte Bandbreiten, minimale Verzögerung oder konstante Verzögerungscharakteristika (Begrenzung des Jitters).
Beim Aspekt *Services* hingegen werden Anwendungsprotokolle wie E-Mail, WWW oder Video-Datenströme unterschieden, eventuell aber auch unterschiedliche Protokollfamilien, Quell- bzw. Zieladressen oder explizite Markierungen einzelner Datenströme.

Viele Netzbetreiber halten sich beim praktischen Einsatz solcher Differenzierungstechniken noch zurück, weil die Folgen, insbesondere die Wechselwirkungen mit Kosten- und Preismodellen schwer vorhersehbar sind [4]. Auch die Qualität der QoS-Implementierungen ist mitunter nicht auf dem Niveau der einfachen *Best-Effort*-Weiterleitung.

Eine nähere Betrachtung macht klar, dass es sich bei vielen QoS-Zusicherungen eher um eine Art „Versicherung gegen Netzengpässe“ handelt, d.h. wenn genügend Bandbreite vorhanden ist, werden sich kaum Warteschlangen bilden und eine irgendwie geartete Differenzierung kommt überhaupt nicht zum Zuge. Ein kostenbewusster Netznutzer wird also hochwertigere Klassen erst dann (vielleicht kurzzeitig) wählen, wenn das Netz Überlastungserscheinungen zeigt. Das wiederum ist aus Sicht der Netz-Dimensionierung der denkbar ungünstigste Zeitpunkt.

In diesem Beitrag wollen wir zeigen, wie das QoS-Instrumentarium für ganz andere Zielfunktionen heute schon praktisch einsetzbare Lösungen für nichttriviale Management-Aufgaben liefern kann.

Forderungen aus der Praxis gehen oft in die Richtung, dass geschäftskritische Anwendungen beispielsweise nicht durch Videokonsum, Tauschbörsen etc. behindert werden sollen. Ferner hat eine Reihe von Netzanbietern auf der Suche nach möglichst differenzierten und feinstufigen Preismodellen neben der Zugangsbandbreite auch Datenmengen in die Tarifierung einbezogen. Für experimentierfreudige Umgebungen wie den Hochschulbereich ergibt sich daraus ein Dilemma:

- Bei hoher Zugangsbandbreite mit kleiner Durchschnittsauslastung werden viele anspruchsvolle (d.h. zeitkritische) Anwendungen auch ohne spezielle QoS-Vorkehrungen im statistischen Mittel gut funktionieren.
- Eine Reihe von „gierigen“ Anwendungen kann aber fast beliebige Bandbreiten füllen, kritisch sind hier beispielsweise einige Vertreter der Klasse *Peer-to-Peer*.

Da im Hochschulbereich eine direkte Weitergabe von Kosten an Endnutzer wenig üblich bzw. praktikabel ist, wurde ein Weg gesucht, wie individuelle Nutzer trotzdem zu einem „vernünftigen“ Umgang mit Netzressourcen veranlasst werden können.

Ein erster Ansatz bestand darin, für jeden Nutzer der Hochschule (bzw. einer bestimmten Nutzergruppe) die Datenmenge über einen gewissen Zeitraum zu begrenzen. Nach einer Überschreitung dieser Grenze wurde der Netzzugang gesperrt (ggf. erst nach einer Verwarnung bei erstmaligem Überschreiten).
Die Festlegung dieses Limits ist nicht einfach. Wenn das zur Verfügung stehende Transfervolumen einfach auf die Anzahl der Nutzer aufgeteilt wird, ergibt sich sicherlich eine unnötig enge Beschränkung, da nur wenige Nutzer das Limit tatsächlich voll ausnutzen werden. Das Limit wird also nach statistischen Erfahrungen höher angesetzt. Wenn z.B. durch eine neue Nutzungsart plötzlich viele Nutzer dem Limit nahekommen, haben wir ein Problem.
Die beschriebene Verfahrensweise wurde seit Einführung der Volumentarife beim DFN-Verein im Chemnitzer Studentennetz (CSN) eingesetzt.

Als weiteres Problem erwies sich der Aufwand für Ausnahmeregelungen (z.B. für bestimmte Aufgaben beim Studium). Wochen- oder Monatslimits statt der verwendeten Tageslimits könnten dieses Problem zwar entschärfen, dies würde aber die statistischen Annahmen deutlich beeinflussen (*am Monatsende wird noch schnell verbraucht, was „übrig“ ist*).

2 Lösungsansatz

Den Ausweg aus den beschriebenen Schwierigkeiten suchten wir in einem intelligenteren Management des Netzzuganges. In unserem Fall ist ein Nutzer eindeutig über die IP-Adresse seines Rechners identifizierbar, was der Lösung des Problems entgegenkommt. Das von einem Nutzer pro Zeiteinheit (z.B. täglich) beanspruchte Datenvolumen wird gemessen. Danach wird er in Verkehrsklassen mit verschiedenen verfügbaren Bandbreiten eingeordnet. Im Extremfall kann sich der Nutzer in einer Klasse wiederfinden, in der es nicht mehr möglich ist, ein bestimmtes Verkehrsvolumen zu überschreiten.

Am Ende jedes Messzeitraums erfolgt gegebenenfalls eine Einordnung in eine andere Klasse. Bei Überschreitung einer bestimmten Verkehrsmenge wird der betreffende Nutzer am nächsten Tag in eine niedriger priorisierte und in der Bandbreite begrenzte Klasse eingeordnet.
Zum Schutz vor Extremfällen erfolgt auch eine Auswertung in kleineren Zeitabständen (stündlich), um auf Ausreißer schneller reagieren zu können. Nach einer längeren Einhaltung der vorgesehenen Verkehrsmengen werden die Nutzer schrittweise wieder in eine Klasse mit höherer verfügbarer Bandbreite eingeordnet.

Noch vorteilhafter als eine feste Anzahl von Klassen wäre eine separate Klasse für jeden Nutzer. Die dabei entstehende Klassen- und Regelmenge für einige 1000 Nutzer erwies sich allerdings als Herausforderung für die vorgesehene wenig kostenaufwendige Routertechnik (bei Durchsatzzielen > 100 MBit/s).

3 Festlegung der Parameter für die Klassen

Um die folgenden Erläuterungen etwas konkreter zu fassen, haben wir die initial verwendete Klasseneinteilung in Tabelle 1 dargestellt.

Klasse	Ab Menge	Bandbreite	Tage
1	100 MByte/Tag	100 MBit/s	2
2	200 MByte/Tag	50 MBit/s	4
3	300 MByte/Tag	30 MBit/s	6
4	400 MByte/Tag	5 MBit/s	8
5	500 MByte/Tag	500 KBit/s	10

Tabelle 1: Parameter der Klassen

Die konkreten Werte der Bandbreiten sollen durch eine automatische Regelung ermittelt werden. Wichtig für die Initialwerte ist ein faires Verhältnis von Limits und Bandbreiten zwischen den Klassen, da dieses Verhältnis auch bei der Regelung immer bestehen bleibt.

Die Anzahl der Tage in einer Klasse ist nicht etwa willkürlich festgelegt, sondern so, dass sie näherungsweise dieser Formel folgt:

$$t = \frac{T_t}{T_n} \tag{1}$$

Dabei ist T_t die verursachte Menge pro Tag (von engl. *Traffic*), T_n die Menge, die jedem Nutzer pro Tag theoretisch zustehen würde (in unserem Beispiel 30 MByte). So wird der insgesamt verursachte Verkehr entsprechend nach unten korrigiert. Am Ende der Zeitspanne wird nur Verkehr in der Größenordnung des vorgegebenen Tagesmittels (die 30 MByte/Tag) verursacht worden sein.

Wenn der Nutzer durch eine erste Überschreitung eines Limits beispielsweise in Klasse 3 eingeordnet wird, würde er im Normalfall sechs Tage in dieser Klasse verweilen, danach noch vier Tage in Klasse 2 und zwei Tage in Klasse 1 (Abbildung 1).

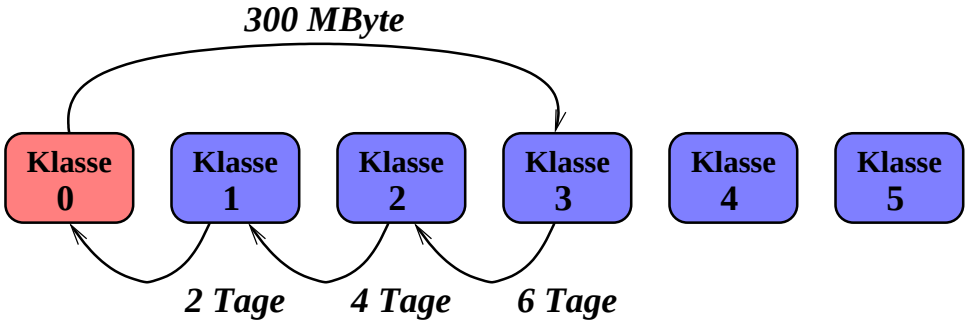

Abbildung 1: Klassenübergänge, wenn Klasse = 0

Dies gilt allerdings nur, solange er in dieser Zeit keine weiteren Limitüberschreitungen verursacht. Ist ein Nutzer bereits in eine Klasse > 0 eingeordnet, wird er bei einer erneuten Überschreitung nicht in die seinem Verkehr entsprechende Klasse aus der Tabelle zugeordnet, sondern in die nächsthöhere (Abbildung 2).

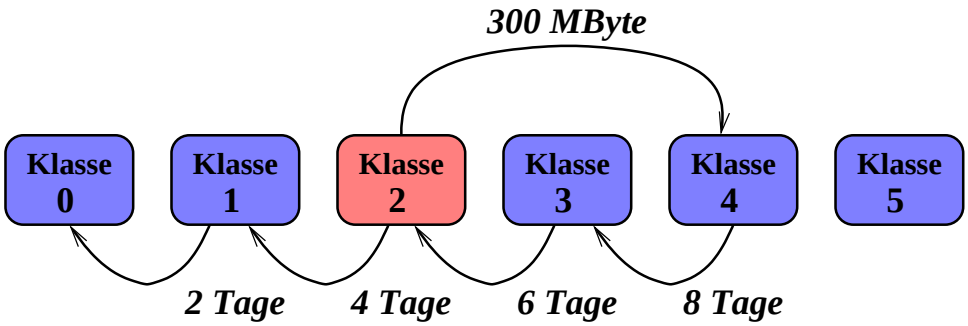

Abbildung 2: Klassenübergänge, wenn Klasse > 0

Insgesamt handelt es sich um ein relativ faires Verfahren, da z.B. ein Nutzer, der zweimal die 100-MByte-Grenze überschreitet, genauso in Klasse 2 eingeordnet wird wie ein Nutzer, der einmalig die 200-MByte-Grenze übertritt.

4 Automatische Regelung

Die automatische Regelung der Bandbreiten dient zur optimalen Ausnutzung des vorgegebenen Monatslimits.
Eine Zuordnung zu Begriffen der Regelungstechnik lässt sich entsprechend Tabelle 2 treffen.

Regelstrecke	Netzwerk
Regelgröße	Gesamtverkehr pro Monat
Führungsgröße	Klassenparameter, Bewertung
Regelabweichung	hochgerechneter Gesamtverkehr
Störgröße	Nutzerverhalten
Stellgröße	Bandbreiten der Klassen
Regler	Evaluator
Messglied	Meter
Stellglied	Traffic Control

Tabelle 2: Zuordnung zu Regelungsbegriffen [5, 6]

Zur Regelung der Bandbreiten wird der bisher in diesem Monat verursachte Verkehr (T) auf den gesamten Monat hochgerechnet (T_m). Das erfolgt über den Quotienten aus bereits vergangenen Tagen (D) und Gesamtanzahl der Tage des Monats (D_m):

$$T_m = T \cdot \frac{D_m}{D} \tag{2}$$

Übersteigt dieser Wert das Monatslimit, werden die Bandbreiten (B) mit dem Quotienten aus Monatslimit (L_m) und Verkehrsmenge pro Monat (T_m) multipliziert, was eine Anpassung nach unten ergibt:

$$B_{neu} = B_{alt} \cdot \frac{L_m}{T_m} \tag{3}$$

Ist das Monatslimit dagegen noch nicht erreicht (es stehen also noch Kapazitäten zur Verfügung), werden die Bandbreiten nach oben korrigiert. Das geschieht diesmal über die Quadratwurzel des o.g. Quotienten, damit die Anpassung nach oben etwas verhaltener abläuft und Oszillationen der Bandbreiten vermieden werden:

$$B_{neu} = B_{alt} \cdot \sqrt{\frac{L_m}{T_m}} \tag{4}$$

Die Regelung erfolgt erst nach den ersten fünf Tagen eines Monats, da die Werte zu Anfang noch zu starken Schwankungen unterworfen sind, als dass eine zuverlässige Hochrechnung möglich ist. Weiterhin werden Korrekturfaktoren nahe 1 nicht berücksichtigt, wieder zur Vermeidung unnötiger Bandbreitenänderungen.

5 Architektur des Systems

Die erläuterten Prinzipien und Verfahren werden durch ein System realisiert, das zwischen dem externen Netzzugang und dem internen Netz mit den Nutzern angeordnet ist (Abbildung 3).

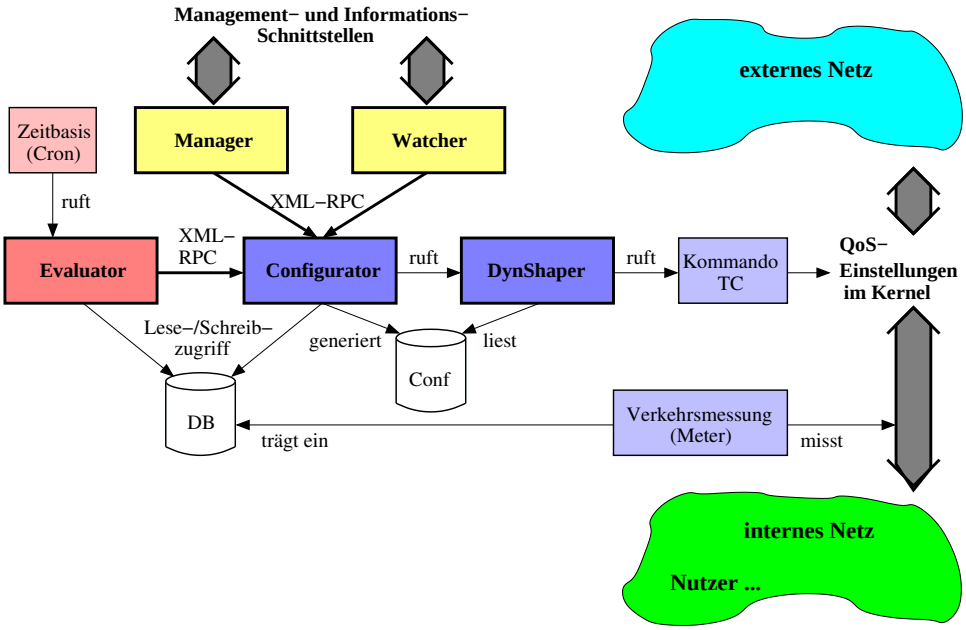

Abbildung 3: Architektur des Systems

Manager

Der *Manager* dient zur Verwaltung des *DynShapers*. Er legt die Parameter für die verwendeten Interfaces, Pfadnamen zu Programmen und die Monatsobergrenze fest, stellt die Werte für die einzelnen Klassen ein und definiert Ausnahmeregelungen (z.B. Sonderbehandlung von bestimmten Protokollen). Die Einstellungen müssen nur am Anfang vorgenommen werden, im laufenden Betrieb regelt sich das System soweit wie möglich selbst.

Watcher

Mit dem *Watcher* ist es den einzelnen Nutzern möglich, ihre aktuellen Werte wie den verursachten Traffic, die Verkehrsklasse, der sie momentan zugeordnet sind sowie deren Parameter abzufragen.

Configurator

Durch den *Configurator* erfolgt die Konfiguration des *DynShapers*. Er verwaltet die Konfigurationsdaten und beantwortet XML-RPC-Anfragen, die Konfigurationsoptionen lesen oder

schreiben wollen. Deswegen muss er ebenso wie der *Manager* und der *Watcher* vom Webserver erreichbar sein. Genutzt wird der *Configurator* von den beiden bereits genannten Programmen und vom *Evaluator*.

In der Datenbank werden folgende Einstellungen gespeichert:

- internes und externes Interface mit ihren Bandbreiten
- das Monatslimit und ob eine automatische Regelung stattfinden soll
- ab wann eine E-Mail bei Einordnung in eine schlechtere Klasse geschickt werden soll
- Pfade zum *DynShaper*, *TC*, *ModProbe* sowie zur Konfigurationsdatei und zum Logfile
- die einzelnen Klassen mit Bandbreite (Rate), Limit, Tagen und IP-Adressen sowie der Festlegung, ob ankommender und/oder abgehender Verkehr beschränkt werden und ob diese Beschränkung sofort erfolgen soll (bei Extremfällen)
- die Ausnahmen mit Bandbreite (Rate), Match-Anweisungen für ankommenden und abgehenden Verkehr sowie Festlegung über harte Begrenzung des Verkehrs (bounded)

Evaluator

Der *Evaluator* ist dafür zuständig, das Verkehrsaufkommen der Nutzer auszuwerten und die Nutzer bei Bedarf einer anderen Verkehrsklasse zuzuordnen. Dazu wird einmal pro Stunde eine Abfrage der aktuellen Werte durchgeführt. Die Klasse und das Datum der Einordnung werden in der Datenbank gespeichert. Weiterhin dient der *Evaluator* zur automatischen Regelung der Bandbreiten für die Klassen, damit das festgelegte Monatslimit nicht überschritten werden kann.

DynShaper

Der *DynShaper* dient zur eigentlichen Begrenzung des Verkehrs. Er wertet die vom *Configurator* erzeugte Konfiguration aus und legt Queuing Disciplines, Klassen und Filterregeln fest. Das Skript kann vom *Configurator* und über diesen auch vom *Evaluator* neugestartet werden, wenn sich Einstellungen geändert haben. Weiterhin sollte es als Startup-Skript vorliegen, um nach einem Neustart die Beschränkungen sofort wieder verfügbar zu haben.

Der *Manager* und der *Watcher* können auf unterschiedlichen Rechnern laufen. Falls sie auf demselben Rechner im gleichen Verzeichnis installiert werden, muss sichergestellt werden, dass auf den Manager nur die Administratoren Zugriff besitzen. Auch der *Configurator* und der *Evaluator* können auf verschiedenen Rechnern installiert werden. Sollen alle vier Komponenten im selben Verzeichnis liegen, muss auf die Einstellung der Zugriffsbeschränkungen besondere Sorgfalt verwandt werden.

6 Realisierung der Verkehrsbeeinflussung

Als technische Plattform wird ein Router auf *Linux*-Basis verwendet. Dort stehen eine Reihe von QoS-Verfahren zur Verfügung: einfache Queuing-Algorithmen, die unterschiedliche Vergabe von Bandbreiten und Prioritäten, Algorithmen zur Überlastvermeidung von TCP-Flows und zur Verteilung von Verkehr auf mehrere Netzinterfaces [7, 8, 9].

Bei der einfachsten Strategie FIFO werden zu versendende Pakete in der Reihenfolge ihres Eingangs in eine Warteschlange eingeordnet, über die sie dann in derselben Reihenfolge gesendet werden.
Beim *Priority Queuing* werden höherpriorisierte Pakete vor niedriger priorisierten in die Warteschlange eingeordnet. Dazu muss jedes Paket analysiert und die Warteschlange gegebenenfalls umgeordnet werden, was unvorteilhaft für den Durchsatz ist.
Das *Class-Based Queuing (CBQ)* ermöglicht demgegenüber eine geringere Latenz. Hier existieren mehrere Warteschlangen, auf die die Bandbreite hierarchisch verteilt wird.

Die Klassen bilden eine Baumstruktur, in der sich die *Kinder* von den *Eltern* bei Bedarf Bandbreite borgen können, wenn dort noch Kapazität vorhanden ist. Im Gegenzug können die *Kinder* nicht benötigte Bandbreite den *Eltern* zur Verfügung stellen, die dann gegebenenfalls an andere *Kinder* verborgt wird. Den Blättern im Baum können unterschiedliche *Queuing Disciplines* zugewiesen werden, so dass verschiedene Service-Klassen durch geeignete Algorithmen behandelt werden können (Abbildung 4).

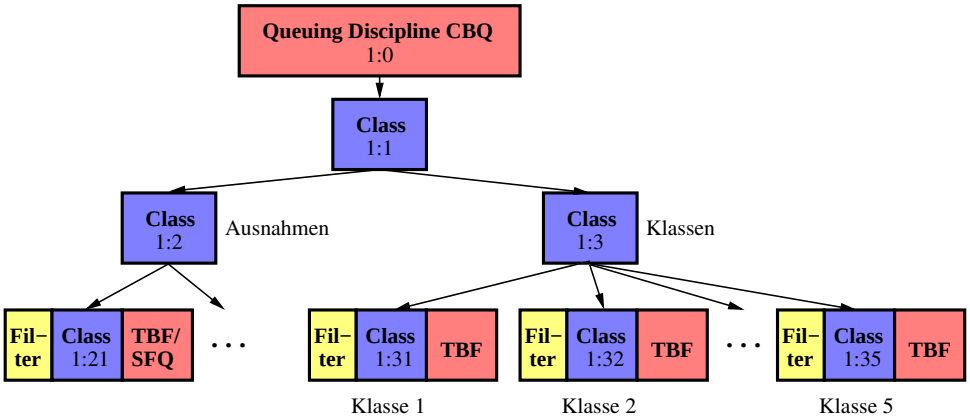

Abbildung 4: Klassenbaum im Chemnitzer Studentennetz

Die Klassifikation erfolgt über Filter, die IP-Adressen oder andere Merkmale im Paket auswerten. Nach der Klassifizierung folgt die eigentliche Verkehrsbeeinflussung (Abbildung 5). Bei uns wird für jeden Nutzer ein Filter angelegt, der ihn seiner entsprechenden Klasse zuweist, so dass pro Klasse mehrere Filter existieren.

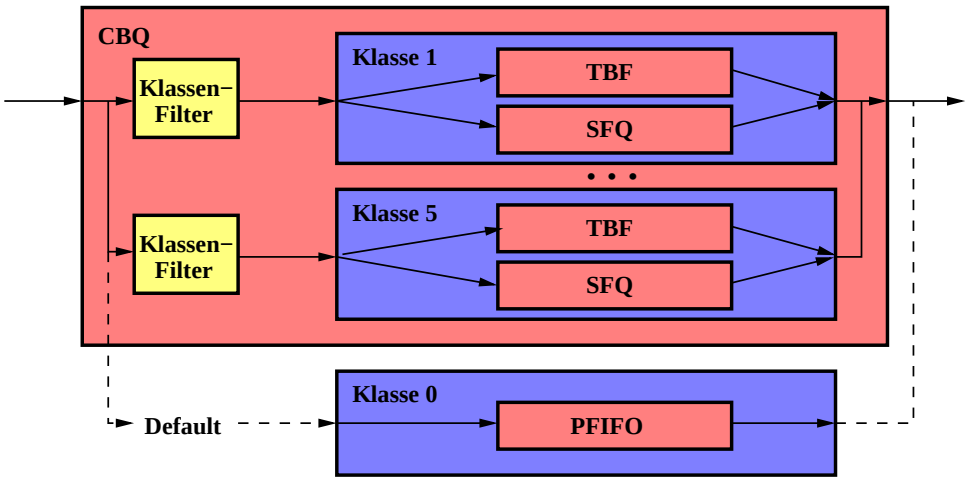

Abbildung 5: Klassifikation durch Filter

Der Token Bucket Filter (TBF) hat die Aufgabe, die Menge und Rate des Verkehrs zu kontrollieren, der in das Netz eintritt. Der Bucket wird analog zu einem regelmäßig mit Münzen „gefütterten“ Geldbeutel vom System in bestimmten Abständen mit Tokens versorgt, von denen jedes eine gewisse Menge von sendbaren Datenbytes repräsentiert. Daten können nur gesendet („gekauft“) werden, wenn für diese Menge genügend Tokens im Bucket („genügend Münzen im Geldbeutel“) sind. Dadurch sind auch Bursts möglich: wenn längere Zeit nichts gesendet wurde und sich mehrere Tokens angesammelt haben - analog zum Sparen von Geld für eine größere Ausgabe. Zusätzlich ist eine maximale Burst-Größe festlegbar.

Mittels Stochastic Fair Queuing (SFQ) werden die Datenströme durch eine Hashbildung über Quell- und Zieladresse auf mehrere Queues verteilt, die gleichmäßig per Round Robin entleert werden. Dadurch kann eine relativ gleichmäßige und faire Vergabe der vorhandenen Bandbreite auf alle Nutzer garantiert werden. Die Hash-Funktion wird in bestimmten Zeitintervallen geändert, um Hash-Kollisionen (verschiedene Verbindungen nutzen dieselbe Warteschlange) zu minimieren.

7 Implementierung

Die Management- und Informationsschnittstellen wurden mit Hilfe des *PHP Hypertext Preprocessor (PHP)* [10] entwickelt. PHP ist gegenüber der herkömmlichen CGI-Programmierung leichter zu handhaben. Es gibt eine einfach einzubindende XML-RPC-Unterstützung [11].

Auch die Serverseite zum XML-RPC ist mit Hilfe von PHP realisiert. Die XML-RPC-Daten werden per `POST` übergeben, es ist also kein separater Server nötig.

Für den *Evaluator* wird zusätzlich ein unabhängiger PHP-Interpreter benutzt, der mit XML- und Datenbankunterstützung kompiliert sein muss.

Der eigentliche *DynShaper* ist ein einfaches (Bash-)Shellskript, da hier nur wenige Konfigurationsoptionen eingelesen werden müssen, die zum Einrichten der Klassen und Filter notwendig sind, was ebenfalls durch dieses Skript erfolgt.

Zum Austausch der Daten zwischen den einzelnen Programmen des Systems dient XML-RPC [12], weil darüber auch komplexere Daten relativ komfortabel übergeben werden können. XML-RPC ist ein Remote-Procedure-Call-Protokoll, welches auf der Basis von XML und HTTP realisiert ist. HTTP dient dabei als Übertragungsprotokoll, welches die XML-kodierten Nutzdaten transportiert. Es ist ein sehr einfaches Protokoll, d.h. es ist einfach zu verstehen und einfach zu implementieren. Der große Vorteil ist seine Sprachunabhängigkeit, so dass Daten zwischen Applikationen in verschiedenen Programmiersprachen ausgetauscht werden können.

Als Datenbanksystem wird im CSN schon seit einiger Zeit PostgreSQL eingesetzt und vom DynShaper-System mitverwendet.

8 Messungen

Bei einer Messreihe auf einem PIII-450 hat sich gezeigt, dass nur eine begrenzte Zahl von Filterregeln vom System handhabbar ist (Abbildung 6). Die Anzahl der Klassen hat dagegen kaum Auswirkungen: Bei 5 Klassen ist der Durchsatz bei der gleichen Filtermenge genauso hoch wie bei 5000 Klassen. Der verwendete Router sollte also so dimensioniert werden, dass er bei der erwarteten Filtermenge noch den benötigten Durchsatz liefern kann.

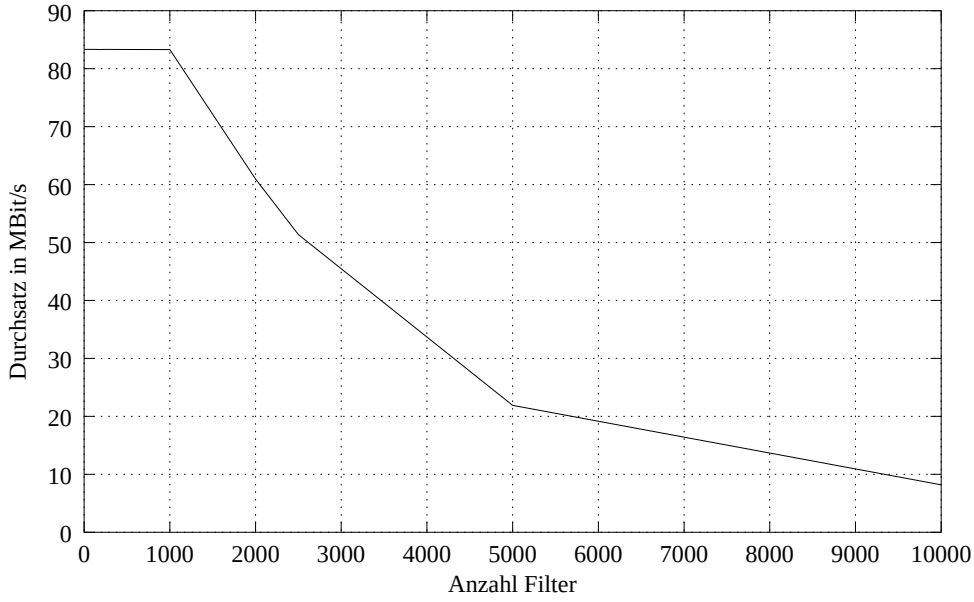

Abbildung 6: Durchsatzrate bei vielen Filtern

Der Test des neuen Systems (der auf einem separaten Rechner erfolgte) verwendete alte Verkehrsdaten, die aus der Datenbank gewonnen wurden. Es wurden keinerlei Begrenzungsregeln aktiv, stattdessen sollte nur das Verhalten des Systems bei in Art und Anzahl realen Daten überprüft werden [13].

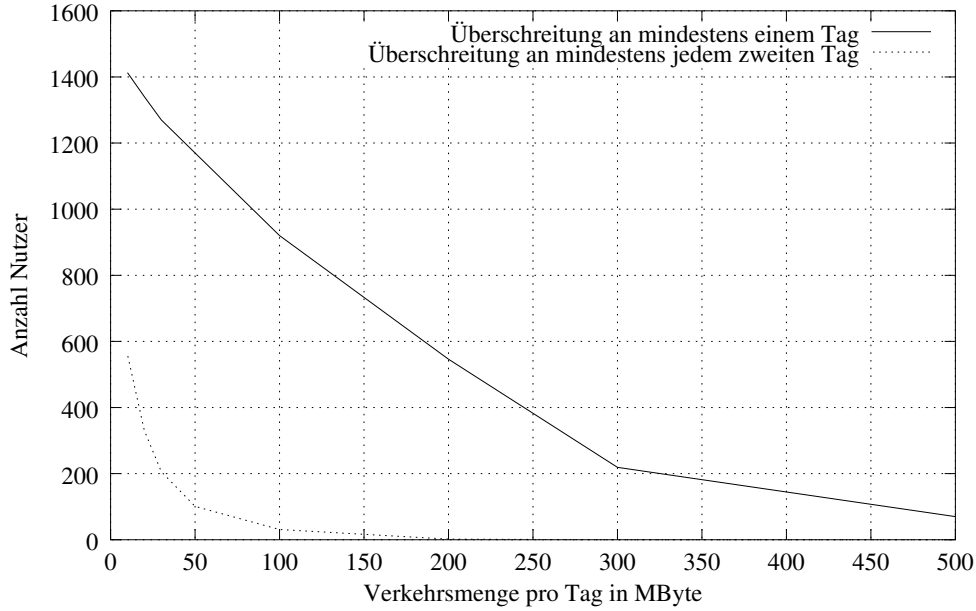

Abbildung 7: Verkehrsmenge pro Tag in MByte

Nur ein relativ kleiner Prozentsatz der Nutzer ist für den überwiegenden Teil des Verkehrsaufkommens verantwortlich (Abbildung 7). Es sind auch nur diese Nutzer von der Bandbreitenbegrenzung betroffen, die große Masse bleibt davon unberücksichtigt! So lassen sich schon durch die Einschränkung weniger Nutzer die Zielparameter relativ stark beeinflussen.

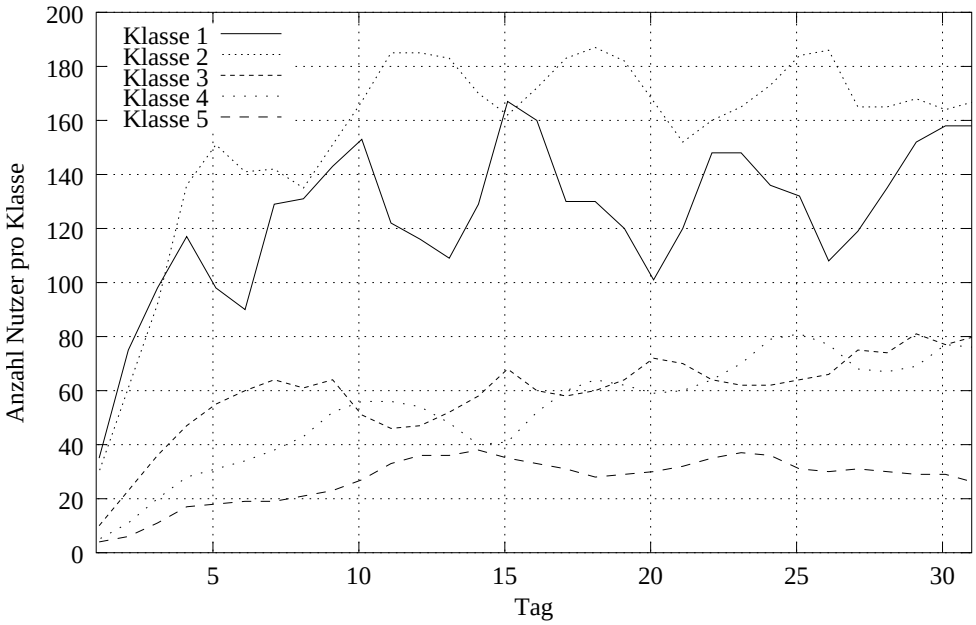

Abbildung 8: Nutzer, für die die Bandbreite begrenzt wird

Die starke Schwankung der Nutzeranzahl in den Klassen 1 und 2 (Abbildung 8) ist auf das geringere Verkehrsaufkommen am Wochenende zurückzuführen. Aufgrund der kurzen Verweildauer

in diesen Klassen wird diese Schwankung nicht geglättet, wie es vor allem in Klasse 4 und 5 geschieht.

Im realen Einsatz mit aktiver Begrenzung des Datenverkehrs ergaben sich ähnliche Werteverläufe. Der benutzte Router zeigte auch keine Durchsatzprobleme.

9 Einsatzszenarium

Das Chemnitzer Studentennetz (CSN) ist eine Initiative von Studenten für Studenten mit dem Ziel, die privaten Rechner in den Studentenwohnheimen an das Campusnetz der TU Chemnitz anzuschließen (Abbildung 9).

Abbildung 9: Wohnheime im CSN (Foto: Karsten Petersen)

Derzeit gibt es etwa 1700 Nutzer, alle Wohnheime sind angeschlossen. Damit hat die TU Chemnitz eines der größten Studentennetze, die Anfänge reichen zurück bis ins Jahr 1994 [14]. Eine (vereinfachte) Netztopologie zeigt Abbildung 10.

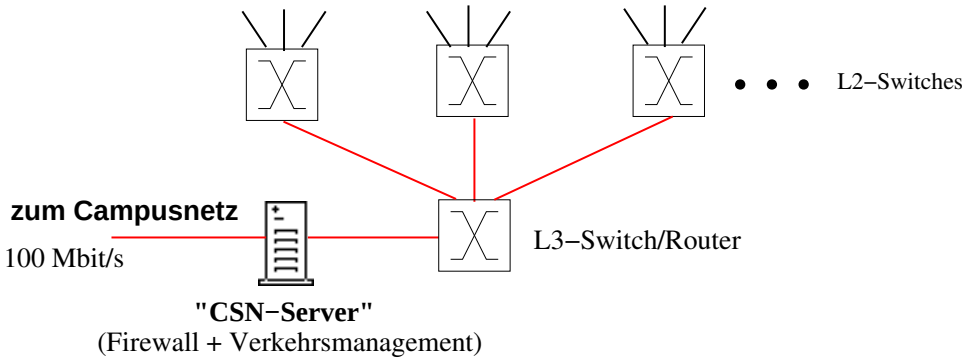

Abbildung 10: Netztopologie des Chemnitzer Studentennetzes

Die Bandbreitenbeschränkungen werden auf dem CSN-Server eingesetzt, über den der Anschluss an das Campusnetz hergestellt wird. Nachdem der *DynShaper* seit August 2001 im Testbetrieb lief (ab September 2001 zeitweise mit aktiven Begrenzungsregeln), wurde er ab Oktober 2001 offiziell in Betrieb genommen [15].

10 Ausblick

Im Laufe des Testbetriebs wurden einige Aspekte deutlich, die noch verbesserungswürdig sind.

Zum einen soll eine gleitende Zählung des Verkehrs über die letzten 14 Tage erfolgen, so dass Spitzen abgeschwächt und einmalige Überschreitungen nicht so hart sanktioniert werden. Die Einordnung der Nutzer in die Klassen erfolgt stündlich neu.

Da die Bandbreite einer Klasse fair auf alle offenen Verbindungen aufgeteilt wird, ist es durch eine genügend hohe Zahl an Verbindungen möglich, einen Großteil der Bandbreite dieser Klasse für sich zu beanspruchen. Vor allem in den stärker beschränkten Klassen tritt dieser Effekt in Erscheinung. Hier könnte nach Wegen gesucht werden, um die Bandbreite auf Nutzer (oder IP-Adressen) und nicht auf die Verbindungen aufzuteilen.

Als alternative Lösungsmöglichkeit bietet sich der Einsatz von HTB (*Hierarchical Token Bucket*) [16] an. Diese *Queuing Discipline* wird eventuell das CBQ ablösen, da dieser Algorithmus einfacher und genauer als CBQ ist.

Noch nicht vollkommen zufriedenstellen gelöst ist die Behandlung von Verkehr mit unterschiedlichem „Wert“. So wird derzeit der Verkehr zu einigen Rechnern innerhalb der Universität von der Begrenzung ausgenommen. Damit werden Anreize geschaffen, interne Ressourcen wie z.B. den FTP-Server des Rechenzentrums stärker zu nutzen. Der Verkehr zum lokalen WWW-Proxy/Cache-Server wird in die Begrenzung einbezogen (der Caching-Gewinn ist bei der Festlegung der Klassengrenzen berücksichtigt).
Ein Problem ist die Nutzung von Rechnern innerhalb der Universität als „Transferstelle“. Hierfür muss vorerst eine Kombination aus administrativen Regelungen und Verkehrsmessungen zur Feststellung von Verstößen dafür sorgen, dass an dieser Stelle keine Umgehung des Bandbreiten-Managements stattfindet.

Ansonsten sind die Erfahrungen überwiegend positiv. Das System regelt sich weitestgehend selbst, so dass bestimmte Zielparameter eingehalten werden, wie z.B. eine bestimmte Monatsobergrenze der Verkehrsmenge und garantierte Mengen pro Nutzer und Monat. Der Vorteil dieser automatischen Regelung ist die optimale Ausnutzung von vorgegebenen Limits. So sind z.B. in Ferienzeiten, in denen weniger Personen das Netz nutzen, höhere Bandbreiten für die einzelnen Verkehrsklassen vorhanden.

Ein vielleicht unterschätzter Aspekt war die Gewinnung einer positiven Akzeptanz durch die Nutzer. Anfängliche Darstellungen und Begriffswahl ließen Assoziationen mit *Führerschein-Strafpunkten* aufkommen. Mittlerweile sind Begriffswelt und Resonanz aber sachlicher geworden, viele Nutzer verstehen das System als einen Mechanismus zur fairen Verteilung knapper Ressourcen, der sachkundiges und vernünftiges Handeln belohnt.

Literatur

[1] Geoff Huston: „Internet Performance Survival Guide - QoS Strategies for Multiservice Networks“, Wiley Computer Publishing, 2000

[2] Geoff Huston: RFC 2990 „Next Steps for the IP QoS Architecture“, November 2000
http://www.ietf.org/rfc/rfc2990.txt

[3] Paul Ferguson, Geoff Huston: „Quality of Service - Delivering QoS on the Internet and in Corporate Networks“, Wiley Computer Publishing, 1998

[4] T. Li, Y. Rekhter: RFC 2430„A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)“, Oktober 1998
http://www.ietf.org/rfc/rfc2430.txt

[5] Heinz Töpfer, Peter Besch: „Grundlagen der Automatisierungstechnik“, 2., durchgesehene Auflage, VEB Verlag Technik Berlin, 1989

[6] Torsten Naumann: „Diplomarbeit - Ressourcensteuerung für Internet-Zugänge“, TU Chemnitz, Professur Rechnernetze und verteilte Systeme, 1997

[7] Jan Horbach: „Studienarbeit - QoS and/or fair Queuing“, TU Chemnitz, Professur Rechnernetze und verteilte Systeme, 2000

[8] Saravanan Radhakrishan: „Linux - Advanced Networking Overview“, 1999
http://qos.ittc.ukans.edu/howto/

[9] Mark Lamb: „iproute2+tc notes“, 1999
http://snafu.freedom.org/linux2.2/iproute-notes.html

[10] PHP Group: „PHP Hypertext Preprocessor“, 2001
http://www.php.net

[11] „XML-RPC Resources“, Useful Information Company, 1999-2001
http://xmlrpc.usefulinc.com

[12] UserLand: „XML-RPC Home Page“, UserLand Software, Inc., 2001
http://www.xmlrpc.com

[13] Jan Horbach: „Diplomarbeit - Dynamische Bandbreitenbeschränkung mit QoS“, TU Chemnitz, Professur Rechnernetze und verteilte Systeme, 2001
http://archiv.tu-chemnitz.de/pub/2001/0100/

[14] „[CSN] Informationen - Allgemein“, Chemnitzer Studentennetz, 2000
https://www.csn.tu-chemnitz.de/info/allg_info.html

[15] „[CSN] Informationen - Beschränkungen“, Chemnitzer Studentennetz, 2001
https://www.csn.tu-chemnitz.de/info/limits.html

[16] Martin Devera, Don Cohen: „HTB Linux queuing discipline manual“, 2002
http://luxik.cdi.cz/~devik/qos/