CHEMNITZ UNIVERSITY OF TECHNOLOGY

Faculty of Computer Science

Chair for Computer Engineering

Diploma Thesis

Using Case-based Reasoning to Control Traffic Consumption

Markus Schade

Chemnitz, January 30, 2007

Supervisor: Prof. Dr. W. Hardt
Dr. Markus Borschbach

Acknowledgement

The writer would like to thank the team of the Chemnitzer StudentenNetz for their confidence and the opportunity test his theories in an actual network. I also wish to express my gratitude to all those, who have supported me in my research. Thanks for everything.

Abstract

Quality of Service is commonly used to shape network traffic to meet specified criteria. The various scenarios include limiting and reserving bandwidth for a particular application, host or user, prioritizing latency sensitive traffic or equal distribution of unreserved bandwidth. There are, however, no commercial or open source applications (apart from the DynShaper software) known to the author, which are able to distribute and control a traffic quota by more sophisticated means than fixed per user limits and simple disconnecting after the user reaches the limit. The lack of such applications can largely be attributed to the fact that most commercial internet connections are sold on a per bandwidth basis without traffic limitations. In case such limitations do exist and continuous network operation is required, the only way to meet both conditions is to limit the available bandwidth. Restricting the bandwidth to a fixed rate is a rather awkward albeit effective solution. The DynShaper software takes a more sophisticated approach to this problem. It distributes the quota on a daily basis, where each day receives the same share. The users are sorted into predefined groups with different bandwidths depending on their recent consumption. This classification is periodically updated to ensure the sorting order is maintained. The bandwidths of these groups is dynamically adjusted depending on the actual consumption to provide an efficient utilization.

The basic concept of adaptive bandwidth control based on share utilization is common to all distribution models used by DynShaper. This thesis proposes another distribution model using a case-based reasoning approach, a method for machine learning which is classified as conventional artificial intelligence. Case-based reasoning tries to solve new problems based on the solutions of similar problems from the past. Controlling the network traffic to remain within a fixed quota can be modeled as such a problem if the traffic patterns are recurring. Possible solutions can be derived from statistical data and altered to suit the new problems. When an untested solution is applied, the software supervises the execution and revises the solution accordingly, if the actual results deviate from the precalculated schedule.

Tests of the model in the Chemnitzer StudentenNetz showed that case-based reasoning is a feasible concept for controlling network traffic, especially in non-standard scenarios. Its main disadvantage lies in the need for statistical data, the need for existing successful solutions. This disadvantage can be mitigated by using the average-based distribution model until sufficient statistical data has been gathered. Using the case-based model, the quota utilization could be

raised to over 93 % in a non-standard situation (December with Christmas holidays) and to 99 % in a standard scenario, while the previous average-based control yielded only 85 % and 95 % respectively.

Contents

1 Introduction

Virtually everything is limited in some way, whether it is time, money or fossil fuels. Even the most abundant resources can be depleted given enough time. Some limitations are due to physical constraints, while others exist for purely political reasons. Whatever they are, man has tried to control what can be controlled and to exploit which cannot.

The Internet has grown exponentially since the late nineties as has the urge for bandwidth. The result is a constant need for more and better hardware to cope with the demand. For an Internet Service Provider (ISP), new hardware involves long-term investments, which have to be refinanced by its customers. However, not all of them are willing or able to pay premium prices for a first class internet connection. In order for the lines to be profitable, limited versions are offered. Possible limitations include the amount of traffic, which can be transferred without additional charge, or the artificial reduction of bandwidth.

To control a network traffic quota, the ISP has to implement some form of accounting, which is required for billing on volume. The customers of such lines face a similar problem, namely how to make the most use of the quota. As simple as it may seem for a single user to control his own internet usage, the more difficult it becomes, when redistributing the quota to a number of people, where each one of them should receive a fair share. In this case network control software can help to achieve fair distribution and reduce the risk of uncontrolled traffic consumption.

This thesis focuses on optimizing the utilization of a traffic quota, which has to be distributed to a large number of people. Its aim is to provide an alternative choice to the average-based algorithm currently used in the DynShaper [1] software. As the quota must not be exceeded, the algorithm has to take reasonable preventive measures to safeguard against such a situation. The next sections provide an outline for this thesis and its motivation. They will be followed by a detailed description of the thesis assignment and examples of related work.

1.1 Motivation

Traffic quotas can be hard or soft. After reaching a hard quota, further access is either denied or restricted to a low bandwidth. Soft quotas permit the user to exceed the agreed amount, either temporarily or at additional charge. Neither of those two alternatives is desirable, therefore a software is needed, which allows to efficiently utilize and control such a quota.

Network Traffic Control and Resource Management Problems

The primary aim of network traffic control is to reduce congestion, packet loss and latency. Another aspect is to reduce excessive usage to ensure sufficient bandwidth is available for all users in a shared network. The Chemnitzer StudentenNetz (CSN) uses a comparative approach, which assigns different bandwidths to each user based on the consumed traffic. The reason behind this is to provide a fair distribution without disabling network access of individual users. A global traffic quota for all download traffic imposes an upper limit upon the distribution scheme. DynShaper dynamically adapts the available bandwidth to meet this demand. The algorithm used to calculate the scaling factor is subject of discussion in this thesis.

1.2 Task of Thesis

A monthly traffic quota, granted by the Computing Center of the Chemnitz University of Technology (URZ), can be freely distributed by the CSN among the students living in the dormitories, but it must not be exceeded. In order to maximize its utilization the CSN uses a software (DynShaper [1]), which leverages Quality of Service (QoS) measures, provided by the Linux kernel, to dynamically adapt the available bandwidth, thus controlling traffic consumption.

Motivation for this diploma thesis is to further improve the utilization of this traffic quota using case-based reasoning as a new way to project daily consumption and control the available bandwidth. The goals are:

- Identify and analyze the deficiencies of the current feedback algorithm.
- Determine if these can be eliminated or at least mitigated.
- Evaluate if a case-based reasoning approach can be applied to a traffic control scenario.
- The algorithm should use existing statistical data to provide a strategy for each month, which does not exceed the quota.
- Suggest possible countermeasures and revision strategies to the projection if the results should deviate. The remedies should make a reasonable effort to limit the probability of exceeding the quota.

- The algorithm is to be implemented as an add-on to the DynShaper software originally developed by Jan Horbach and, if possible, tested in a real world scenario to provide empirical data.

1.3 Document Layout

The Dynshaper software relies on several key features and auxiliary tools to fulfill its purpose. Chapter 2 provides an introductory background, dealing with the flow control issues of TCP/IP, the concepts of QoS, as well as network monitoring and accounting. In addition to the network related topics, the basic principle of case-based reasoning is explained.

As the motivation for this thesis is to improve an existing solution, the current average-based algorithm is analyzed in Chapter 3. In the next two chapters, the proposed algorithm and its design aspects are presented and discussed. The test environment, the test cases and an analysis of the results will follow in chapter 6. The final chapter (7) concludes the thesis, making suggestions for further developments arising from this work. It also evaluates the extend to which the aims of this study have been achieved.

1.4 Related Work

The interest in network traffic control is shifting towards bandwidth management as more and more internet connections come with traffic flat rates. In the consumer market, the price for a Digital Subscriber Line (DSL) flat rate has dropped below 10 Euros a month. While traffic has become cheap, available bandwidth can often not satisfy the demand. Therefore various solutions exist, which use different approaches to solve the problem of insufficient bandwidth.

1.4.1 Traffic Quotas

In large shared networks without corporate-like regulation of appropriate uses of internet, traffic quotas are employed to ensure that excessive usage by single users is restricted. In general this constitutes in an allowed traffic volume per user per day. If this quota is exceeded the user is either completely disconnected or the bandwidth is drastically reduced. Bandwidth reduction can either be instantaneous or gradually.

For example, the Oregon State University (USA) allows one gigabyte per user per day or an average of 12 kB/s [2]. If a user exceeds this limit, his bandwidth is gradually reduced. A different scheme is used at the residence hall network [3] at the University of Stuttgart (Germany). Here the users receive 4 points per hour. Points or fractions of them are deducted from the users budget depending on the source of the traffic. Below zero points, the user is disconnected until the budget has increased to positive values.

The downside of fixed traffic quotas is, that they are too inflexible to fit the actual purpose: reducing congestion, avoiding uplink saturation, or even controlling a monthly quota. This is largely because a quota is established based on the average case user. However, in times of reduced network activity, such as weekends or holidays, excessive bandwidth is left unused. In the other case the quota cannot guarantee a fair share of the bandwidth for each user.

1.4.2 Traffic Prioritization

Traffic Prioritization, being one of the classic applications of QoS, is found among home users and corporate entities alike. Its fundamental aim is to reduce latency and increase throughput for the desired types of traffic at the expense of less important ones.

Because quality of service measures can be applied even on lines with low bandwidth, there are a large number of applications promising to deliver fast downloads, low latency and guaranteed bandwidth for business critical applications. Commercial products such as Packeteers PacketShaper [4] analyze network traffic at the application level to discriminate between important and unsanctioned traffic flows.

2 Fundamentals

This chapter provides information about the underlying mechanisms and algorithms of the various subsystems on which the proposed algorithm depends. It begins with the ingredients required to establish a control, followed by a mapping on the network traffic scenario. In the next sections, the main building blocks are further explained.

2.1 Internet Traffic Properties

The underlying protocol of the Web is the Internet Protocol (IP) or, more precisely, the IP suite, because it consists of a number of layered protocols. Each protocol is concerned with a different aspect. Therefore, traffic control must at target those, which regulate the flow of data. The most common protocol used in the transport layer is Transmission Control Protocol (TCP) [5]. It accounts for 60 % to 90 % of all IP traffic in the Internet today. When examining a sample from the CSN, the author discovered that the local traffic is comprised of over 95 % TCP (table 2.1).

Protocol	Percentage
TCP	95.53 %
UDP	4.43 %
others	< 1 %

Table 2.1: Protocol breakdown of IP traffic (Source: 1 TB traffic sample from CSN)

2.1.1 TCP Flow Control

As the most common transport protocol TCP has been analyzed by many researchers and constantly improved to meet the demands of today's Internet. It was first standardized in RFC793 [6] by Jon Postel. Besides providing reliable in-sequence delivery and connection oriented stream transmission, TCP features a flow control mechanism and a congestion avoidance algorithm. The following sections present these features to show that TCP is a bandwidth-friendly protocol, which is sensitive to artificial traffic engineering, as it throttles when packets are delayed or lost.

When TCP was first designed, it was obvious that a stream oriented protocol needed a valve to control the stream. The window field in the TCP header is used to advertise the amount of data each side is currently able and/or willing to accept. It is a 16 bit field which allows a maximum window size of 64 kB. It was later increased by the TCP window scale option, defined in RFC1323 [7], to a maximum of one Gigabyte. The option is only used during the three-way handshake. A window size of zero indicates that no data can currently be accepted, thus stopping the data stream. As long as TCP segments are exchanged, the connection is kept in an established state, theoretically indefinitely, even with a zero window. In this hold state, the sending side continues to send segments with one byte of data in order to immediately seize the opportunity to deliver data in case of a window update.

2.1.2 TCP Segmentation

While the window field controls the flow within the connection, the congestion avoidance algorithm is concerned with the timely delivery of the TCP segments by the network. The lower layers (1-3) of the OSI model have their own restrictions, which in turn impact TCP in the transport layer (4). Theoretically, a segment could carry as much data as the window permits, but TCP operates on top of IP. Its segments are encapsulated in IP datagrams, which are restricted to 64 kB (IPv4). These are transported by a data link layer technology such as Ethernet or Asynchronous Transfer Mode (ATM), which have a fixed Maximum Transfer Unit (MTU), the largest possible frame size. Therefore, larger IP datagrams have to be fragmented and must be reassembled at the receiving side. Reassembling takes up processing power and degrades performance, because a lost fragment causes the retransmission of the entire IP datagram. In addition, RFC791 [8] states, that all hosts MUST only accept and reassemble datagrams of up to 576 octets, a byte consisting of 8 bits. Larger datagrams do not have to be accepted or reassembled and hence should only be sent, if it is assured that the recipient is accepting them. This is the main reason, as stated in RFC879 [9], to put the default Maximum Segment Size (MSS) at 536 bytes, which is 576 bytes minus 20 bytes each for the IP and TCP header. Nevertheless both sides can agree on a larger MSS by adding the option during the three-way handshake. So if both sides used ethernet, the MTU of 1500 bytes would allow for a MSS of 1460 bytes without causing fragmentation. Thus the MSS option provides reasonable assurance, that the destination is prepared to accept larger IP datagrams.

This leads to the conclusion, that the data stream must be broken up in multiple TCP segments, which should fit the MTU to avoid fragmentation.

2.1.3 TCP Segment Transmission Algorithms

The reliable *and* efficient transmission of TCP segments relies on multiple algorithms, most of which have been implemented as the result of the research by Jacobson [10]. Because these new algorithms were made mandatory by RFC1122 [11] in 1989, the original ones are not discussed here.

The algorithms controlling the transmission are all sender-side, as it is the responsibility of the sender to reliably deliver the segments. This means, that the sender has to infer the current network conditions from the limited information available. At the beginning of the transmission the bandwidth, condition and delay of the underlying connection are unknown. The algorithm has to probe how much bandwidth is available and can safely be used for transport without incurring losses.

2.1.3.1 Slow-Start

The network probing consists of two phases. The first phase is called *Slow-Start*, although the name contradicts the actual behavior. After the connection is established, a variable, the congestion window (*cwnd*), at the sender is set to one TCP segment. This denotes the amount of data which can be in transit until an acknowledgement is received. The reception of data is acknowledged by sending an ACK packet every two segments. The basic principle of the algorithm is to increase the congestion window by one segment for every ACK received until the window reaches a second variable called slow-start threshold (*ssthresh*). This results in an almost geometrical increase of used bandwidth provided no losses occur.

2.1.3.2 Congestion Avoidance

After reaching the threshold, the growth is reduced to a linear factor. This phase is called *Congestion Avoidance*. If a segment is lost during this phase, the slow-start threshold is reduced and the congestion window is reset to one full sized segment or, in short, to the MSS. A segment is considered lost, if the sender fails to receive an acknowledgement before the timer expires. The value of *ssthresh* after the loss is the maximum of either half of the outstanding, not yet acknowledged, data in the network (flight size) or two sender MSS. RFC2581 [12] states that a common mistake is to use the congestion window rather than the flight size, which may increase *cwnd* beyond the receive window in some implementations.

The algorithm, which implements these measures, is called TCP Tahoe, as it was first implemented in the 4.3BSD-*Tahoe* Unix distribution released by the University of California (Berkeley). The 4.3BSD-*Reno* release in 1990 added *Fast Retransmit*, retransmission of lost segments upon three duplicate ACK segments, and *Fast Recovery*, which sets the congestion window to the reduced slow-start threshold plus three segments after a triple ACK and plus one for each duplicate ACK thereafter. This avoids falling back to *Slow-Start*. The recovery phase ends upon the reception of the cumulative ACK for the outstanding segments and the connection returns to the congestion avoidance state. Both algorithms were documented in RFC2001 [13], which was published seven years later. The repeated process of increase and setback leads to a fair division of available bandwidth if there are multiple concurrent connections competing with one another.

The continued development led to various "flavors" of TCP, such as TCP *Westwood*, *HS-TCP* (HighSpeed TCP) or *BIC-TCP* (Binary Increase Congestion control), which is currently the default implementation in recent Linux 2.6 kernels. Each uses a different approach to achieve maximum throughput, but all are equally sensitive to lost packets. Throughput is reduced, as is the used bandwidth.

2.2 Quality of Service

When a packet leaves the local subnet, it is routed by a dedicated device, a router, towards its destination. These routers queue the packets in their memory and forward them on a best effort basis. The alteration of the forwarding behavior is the subject of QoS. By reordering the queue, thus changing the delay, or deliberately dropping certain packets, QoS attempts to shape traffic patterns to meet preset standards.

The Linux kernel is able to perform both reordering and dropping. However, only packets, which are sent or forwarded, can be delayed. Incoming packets can only be dropped. This can be compared to regular mail. It is not possible to control the amount of letters received, but only the ones sent. QoS induces artificial packet loss and delay, which will eventually reduce bandwidth usage if most packets are TCP segments. However, there is no such mandatory throttling algorithm for User Datagram Protocol (UDP) packets. If a UDP-based protocol has no congestion sensitivity at all, it could consume all available bandwidth even if almost all packets are dropped by the router, which will effectively suppress any TCP connections. Fortunately this is only an absolute worst case scenario and does not reflect the actual portion of UDP traffic in the Internet as shown in section 2.1.

Returning to the algorithms, which implement QoS, the kernel offers multiple choices. They can be categorized in classless and classful queuing disciplines. The first ones are used to enforce a certain allocation policy without subdividing the bandwidth. For example, the Token Bucket Filter (TBF) allows to slow down an interface to an average rate. The Stochastic Fair Queueing (SFQ) is used to attain a fair sharing, in which no single connection can dominate the bandwidth. The classful queuing disciplines include Class Based Queuing (CBQ) [14] and Hierarchical Token Bucket (HTB) [15], which is the algorithm of choice for recent releases of the DynShaper software. In the first releases, CBQ was used, which does shaping based on link idle time calculations. When the behavior became erratic, the switch to HTB was made. This algorithm extends the TBF principle to a hierarchical structure, a tree in which the bandwidth can be divided as needed. It is also possible to allow classes to borrow from their parent classes. How to setup and configure a proper shaping hierarchy to meet the desired policy is beyond the scope of this paper. Documentation can be found in [16] and [17].

2.3 Control Theory

Whoever has to control a restriction, or distribute a limited resource, faces at least three problems [18]. The first one is to set a reasonable limit or to devise a distribution scheme. Secondly, the limit must be controlled. Thirdly, to supervise the control, a measurement device is required. A fourth problem arises if there is a possibility of deviations. In such cases appropriate countermeasures must be developed. In a speed limit scenario the upper bound could be 55 miles per hour, the control would be the accelerator and the monitor the speedometer. As there are always some who ignore any restriction, however reasonable it may be, the countermeasure for speeding are fines.

Monitoring the adherence to a limit, without the involvement of distribution, is a fairly simple task. Distribution can occur in many forms, such as spatial, temporal or per-capita. The number of possible strategies and algorithms performing the actual allocation is even higher. Some of them are more successful than others. For example, if a month's income is spent on the first day, it is impossible to deal with unexpected costs. A carefully devised budget, which takes deferrable and non-deferrable expenses into account, is a more likely approach to succeed. A further distinction can be made, whether a full utilization of the resource is desired or if it should only be used to the required extend. This depends on various factors like the terms of replenishment and reusability of unused portions.

An open-loop controller uses a predefined model and the current state to adjust its input to produced the desired output [19]. It is not suitable, if there are disturbances or errors, because the controller does not observe the actual output. To engage in machine-learning the controller requires feedback to compensate for errors and to correct its inputs accordingly. The classic example is the closed-loop controller, where the feedback from the output is measured by a meter and is fed back into the input, which closes the loop. However, this controller assumes, that the process is linear and time-invariant. Dealing with time-varying systems like traffic control requires an adaptive model [20], which tunes itself to match the altered dynamics of the system [21].

2.4 Case-based Reasoning

Case-based Reasoning (CBR) is one possible solution to such problems. It belongs to the field of artificial intelligence and describes the process of solving a problem based on the experience of similar problems from the past [22]. A cooking recipe is a widely used example for case-based reasoning. The first step is to retrieve a previous solution. Assuming John tries to make spaghetti carbonara. His memory tells him that he has successfully made spaghetti bolognese. In the next step, the retrieved solution is adapted to fit the new problem. In this example the substitution of the sauce. While frying the egg in a pan as he did before with the ground beef, he discovers that the egg has turned solid and thereby unusable for a sauce. Now John proceeds to the next phase and revises his cooking recipe by slowly warming the egg in a cooking pot together with the other ingredients. As this is now successful, he retains this new solution in his memory for further reference. This concludes the four-step process of case-based reasoning [23].

This method is classified as machine learning, as it gains experience by solving problems and storing their solutions in a growing memory. One of the first researchers to mention this concept was Roger Schank [24]. An interesting feature of CBR is, that it resembles everyday human problem solving. Successfully solved past cases are generalized as needed and mapped to the new problem, like having touched a hot stove as a memory and being confronted with a burning candle. In this case most people will decide not to touch the flame because hot things might lead to burns, which are unpleasant at best.

Another similar method is the rule-induction algorithm, which also uses generalized past-cases to solve a new one. The difference between rule-induction and case-based reasoning is the moment of generalization. In the first case, the algorithm is trained with examples which implicitly create a generalized solution. When confronted with a new problem, the rule-induction algorithm can only draw on its precomputed solution, while CBR just begins to retrieve and generalize at this point.

3 Moving Average Control Algorithm Analysis

In this chapter the pros and cons of the Moving Average Control algorithm are discussed. The cons led to the scheduling algorithm in the next chapter and the pros influenced the evolution of the control, which is discussed in chapter 5. First the building blocks of the DynShaper system are briefly described. The second part of the chapter deals with the actual control algorithm and its deficiencies including the motivation for this thesis.

3.1 User Groups

Jan Horbach's diploma thesis [25] was not the first work, which assigned users to groups according to their usage patterns. However, its five groups were considered to be too coarse for a subtle shutdown by bandwidth reduction. Mostly because the reduction steps were to steep and too sudden. The desire for a more fine grained solution resulted in doubling the number of groups to ten.

$$
\begin{aligned}
t_{cls} &= \frac{t_{limit}}{n} \\
t_{cls} &= \frac{1000\,\mathrm{GiB}}{10} \\
&= \underline{\underline{100\,\mathrm{GiB}}} \\
bw_n &= \frac{t_{cls} \cdot 8\,\mathrm{bit}}{30\,\mathrm{d} \cdot 86400\,\mathrm{s}} \\
bw_{10} &= \frac{100\,\mathrm{GiB} \cdot 1024^2 \cdot 8\,\mathrm{bit}}{30\,\mathrm{d} \cdot 86400\,\mathrm{s}} \\
&\approx \underline{\underline{324\,\mathrm{kbit/s}}}
\end{aligned}
\qquad
\begin{array}{lcl}
t_{cls} & - & \text{quota per group} \\
t_{limit} & - & \text{monthly quota} \\
n & - & \text{number of groups} \\
bw_n & - & \text{bandwidth nth group} \\
bw_{10} & - & \text{bandwidth group 10}
\end{array}
\tag{3.1}
$$

The groups are given a starting bandwidth, which can later be scaled by a factor, called the bandwidth factor. At the beginning only the bandwidth of the uplink device is known and assigned to the group for casual users, group number one. As described in [26], the values for the other groups can be calculated. The slowest group (10) is for heavy users, which will most likely use up any bandwidth available (100 % utilization). Because each group shall receive the same portion of the monthly quota, the starting bandwidth of group ten is calculated as in equation 3.1 to allow exactly one n-th of the quota at full utilization. After the boundaries have been determined, the gap in between has to be filled. The usage patterns in [26](ch. 3) show a more or less constant ratio between the number of people in a group and the next one. A geometric progression has a common ratio between each element of its sequence. Thus the bandwidth of group ten is used as the first element of the progression and the interface bandwidth as the tenth element.

$$bw_{n-i} = bw_n \cdot q^i \quad 1 \leq i < n \qquad \begin{array}{lcl} q & - & \text{common ratio} \\ n & - & \text{number of groups} \\ bw_n & - & \text{starting bandwidth group ten} \end{array} \tag{3.2}$$

$$\begin{aligned} q &= \sqrt[i]{\frac{bw_{n-i}}{bw_n}} \\ q &= \sqrt[9]{\frac{bw_1}{bw_{10}}} \\ &= \sqrt[9]{\frac{100\,\text{Mbit/s}}{0.324\,\text{Mbit/s}}} \\ &\approx \underline{\underline{1.9}} \end{aligned} \tag{3.3}$$

Group	Bandwidth	Ratio q	Bandwidth previous Group
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	-	-

Table 3.1: Initial Bandwidths (10 Groups, 1000 GByte quota and 100 Mbit/s Uplink)

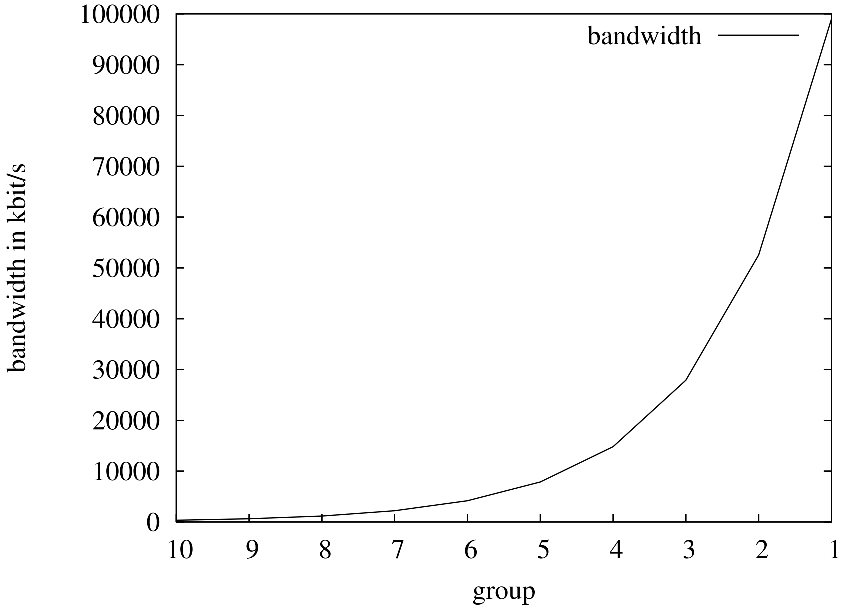

Figure 3.1: Bandwidth Distribution with 10 Groups

The common ratio changes whenever the quota, the number of groups or the interface bandwidth is changed. Most changes resulted from increases of the traffic quota. All values should be in an equilibrium. There is little sense in having a gigabit uplink with only a terabyte traffic quota for 1500 users. An average use of 100 Mbit of such an uplink could theoretically exhaust this quota on a single day. Basically there are limitations to this kind of traffic engineering by dynamic shaping . Whether or not a scenario can be managed, depends on the traffic of the average user, their number and the targeted quota. For example, if the traffic average is at 50 MB per day and there are 200 users, the quota should be 300 GB. This means, that the quota should be at least as high as if every user was an average case. If the quota is lower than that, the scenario will become increasingly difficult to manage. In extreme cases the bandwidth might be constantly lowered without being able to remain within the quota. It also constitutes of an average rate of 118 kB/s or approximately 1 Mbit/s. The uplink should therefore be no faster than 10 Mbit/s to ensure controllability.

3.2 User Classification

The user groups described in the previous section are designed to accommodate certain types of users, ranging from the casual email reader to the peer-to-peer leeching heavy user. While the former has a lower consumption than the average, the latter accounts for most of the traffic. The system uses a 14-day moving average to flatten bursts in the traffic patterns, resulting in a more accurate description of the user type. Thus a user with a one time peak can be discriminated from a user with constant highs.

The premise is that each group receives the same share of the quota. Therefore the users in each group must have the same amount of traffic, which also means that bandwidth must not be borrowed from other groups but only from other users in the same group. Dividing the sum of moving averages of all users by the number of groups gives the amount of traffic each group shall have. The users are sorted by their moving averages in descending order. Then the groups are filled, starting with group ten, by assigning users to it until the traffic generated by the users exceeds the calculated amount of traffic for each group. The algorithms moves on to the next group and continues the assignment up to the group traffic limit. The process is repeated until all groups are filled (figure 3.2).

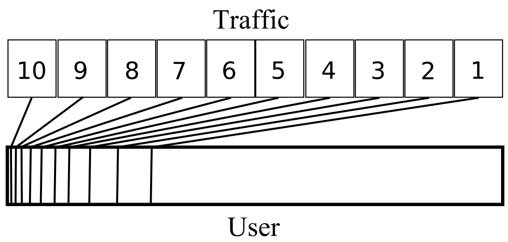

Figure 3.2: User Classification

As time progresses, the moving average changes, thereby possibly violating the above property that each group has the same traffic. User classification should be rerun each time an accounting mechanism has updated the traffic data. If updates are too frequent, an interval of an hour between classification runs is sufficient.

It should be noted, that shaping has also a psychological effect beyond its obvious physical impact. If a user experiences a slow-down of his internet connection, he will most likely do nothing at first, hoping this condition will vanish. If the slow-down is severe enough, the user may be agitated and will search for a reason. Upon discovering, that it is his behavior, the user has the choice to change it or continue. This can result in an avalanche effect, when many users decide to reduce their consumption in order to regain their former bandwidth.

3.3 Limit Control

The bandwidth steps and the continuous reclassification might be sufficient in a scenario with constant traffic patterns. Since such a thing will only exist under lab conditions, the control attempts to adjust the real world to match the desired state. Dividing the monthly quota by the number of days in the current month results in a theoretically allowed daily value, the standard day. The aim of the moving average control is to keep the 14-day average as close as possible to this value.

To provide a means of control, the bandwidths have to be adjusted. The starting bandwidth of each group is multiplied by a common factor, which alters the bandwidth within a defined range. This factor is called the bandwidth factor and has a starting point of one, which has initially no effect. Its boundaries are derived from practical considerations and technical restrictions. The most basic requirement is that the factor must always remain positive and never be zero for neither negative nor zero bandwidths are possible. Starting with the lower bound, the ethernet standard allows a MTU of 1500 bytes for a single IP packet without needing fragmentation, which means that the applicable lower bound is at 12 kbit/s. Multiplied by the number of users in the lowest (slowest) group, will allow each of them to transmit at least one full packet per second. In case of extremely small quotas, the lower bound can be calculated with a 1 kbit/s per user as in equation 3.4, but it should at least equal the aforementioned 12 kbit/s.

$$
\begin{aligned}
f_{min} &= \frac{bw_{min} \cdot u_n}{bw_n} \\
f_{min} &= \frac{1\,\text{kbit/s} \cdot 16}{324\,\text{kbit/s}} \\
&\approx \underline{\underline{0.0494}} \\
f_{min} &= \frac{12\,\text{kbit/s} \cdot 16}{3314\,\text{kbit/s}} \\
&\approx \underline{\underline{0.0579}}
\end{aligned}
\qquad
\begin{array}{lll}
f_{min} & - & \text{minimum factor} \\
bw_{min} & - & \text{guaranteed bandwidth} \\
u_n & - & \text{users in group n} \\
bw_n & - & \text{initial bandwidth group n}
\end{array}
\tag{3.4}
$$

The upper bound, on the other hand, has mostly practical considerations. Firstly, it prevents the bandwidths to rise to astronomical values during extended periods of reduced demand, which would inhibit the ability to exert any control for an extended period of time because of the time required to reach effective levels again. Secondly, it marks a threshold beyond which shaping is futile because it no longer imposes a hindrance to the user. The question is, at which point is the bandwidth, which is used by downloads, approximately equal to the one available? The answer to this question depends on the available technology to the average user. As of 2006, commercial offerings for Asynchronous Digital Subscriber Line (ADSL) internet connections are between one and six megabits per second. Thus, any site has to match its resources to the bandwidth demands if it is willing to keep its customers. The answer to the question when to stop is also a technical one. Martin Devera, the author of the HTB shaping code, explains in [17], how bandwidth is borrowed by shaping classes in proportion of their quanta. It also states that the quantum is an integer number, which is automatically computed as the rate of a class is divided by the global r2q parameter (default 10). Furthermore, it should be as small as possible and larger than the MTU. Given the quantum's maximum of 200,000[1], the highest bandwidth without changing r2q or explicitly specifying a quantum is 16 Mbit/s ($200,000 \cdot 8\,\text{bit} \cdot 10 = 16 \cdot 10^6$ bit).

$$
\begin{aligned}
f_{max} &= \frac{bw_{max}}{bw_n} \\
f_{max} &= \frac{16\,\text{Mbit/s}}{0.324\,\text{Mbit/s}} \\
&\approx \underline{\underline{49.3827}}
\end{aligned}
\qquad
\begin{array}{lll}
f_{max} & - & \text{maximum factor} \\
bw_{max} & - & \text{cutoff bandwidth} \\
bw_n & - & \text{initial bandwidth group n}
\end{array}
\tag{3.5}
$$

Because this value approximately equals 1 Mbit/s per user in the highest group (average 12.38 users) at full concurrent use, it is used as cutoff value (equation 3.5) for the bandwidth factor. If the bandwidth of a group is above the cutoff bandwidth, shaping rules are no longer created. Shaping is completely suspended upon reaching the maximum factor as no group is any longer below the threshold. The factor remains at this value until reduced again by the control.

3.3.1 Linear Adjustment

After defining its range and before discussing its shortcomings, the rules of the control algorithm must be described. As the control is based on maintaining the daily average, its allowed value is computed at first by dividing the monthly quota by the number of days in the current month.

[1] `net/sched/sch_htb.c` in Linux kernel sources

$$t_d = \frac{t_{limit}}{d_m} \qquad \begin{array}{lll} t_d & - & \text{allowed daily consumption} \\ t_{limit} & - & \text{monthly quota} \\ d_m & - & \text{days in month} \end{array} \tag{3.6}$$

Based on the data of an accounting solution (such as Cisco Netflow[2]) the 14-day moving average of all users is computed in the next step. Whether this includes only download or both directions, can be changed at implementation level.

$$k = \frac{t_d}{t_{avg_{14}}} \qquad \begin{array}{lll} k & - & \text{adjustment} \\ t_d & - & \text{allowed daily consumption} \\ t_{avg_{14}} & - & \text{14-day moving average} \end{array} \tag{3.7}$$

Multiplying the current bandwidth factor with the computed adjustment k (3.7) results in a new bandwidth factor. If the new factor is not within the aforementioned boundaries, its value is reset to minimum or maximum value, respectively. After this check the factor is multiplied with the starting bandwidth of each group.

$$f_{new} = f_{old} \cdot k \qquad \begin{array}{lll} f_{new} & - & \text{new bandwidth factor} \\ f_{old} & - & \text{current bandwidth factor} \\ k & - & \text{adjustment} \end{array} \tag{3.8}$$

3.3.2 Power Law Adjustment

The average method is a continuous control scheme. There are no scheduled resets and no forecasts, which would allow to take preemptive measures upon impending major changes in traffic consumption. Therefore the control requires more leverage in order to lower the bandwidth factor faster. The reduction has to be swift as there might not be enough time left to make up for any excess traffic. In order to detect changes early, an indicator is needed. The seven-day moving average is chosen to reflect the current trend in contrast to the slowly affected 14-day moving average.

[2] `http://www.cisco.com/go/netflow`

$$r = \frac{t_{avg7}}{t_{avg14}} \qquad \begin{array}{lcl} r & - & \text{rise} \\ t_{avg7} & - & \text{seven-day moving average} \\ t_{avg14} & - & \text{14-day moving average} \end{array} \tag{3.9}$$

The ratio of both values (3.9), the rise, is used as base for the exponentiation designed to give more leverage to the adjustment value k. However, the adjustment is only altered if the ratio is $g \geq 1.01$ and there has been at least one day in the past seven days, which has exceeded the allowed daily consumption.

$$k_e = \frac{k}{r^4} \qquad \begin{array}{lcl} k_e & - & \text{amplified adjustment} \\ k & - & \text{adjustment} \\ r & - & \text{rise} \end{array} \tag{3.10}$$

In case of a positive trend (above one), the adjustment is exponentially amplified by the rise ratio, which allows for a rapid reduction of the bandwidth factor.

3.4 Problem Cases

Before discussing the errors, it should be noted, that the average control method has served its purpose magnificently in the past five years at its birth site, the CSN. It has never failed to keep the traffic consumption below the limit set by the quota, albeit it did not operate directly with the quota. In spite of its apparent success, there were issues, which questioned the control's ability to fulfill its purpose under all circumstances. Fortunately, such events never occurred. The more tangible problems surfaced in the last two years, when the quota increases were not as high as before and the shaping was operating continuously for the first time. In contrast to the first issue, those problems concerned the inefficiency caused by the mathematics within the algorithm. Because these flaws recurred every year in a predictable fashion, they motivated the author to seek a solution, which led to this thesis.

3.4.1 Possible Quota Violation

The earliest problem case is a scenario concerning the fact, that the control lacks any notion of how much traffic remained for a given month. By operating solely on an allowed daily value, it provides an opportunity to exceed the quota without being aware of it. The problem exploits the fact, that the linear adjustment is slow, too slow to lower the bandwidth fast enough to prevent a transgression.

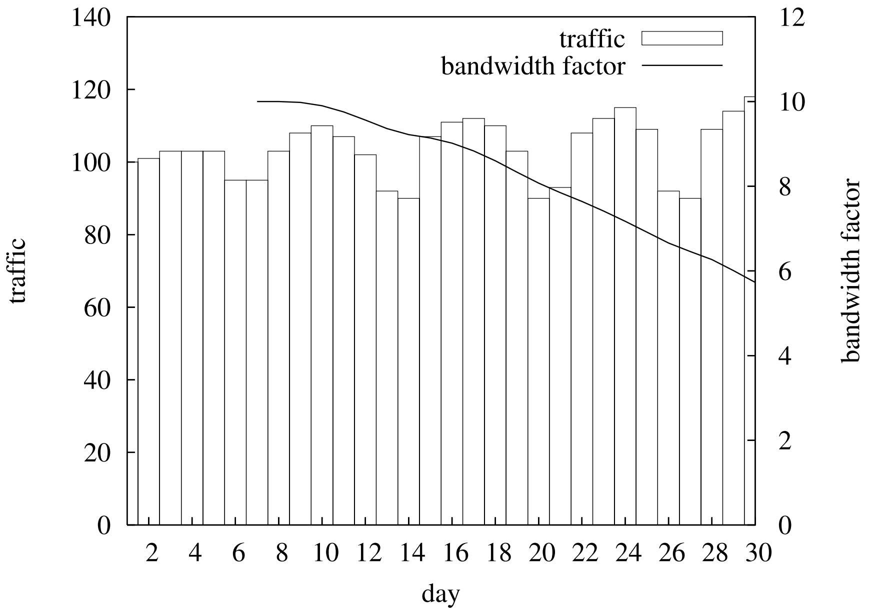

Figure 3.3: Scenario with Quota Violation

Figure 3.3 shows an exemplary scenario, in which the quota is exceeded by 3.5 %. Small increases in consumption, which stay just below the threshold of 1.01, prevent a swift intervention by the amplified adjustment value (table 3.2).

day	traffic	7-day avg	14-day avg	adjustment	rise	final adj.	factor
1	100	100					
2	101	100					
3	103	100.14					
4	103	100.57					
5	103	101					
6	95	101.43					
7	95	100.71					10
8	103	100	100	1	1	1	10
9	108	100.43	100.21	0.9979	1.0021	0.9979	9.9786

10	110	101.43	100.79	0.9922	1.0064	0.9922	9.9008
11	107	102.43	101.5	0.9852	1.0091	0.9852	9.7545
12	102	103	102	0.9804	1.0098	0.9804	9.5632
13	92	102.86	102.14	0.979	1.007	0.979	9.3626
14	90	102.43	101.57	0.9845	1.0084	0.9845	9.2178
15	107	101.71	100.86	0.9915	1.0085	0.9915	9.1394
16	111	102.29	101.36	0.9866	1.0092	0.9866	9.0171
17	112	102.71	102.07	0.9797	1.0063	0.9797	8.8341
18	110	103	102.71	0.9736	1.0028	0.9736	8.6006
19	103	103.43	103.21	0.9689	1.0021	0.9689	8.3328
20	90	103.57	103.21	0.9689	1.0035	0.9689	8.0733
21	93	103.29	102.86	0.9722	1.0042	0.9722	7.849
22	108	103.71	102.71	0.9736	1.0097	0.9736	7.6416
23	112	103.86	103.07	0.9702	1.0076	0.9702	7.4139
24	115	104	103.36	0.9675	1.0062	0.9675	7.1731
25	109	104.43	103.71	0.9642	1.0069	0.9642	6.9162
26	92	104.29	103.86	0.9629	1.0041	0.9629	6.6593
27	90	102.71	103.14	0.9695	0.9958	0.9695	6.4564
28	109	102.71	103	0.9709	0.9972	0.9709	6.2684
29	114	105	104.36	0.9582	1.0062	0.9582	6.0066
30	118	105.86	104.86	0.9537	1.0095	0.9537	5.7284
Sum	3105						
Quota	3000						
Avg	103.5						

Table 3.2: Quota Violation Scenario

For this scenario it was assumed, that previous consumption was constantly at the allowed maximum (100). The second assumption was that the bandwidth factor is also at such a high value. Although the factor is slowly dropping, the consumption does not respond to this in any way. This is not unfeasible, but a common phenomenon. If the bandwidths of the groups are evenly spaced approximately doubling each time or more, the reduction of the factor by less than 50 % at the end of the month has only affected the slowest group, which represents 10 % of the consumption. It reduced its bandwidth to half of the cutoff bandwidth, which is in this case still five times more than the initial one.

Without changing the algorithm there are only a few possibilities. One of them is to use a safety margin and lower the allowed daily consumption slightly to create a cushion for errors. The price for this safety is a constantly lowered efficiency for a highly unlikely but possible scenario. However, the concept of a safety net should not be dismissed, especially because this thesis aims to raise the utilization to the brink of overstepping the quota. In fact, even if the consumption is within normal standards, there is always room for unexpected events such as highly anticipated software releases or news coverage of exceptional events. In the light of trying to predict traffic consumption by planning and forecasting techniques, there is always the chance of misprediction. The margin should therefore cover as much ground as possible without lowering the average use of the quota significantly.

The second option would be to lower the threshold after which the amplified adjustment is used. Considering the distribution of the bandwidths to each group, it might also be conceivable to change the control altogether to accommodate this fact more effectively. But having a more sensitive control will most certainly lead to a higher degree of oscillations, which are currently dampened by the controls reluctant behavior towards small inputs. Of course there are techniques to counteract oscillation, but the current control's spotless track record speaks for itself. Alterations should therefore be done with the utmost care.

3.4.2 Non-uniform Traffic Distribution

One of the primary motivations for this thesis is to ready the DynShaper software to handle the non-average case. As long as nothing out of the ordinary occurs, the current control performs just fine. During the semester the traffic cycles each week. Many students return home on the weekend to their families and traffic consumption drops. On Mondays, or Tuesdays at the latest, they return. There are occasional blips on the radar when a public holiday falls on a Friday or Monday, but those have little effect in a monthly scale. However, other, more extended periods of altered traffic consumption have a profound impact on the accustomed cycle. Some take place at changing times, such as Easter, others always come about the same date. As far as it concerns this thesis, Easter is just a two day public holiday which extends a weekend, but it is not followed or preceded by additional free days for students. This means, that instead of the occasional non-standard day, there are two. A 50 % reduced consumption on both days (standard day) will only constitute 3–4 % of the quota, still yielding a possible 97 % utilization, which is considered quite efficient. In short, all those little disturbances and diversions have not enough impact to warrant a change in control. There is only one event every year, which radically lowers the consumption in a predictable manner to consider seeking a way to take advantage of this knowledge: Christmas (figure 3.4).

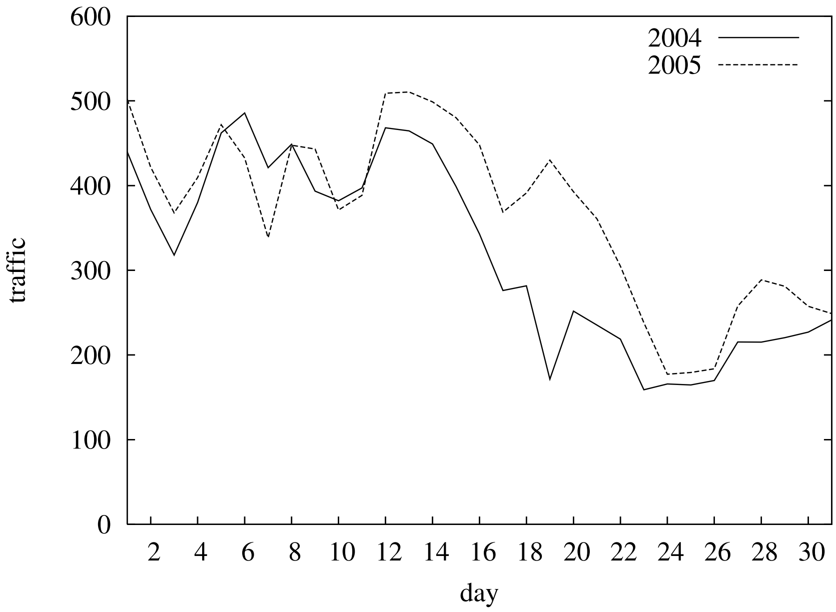

Figure 3.4: December Scenario

The difference between Christmas and other holidays is, that it covers a full week in December (24th to 31st) plus a non-specified number of lecture-free days before Christmas Eve and after New Year's Day. However, the focus is on the free days in December for this case. As figure 3.4 shows, the consumption rapidly decreases in the week before Christmas Eve and remains at this level for the rest of the month. In order to improve the utilization in December the unused portion has to be redistributed to days in need for more traffic, namely those in the first two and a half weeks. Without altering the mechanism, a human operator could raise the quota, which would trick the control into allowing more traffic per day than usually. Unfortunately this is also the disadvantage of this solution, because it requires human interaction, which is not only frowned upon but also regarded as inelegant. The operator would also have to monitor traffic continuously, keep track of the aggregated consumption in relation to the quota and maybe adjust the bandwidth factor manually in case something goes wrong. All this during a time, which most people tend to spent with their families. Monitoring and adjusting a controller are, on the other hand, tasks at which a machine is very good at. A machine, or an algorithm in this case, will need specific instructions how to redistribute the unused traffic portion or an external function, which calculates this in advance and presents the control with a plan that is ready to execute. Thus the second possible solution requires the development of additional code in the control software, which increases complexity, but once it is deployed, it will work independently, yielding the same efficiency without human supervision.

3.4.3 Overregulation

It is a common saying that one problem leads to another. The final issue of the average-based control always occurs in January after the Christmas season. In the last part of December, the bandwidth factor climaxes due to the reduced consumption, which the raised factor is supposed to counter. However, this raise has virtually no effect on traffic as most students are no longer in the dormitory, so the factor continues to climb until reaching its maximum value. When the inhabitants return in the first days of January, their 14-day moving average is a clean slate. The shaping system is completely unaware of the events about to happen. Traffic consumption returns to its pre-Holiday heights without warning, or worse to even higher values because of the missing restraints from shaping. This is quickly reflected by the 7-day moving average, which in turn affects the rise ratio. The ratio hurdles over the 1 % threshold with ease and amplifies the adjustment value dramatically, lowering the bandwidth factor close to its minimum within a few days (figure 3.5). While it is initially correct to reduce the bandwidth to regain control of the consumption, the algorithm overshoots the mark. In the next phase the factor rises again slowly, but there is little reaction at first. Then suddenly, as figure 3.5 shows, the traffic consumption picks up again. The rise ratio passes the threshold and a single over-limit day lets the amplified adjustment sever the bandwidth factor again. The ripple continues into early February where it ceases at the end of the lecture period.

The solution to this problem cannot be achieved without changing the algorithm itself. It would require foresight on behalf of the control to anticipate the returning students and to reduce the factor prior to that. Because the controller does not have this insight, it cannot do so. The other solution would include modifying the mathematics, which trigger the amplification. For example, instead of a single over-limit day in the past seven days, the trigger could be altered to fire upon the seven day moving average violating the allowed daily average. In this case, the downward movement would stop two days earlier in the above scenario. The short term

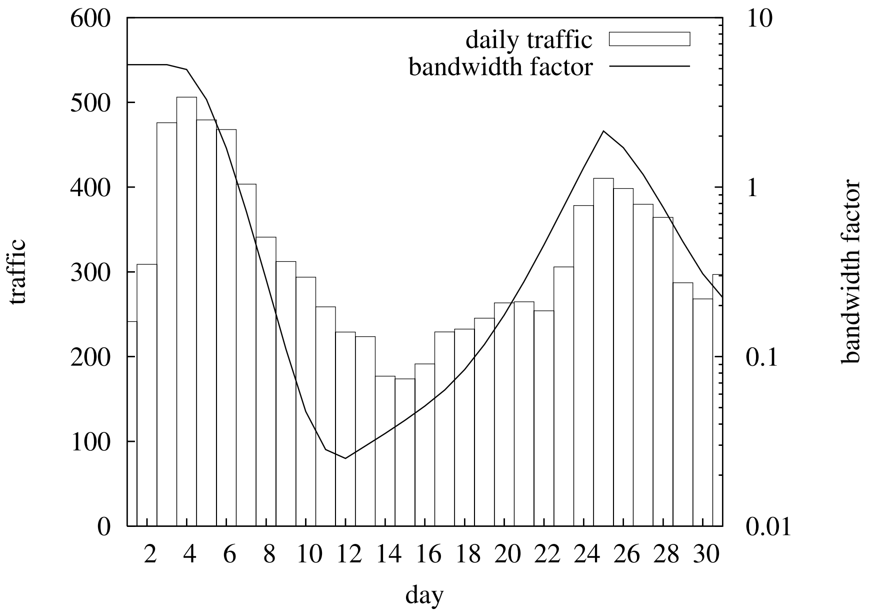

Figure 3.5: Overregulation Scenario (Jan 2005)

overspending would have been made up and therefore the need to continue the reduction. The second reduction also might never have taken place, but maybe only at a later point in time. The problem is, that the bandwidth factor starts out low after the panic cutoff has occurred. The traffic consumption will remain reduced for a longer time than necessary and whether this damage can be undone until the end of the month is impossible for the average-based control to decide.

3.5 Conclusion

The average-based control is a fair and just system. Unfortunately, its global adjustment part has defects in certain situations. While the first one presented has never happened and requires a number of limiting conditions to be occur, the other two recur every year. Although a mere nuisance as inefficiency has no detrimental consequences, the author strives to perfect this software. The motivation is to be able to handle every scenario encountered and potentially every other conceivable and manageable scenario.

4 Traffic Schedule/Forecast

In this chapter the algorithm, which is used to create a schedule, is presented. The terms forecast, prediction and schedule are used interchangeably to describe the same process. Normally it is not appropriate to do so, but in this specific scenario, the describing property of the algorithm changes with the situation. In case of a moderate quota increase, the algorithm forecasts the probable consumption along with a likely sufficient bandwidth factor. But if the limit is raised significantly (e.g. 50 %), the result only predicts a possible course bearing a resemblance to the general pattern for that month. In any other case, the schedule sets the maximum allowed traffic amount for a given day. Because forecast methods were not applied in this particular scenario, the next sections also present the results of classic statistical analysis and discuss why they cannot be used.

4.1 Statistical Traffic Analysis

The primary question, which had to be answered first, was whether traffic has a recurring pattern or not. Figure 4.1 shows the traffic development in the last five years. The data of the first years shows little fluctuation, while there is a distinct pattern visible in recent years. The most important observation however is, that the cycle is constant and not shifting. While the seasonal fluctuations of 2001 are barely discernable in this figure, there are clear highs and lows after 2004. This figure is also a first example, why forecasts do not apply. In November 2004 the monthly consumption peaked at about 12 TiB. Every year there was a steady increase, so the level of November 2005 should have been above the 2004 level. However, as the quota had not been raised at this time, the DynShaper had to control the bandwidth to keep the limit.

Figure 4.2 shows the pattern in 2001 and 2005, this time given each its own axis. As expected, the patterns match. Although it must be noted, that there are possibly problematic differences as in February. A forecast based on the 2001 values will certainly have serious discrepancies if the pattern resembles the 2005 course.

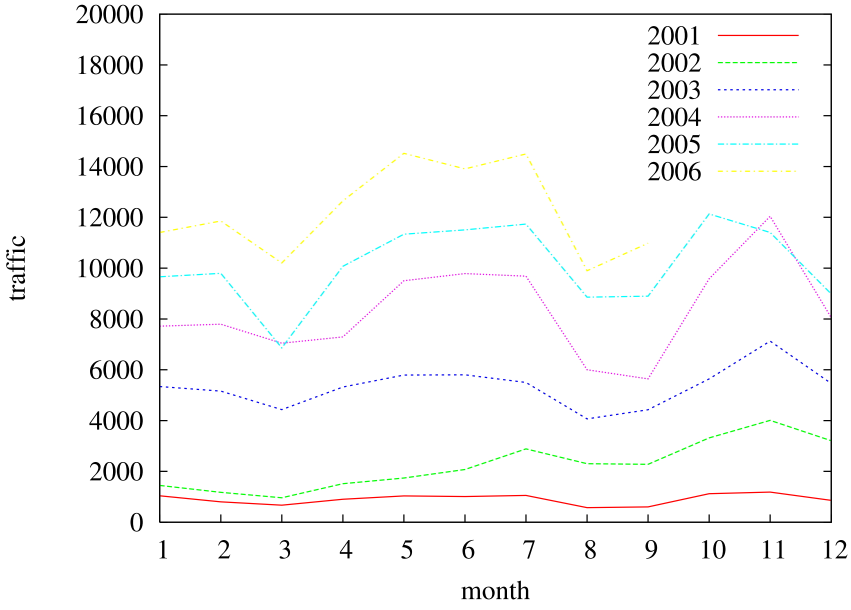

Figure 4.1: Monthly Traffic Consumption

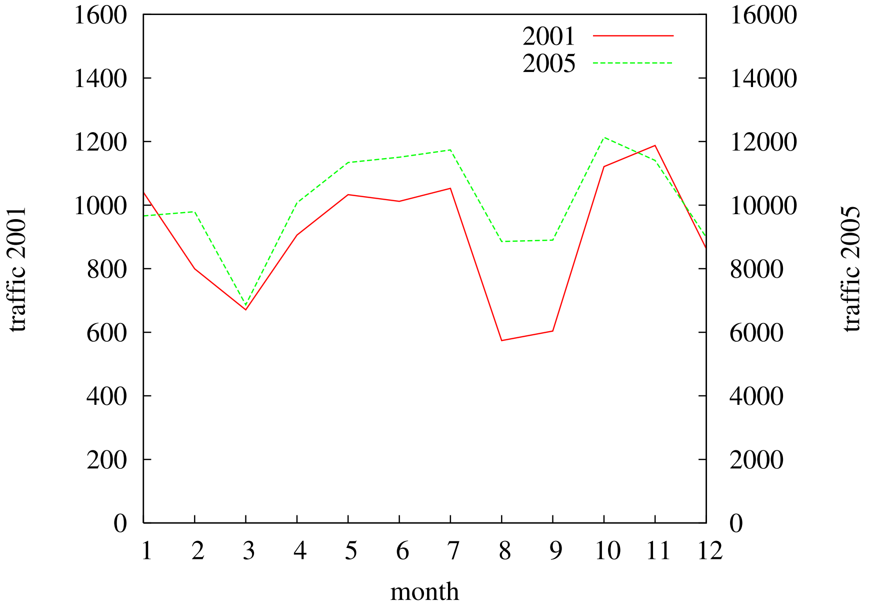

Figure 4.2: Monthly Traffic Consumption

Further analysis reveals also a recurring pattern within each month. Regular occurrences include decreased consumption on the weekends and significant drops on holidays, like Christmas. As an example for this, figure 4.3 displays the traffic patterns of June.

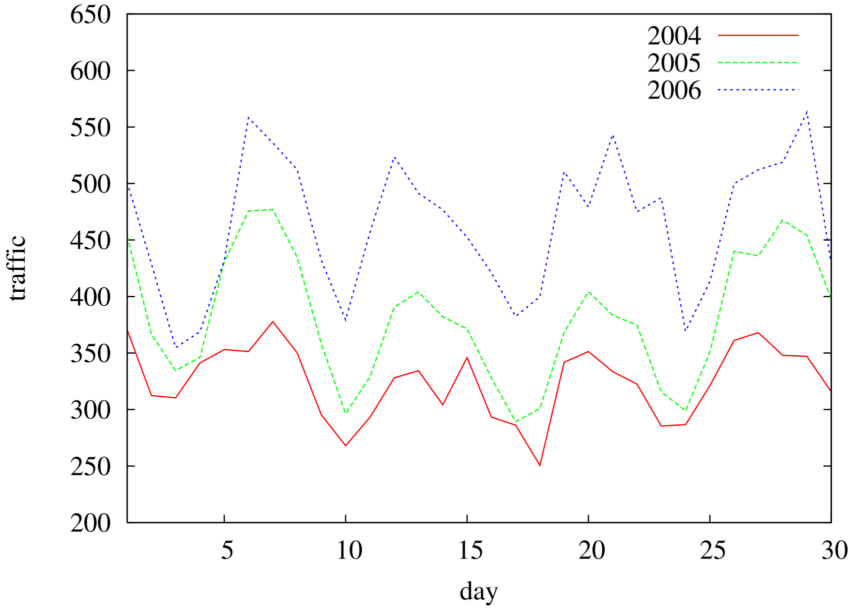

Figure 4.3: Daily Traffic Consumption in June

Similar to the monthly cycle, the differences between the highs and lows have increased. Most importantly, the peaks coincide when the data is date-adjusted like is was done in this figure. However, there is no exact repetition, albeit the general pattern matches.

Time Series Analysis

In order to learn more about the traffic patterns, the statistical data was subjected to a time series analysis [27][28]. It is used to identify the structure and the causes of time-dependent data by separating the data into two components: trend and seasonality. The seasonal component can be extracted after the general trend has been eliminated from the data. According to figure 4.1, it can be assumed that there is a linear trend, which can be estimated by linear regression [29].

$$\hat{y}_t = a + b \cdot t \; (t = 1, 2, \dots, T) \tag{4.1}$$

$$
\begin{aligned}
b &= \frac{\sum_{t=1}^{T}(t-\bar{t})(y_t-\bar{y})}{\sum_{t=1}^{T}(t-\bar{t})^2} \\
&= \frac{\sum_{t=1}^{T} t\cdot y_t - T\cdot\bar{t}\cdot\bar{y}}{\sum_{t=1}^{T} t^2 - T\cdot\bar{t}^2} \\
&= \frac{\sum_{t=1}^{T} t\cdot y_t - \frac{T(T+1)}{2}\bar{y}}{\frac{1}{12}(T^3-T)}
\end{aligned} \tag{4.2}
$$

The linear function [30] (equation 4.1) has two variables, which must be determined. The slope of the trend b is calculated as in equation 4.2. The value of the displacement a is derived by subtracting the function value from the observed value.

$$
\begin{aligned}
a &= \bar{y} - b\cdot\bar{t} \\
&= \bar{y} - b\cdot\frac{T+1}{2}
\end{aligned} \tag{4.3}
$$

	2001	2002	2003	2004	2005	2006
1	$1.1169\cdot10^{12}$	$1.5570\cdot10^{12}$	$5.7360\cdot10^{12}$	$8.2840\cdot10^{12}$	$1.0374\cdot10^{13}$	$1.2243\cdot10^{13}$
2	$8.5843\cdot10^{11}$	$1.2634\cdot10^{12}$	$5.5427\cdot10^{12}$	$8.3688\cdot10^{12}$	$1.0515\cdot10^{13}$	$1.2730\cdot10^{13}$
3	$7.1996\cdot10^{11}$	$1.0313\cdot10^{12}$	$4.7580\cdot10^{12}$	$7.5635\cdot10^{12}$	$7.3691\cdot10^{12}$	$1.0954\cdot10^{13}$
4	$9.7260\cdot10^{11}$	$1.6292\cdot10^{12}$	$5.7110\cdot10^{12}$	$7.8302\cdot10^{12}$	$1.0815\cdot10^{13}$	$1.3581\cdot10^{13}$
5	$1.1089\cdot10^{12}$	$1.8681\cdot10^{12}$	$6.2179\cdot10^{12}$	$1.02052\cdot10^{13}$	$1.2174\cdot10^{13}$	$1.5587\cdot10^{13}$
6	$1.0865\cdot10^{12}$	$2.2299\cdot10^{12}$	$6.2317\cdot10^{12}$	$1.0509\cdot10^{13}$	$1.2353\cdot10^{13}$	$1.4934\cdot10^{13}$
7	$1.1303\cdot10^{12}$	$3.0970\cdot10^{12}$	$5.9054\cdot10^{12}$	$1.0397\cdot10^{13}$	$1.2600\cdot10^{13}$	$1.5564\cdot10^{13}$
8	$6.1599\cdot10^{11}$	$2.4753\cdot10^{12}$	$4.3649\cdot10^{12}$	$6.4447\cdot10^{12}$	$9.5093\cdot10^{12}$	$1.0626\cdot10^{13}$
9	$6.4819\cdot10^{11}$	$2.4494\cdot10^{12}$	$4.7529\cdot10^{12}$	$6.0587\cdot10^{12}$	$9.5530\cdot10^{12}$	$1.1789\cdot10^{13}$
10	$1.2036\cdot10^{12}$	$3.5655\cdot10^{12}$	$6.0564\cdot10^{12}$	$1.0288\cdot10^{13}$	$1.3025\cdot10^{13}$	$1,7244\cdot10^{13}$
11	$1.2750\cdot10^{12}$	$4.3055\cdot10^{12}$	$7.6526\cdot10^{12}$	$1.2930\cdot10^{13}$	$1.2247\cdot10^{13}$	
12	$9.2603\cdot10^{11}$	$3.4441\cdot10^{12}$	$5.8658\cdot10^{12}$	$8.6477\cdot10^{12}$	$9.6468\cdot10^{12}$	

Table 4.1: Monthly Traffic Consumption 2001 - 2006

The result of these equations and the values in table 4.1 is:

$$\hat{y}_t = -8,31614\cdot10^{11} + 2.17528\cdot10^{11}\cdot t \tag{4.4}$$

By looking at the differences of each year, the average increase between the years is about 85 %. However, the trend has declined. From 2001 to 2002 the average increase was at 150 %, peaking at 300 % in August. Since 2003 the increases have been more moderate, 55 % to 2004 and 23 % to 2005. Nevertheless, it would be a gross misinterpretation to attribute the lowered increases to saturation. In reality the quota was doubled in 2002 and 2003 with moderate raises in the following years. The conclusions from these observations are, that the users are easily capable of exhausting a quota twice the current size within a year and that shaping is the only thing that stops them. This means, that the true trend value is not determinable as it biased by shaping to an uncertain degree. So unless the quota increases exponentially, it must be assumed that the quota will be exhausted unless traffic consumption is restrained.

4.2 Linear Scaling

The positive linear trend revealed by the regression, though very obvious, has a simple implication. The traffic estimate for a month is the statistical value from the previous year multiplied by a constant and in this case modified by an increasing seasonal component. But there are two cases in which a correct prediction does not make sense.

1. months where the utilization is known to be less than quota
2. months where the prediction is higher than the quota

In the first case the control would not raise the bandwidth as no underutilization is detected, although there is unused traffic which could be used by anyone without repercussions. The latter is even worse, as it can happen quite easily, if the annual quota increase is canceled. The worst case would be, if the quota is lowered for some reasons. In any case, no forecast technique known to man could predict such events from existing data. Therefore classic forecast methods are deemed inadmissible.

Nevertheless these methods render some valuable ideas. Linear regression is a simple increase by a constant. To factor in the quota, the multiplication with the constant should render a result that equals the quota. The easiest way to reach this is to divide the quota by the traffic sum of the previous year (equation 4.5).

$$c = \frac{t_{limit}}{t_{sum}} \qquad \begin{array}{lll} c & - & \text{scaling factor} \\ t_{limit} & - & \text{traffic quota} \\ t_{sum} & - & \text{traffic consumption previous year} \end{array} \tag{4.5}$$

Before applying the equation the day shift has to be compensated taking the sum from the second day to the first of the next month. The exception being February in leap years. After determining the constant, each daily value is multiplied by it, thus giving the allowed maximum value for each day.

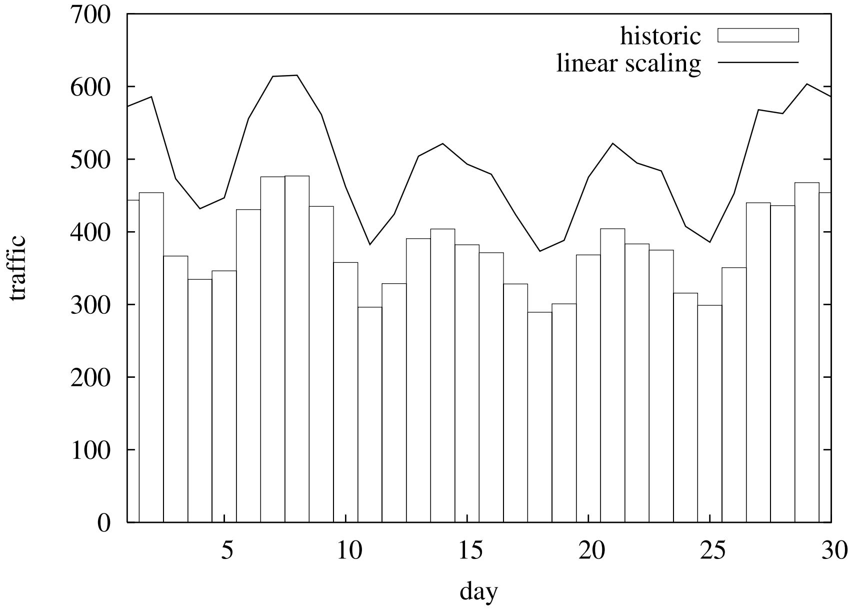

Figure 4.4: Forecasting by Linear Scaling (Data Sample: June)

4.3 Compensational Scaling

While the linear scaling works as long as there is no shaping involved, it yields false results if the data is heavily biased. A forecast based only on the traffic pattern does not account for the bandwidth factor. In figure 4.5 the impact of the reduced bandwidth is clearly visible and yet the highest consumption was on the 17th. This must be taken as indication, that the bandwidth factor has no direct correlation to a particular traffic consumption. What can be deduced is, that there is always a certain demand for traffic, which is dampened to an unknown degree by the shaping groups in use. On a day near or at the maximum, the demand is satisfied. The reason for this, that shaping starts, when the resulting bandwidth of a group drops below the cutoff threshold. As long as there is only one group subject to shaping, the effects pertain to only one n-th, here 10 %, of the traffic. The actual impact on this group is also limited, as the bandwidth is still comparatively high. So for all practical purposes the traffic data of those days can be considered unbiased.

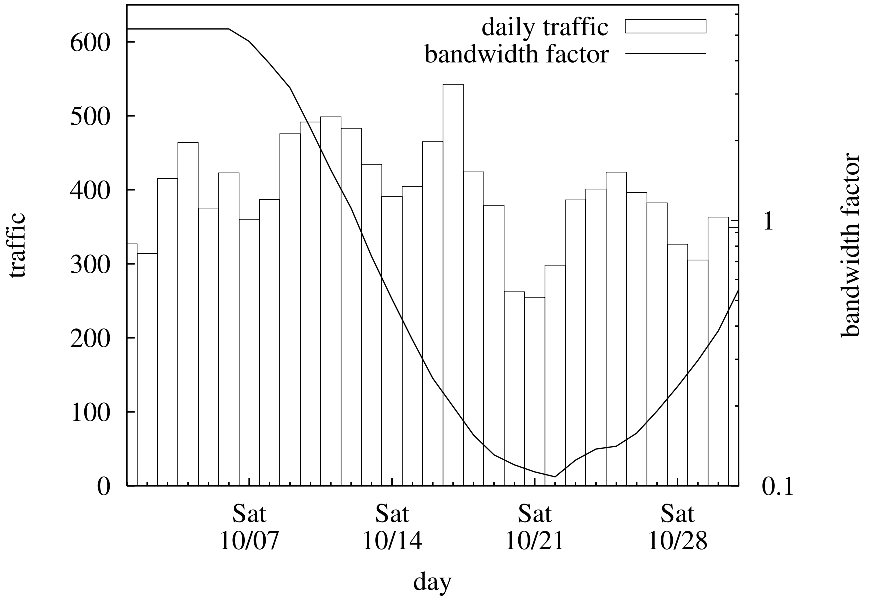

Figure 4.5: Traffic Consumption and Corresponding Bandwidth Factor in October 2005

On any other day, the data is tainted in some degree and thereby falsifies the forecast methods based on statistics. The statistical data must be untainted in some manner in order to be useful at all. However, the only indication, the bandwidth factor, does not have a linear scale as described in section 3.1 and has psychological side effects (3.2). Secondly, the quota constraint requires the forecast to remain below the quota, something a forecast is not intended for. This leads to the conclusion, that the data has to be processed to account for both.

The linear scaling, described in the previous section, could theoretically be used in spite of the biased data if the bandwidth factor were always constant. This has only happened to date, when the quota was far away from being exhausted. For any other month the aim is to approximate the deviation from the ideal, which in this case could be either the mean or median factor. How to determine the deviation depends on whether there is traffic left to be distributed or if it has to be taken away. In the first case, the days with the lowest factor shall receive the highest contribution and in the latter case the lowest deduction.

$$t_{diff} = t_{limit} - t_{sum} \tag{4.6}$$

t_{diff}	-	difference
t_{limit}	-	traffic quota
t_{sum}	-	traffic consumption previous year

If the remainder is positive, the share per deviation is calculated as in equation 4.7 or if negative as in 4.8.

$$t_{share} = \frac{t_{diff}}{\sum_{k=1}^{n} \frac{1}{f_k}} \tag{4.7}$$

$$t_{share} = \frac{t_{diff}}{\sum_{k=1}^{n} f_k} \tag{4.8}$$

The share is then multiplied by the factor or its reciprocal value and the result is finally added to the traffic value of the previous year (equations 4.9 and 4.10).

$$t_{new} = t_{share} \cdot \frac{1}{f_k} + t_{old} \tag{4.9}$$

$$t_{new} = t_{share} \cdot f_k + t_{old} \tag{4.10}$$

Example 4.1

$$t_{diff} = 3000\,GiB \qquad \sum_{k=1}^{n} \frac{1}{f_k} = 120$$

$$t_1 = 415\,GiB \qquad t_2 = 360\,GiB$$

$$f_1 = 1.5 \qquad f_2 = 0.11$$

In the first step, the share is calculated.

$$
\begin{aligned}
t_{share} &= \frac{3000\,GiB}{120} \\
&= 25\,GiB
\end{aligned}
$$

In the next and finally step, the new estimate is calculate

$$
\begin{aligned}
t_{2_{new}} &= 25 \cdot \frac{1}{1.5} + 415\,GiB & \qquad t_{2_{new}} &= 25 \cdot \frac{1}{0.11} + 360\,GiB \\
&\approx 432\,GiB & &\approx 587\,GiB
\end{aligned}
\tag{4.11}
$$

Continuing the example on the values of October yields the following results:

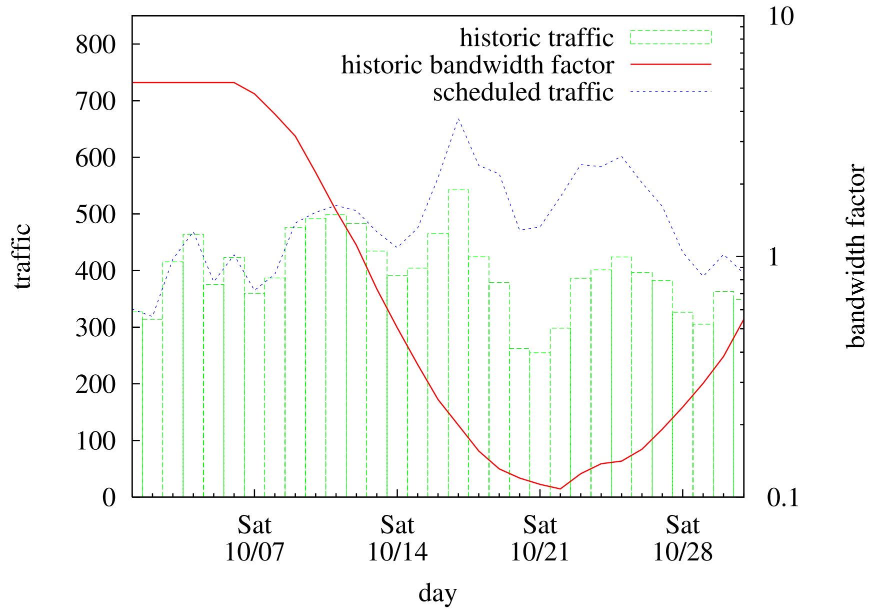

Figure 4.6: Compensational Forecast (October 2006)

As the example shows, days with low bandwidth factors receive higher contributions than those with higher than average ones. In most cases this distribution scheme fits its purpose very good, especially when the amount of traffic to be distributed is small (less than 25 % of the quota). In those cases, almost all of the available traffic goes to low factor days, which is exactly the desired effect. The other benefit of distributing the remainder per share is, that it yields a ready to use schedule for the control.

There are, however, some undesired side effects if the marginal conditions are not met. As noted above, the distribution scheme works very well if the amount is less than 25 %. If the raise exceeds 30 % the distribution deteriorates exponentially, because the share increases while the number of shares per day remains constant. This results in a disproportionate schedule as in figure 4.7.

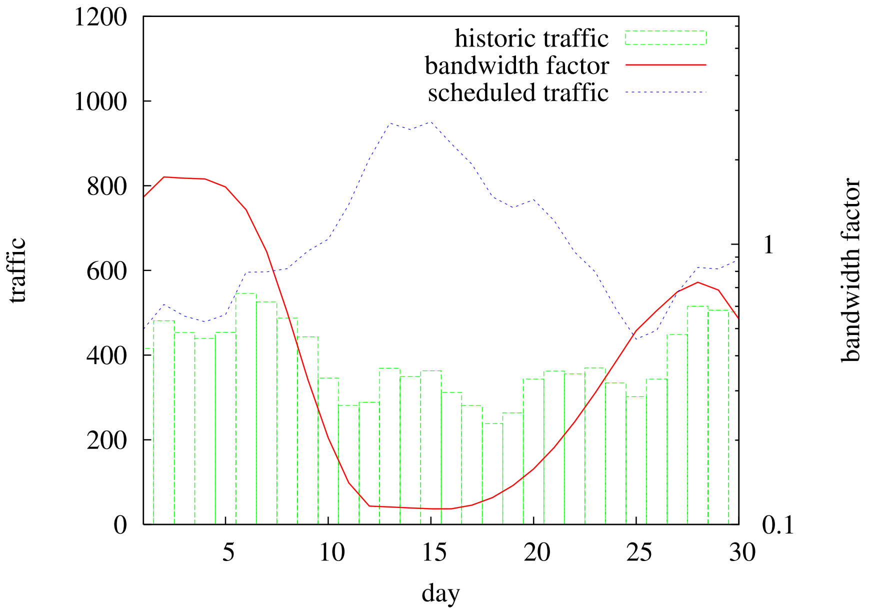

Figure 4.7: Deteriorated Compensational Forecast

The second issue, which affects this method adversely, occurs when the statistical data contains modifications of the bandwidth factor which are not the result of the control algorithm. These user induced readjustments of the bandwidth factor have usually been done with a well-meaning intend to correct obvious misregulations of the control algorithm. The most common modification is a reset to factor one, or some other (higher) factor, in case of overregulation. The result of the adjustment is visible in the form of a sudden leap in the graph of the bandwidth factor and results in a single out of band peak on the day before the modification in the predicted schedule (Figure 4.8).

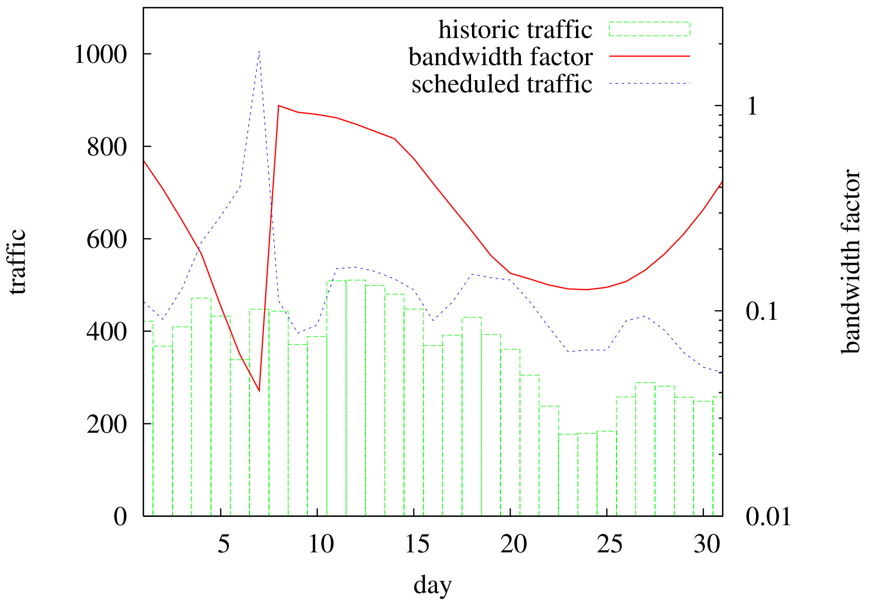

Figure 4.8: Compensational Forecast with Spike

4.4 Hybrid Forecast

As the previous section explains, there are some undesired side effect, which can happen if the input data deviates. The author estimates that there is a very good chance that a quota increase may be higher than it has been in the previous years, because the Deutsches ForschungsNetz (DFN) has stopped to employ quotas as a limiting factor for its internet access products in 2006. Because the involved parties still feel the need to cap the consumption, a quota will still be in effect, albeit the increases may be scheduled in shorter intervals if the demand warrants them.

The final forecast method tries to accommodate both concepts presented in the previous sections. The time series analysis in section 4.1 proved, that there is a constantly rising demand, which was addressed by the forecast method presented in section 4.2. The second method (section 4.3) used a distribution scheme to account for the reduced bandwidth. It estimates a likely consumption without shaping, but its focus is primarily on the days with lower bandwidth than others.

Both methods result in a valid schedule, which means, that each one calculates its adjustments in a way that the sum of all estimates in the schedule matches the quota. Merging two valid schedules for a given month must also result in a valid schedule. For practical purposes and in the interest of future maintainers of this software, a simple mathematical operation was chosen to achieve this result. Namely to average over both estimates for each day. In spite of its simplicity, the result satisfies its intended purpose well enough. The final schedule still matches the quota and the overcompensation for low bandwidth is reduced to an acceptable degree. Figure 4.9 displays an example of the three (scaled, compensated and merged) schedules.

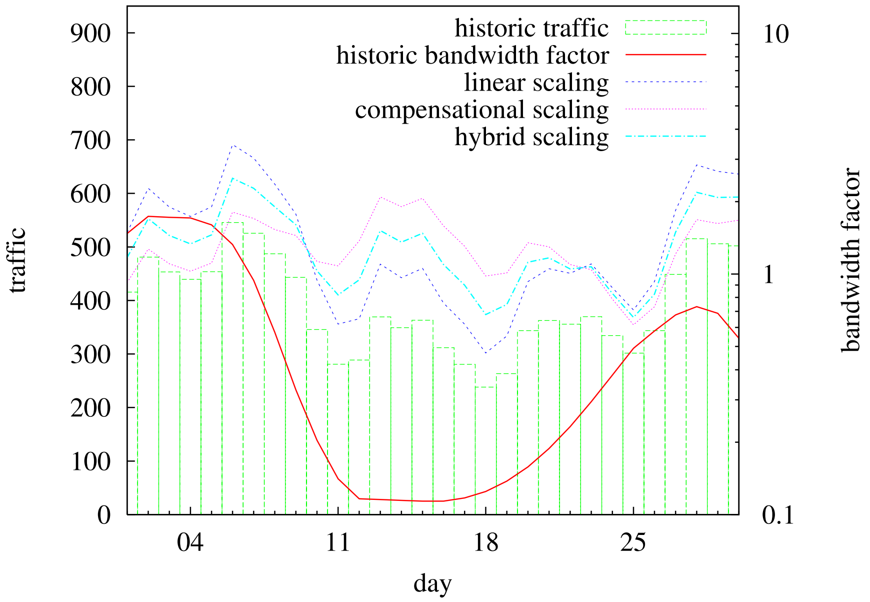

Figure 4.9: Forecast Displaying All Techniques

4.5 Common Operations and Conclusion

One of the goals of this thesis, and the DynShaper software, is to maximize utilization without exceeding the quota. Because there is no cutoff when the limit is reached, a reasonable safety margin must be maintained. Theoretically a control should be able to compensate the error of the previous day, so only the error of the last day of a month cannot be remedied. As the worst case would be the violation of the quota, the last day must be considered lost, at least partially.

4.5.1 Safety Margin

The size of the safety margin is a user defined constant. For this thesis the schedules where calculated with 99 % of the quota. In numbers this allows a misprediction of approximately 30 % for an average day (equation 4.12). Querying the CSN database revealed, that for the current and previous quota (650+ samples) the consumption covers, on average, a small range. Using the hypothetical standard day (quota divided by 30 days) as reference, yields on average a range between 76.8 % and 111 % of the day. The maximum difference is at 141.4 %, but only 7 samples (1 %) where even above the 30 %.

$$
\begin{aligned}
\frac{1}{30} &= \frac{10}{300} \\
\frac{10}{300} \cdot 30\,\% &= \frac{13}{300} \\
\frac{13}{300} - \frac{10}{300} &= \frac{3}{300} \\
&= \frac{1}{100} = 1\,\%
\end{aligned}
\tag{4.12}
$$

The safety margin covers almost the whole observed range, which would cover a complete mispredict in the average case. Nevertheless, it is a compromise between perfect safety and wasting traffic.

4.5.2 Bandwidth Factors

The second part of the algorithm is the planning of the corresponding bandwidth factors. While the traffic forecast has an optimistic outlook with its narrow safety margin, the planning of the bandwidth factor is more conservative, or even pessimistic. Because of changes in the user base, the general incline, release dates of popular software and most certainly many other causes, the traffic consumed by the users on a given day and the corresponding factor are a one time occurrence. However, the statistical data can render a general idea, which can be used as a starting point for the control.

In the past years, the factor rose to the maximum, with the exception of 2005, during the Christmas holiday due to the reduced consumption. When the students returned at the beginning of January, the sudden spike sent the factor down, close to the minimum, only to rise again at the end of January. Other months, such as October, may exhibit similar patterns, albeit not as dramatic. Generally speaking, it should be better to start out with an already lowered factor, hoping it will result in less variability of bandwidth factor. Using the average factor as a starting point was the first idea, but as explained in chapter 3 section 3.3, a factor below one has more weight than above. Therefore the median of the factors was used in the second and third test, because it was lower than the average and has two desirable properties. Firstly, it is not falsified by a few days with high factors, outlier values in a statistical sense. Secondly, it splits the factors in the middle, one half above, the other half below. This seamed the better way to find a factor, which should theoretically be able to yield the same total consumption while remaining constant. Ideally, this would eliminate the need for a control. Practically, the control will not change the factor if the consumption matches the predicted value. However, it became apparent in those tests, that the choice was too conservative, pushing the consumption to a much lower level than anticipated or desired. Thus the forecast was returned to using the average factor.

After determining the average or median of the previous factors, it is scaled with the same constant c as the traffic in section 4.2. It is disputable, whether this is necessary. A positive scaling stems from an underutilized month or an increased quota. On the other hand, the continuous growth of the demand for traffic might be enough to completely consume the formerly unused portion and even a small raise. The tests showed that when using the median, the scaling was insufficient and even lowered the actual consumption far below the predicted values, whereas there was no such effect when in October which used the average.

The final part of the forecast/schedule algorithm was to preload the factor for each day. The idea was to reduce the tension on days scheduled for a less than average consumption and to heighten it on those above. For each day, the ratio between the traffic of the standard day and the scheduled traffic is multiplied with the scaled median/average factor (equation 4.13).

$$t_d = \frac{t_{limit}}{d} \qquad f_i = f_{avg} \cdot \frac{t_d}{t_i}\ (i = 1, \ldots, d) \tag{4.13}$$

i	-	current day
t_d	-	allowed daily consumption
t_{limit}	-	monthly quota
d	-	days in month
f_i	-	scheduled factor
f_{avg}	-	average or median factor
t_i	-	scheduled traffic

This had been introduced to further minimize active control, but was dropped after the tests as the preload was to small and of no consequence to warrant the effort.

4.5.3 Conclusion

Traditional forecast algorithms are at best, if the goal is to estimate the most likely value of a variable in an unknown situation. Time series analysis deduct future outcomes from past values. Other methods try to identify underlying factors which exert an influence. However, they are unable to include the fact, that they can manipulate factors themselves and thus affect the outcome dramatically. Among other reasons this was given as proof to rule out the established forecasting techniques. The result is mathematically simple, but fits the requirements of the scenario. After all, the purpose of the schedule is to aid the control, to guide it, by estimating a likely amount of consumption each day and thereby defining a general pattern. In the best case, the schedule is a forecast, minimizing the work of the control. The average case will provide the control with a reasonable starting point, after which the obligation to remain within the quota is turned over.

5 Control Algorithm

The key elements of the control mechanism are discussed in this chapter. At first, the goals, which the algorithm should try to reach, and a general idea of the algorithm are presented. In the following sections, the algorithm is described in full detail.

5.1 Goals of Traffic Control

The primary object of the traffic control is to keep the total traffic below the quota. Beyond that, the algorithm, read the CSN, has the liberty to distribute the quota as it sees fit. Nevertheless, there are secondary objectives, which require proactive measures by either a human or an intelligent algorithm. Among these are:

- penalize power users
- fairness (e.g. no back doors or other means of circumvention)
- maximize quota utilization
- minimize control

The first two issues are addressed by a proper accounting and classification method, which is described in section 3.2. The other ones depend on an implementation of a global control, a macro management.

In theory, a perfect forecast would yield an optimal input for the traffic control. But like every forecast, it cannot account for disturbances and unscheduled events as trivial as a power outage. The advantage of the forecast is the schedule property, that the quota is not endangered if the consumption matches the predicted value. The basic idea of the control algorithm is to try to match the scheduled values as close as possible by adjusting the bandwidth factor appropriately. Besides fine-tuning the "speed knob", the control has to update the forecast to make up for the errors encountered. This chapter discusses the key elements of the different control paths, as well as the reasons why they were chosen, discarded or retained.

5.2 Control Methods

In chapter 4, it was concluded, that any kind of forecast, schedule or prediction can be no more than an educated guess. It serves as a hint for the control, to tell whether it is on or off track. For this purpose, the ratio between the desired output and the actual output is used to adjust the factor, while the error, the difference between these two values, is first divided by a number of days and then evenly distributed among them. Control theory calls this a closed-loop controller, because it uses feedback to control the output of the system. Its setpoint is the scheduled consumption, which changes daily. After defining a control law, the system adjusts itself to follow the setpoint. The methods discussed in the following sections deal with two possible approaches. In the first case, the control is affected by updating the schedule in a particular style. The other path defines a control law, which affects how the error and error ratio are used to alter the bandwidth factor directly.

For all methods discussed in this chapter, the (global) control is run once after a day is completed, that is, after all accounted data has been inserted into the database and all statistical tables have been populated with the new data. The global Evaluator (update agent) is then run within an hour, at the next scheduled invocation as part of the user-level Evaluator, which re-assigns the users to their appropriate groups ensuring the groups remain sorted.

5.2.1 Relative Proportional Control

At the beginning of this thesis, the author was still convinced, that given a sufficiently good forecast, the control needed only minor adjustments to its initial factors. The hypothesis stemmed from the observation, that the traffic pattern repeated itself every year with minor deviations including the omnipresent increase in consumption.

Any errors encountered are attributed to the deviations, which would eventually be canceled out by the errors of other days. On which day, the error would be remedied is unknown, as is could be on any subsequent day. By this policy, the traffic schedule is updated as in equation 5.1. At first the absolute error is calculated by subtracting the predicted traffic from the real consumption. The next step is to distribute the error on the remaining days in order to maintain a correct solution. The algorithm divides the error by the number of days remaining and deducts the portion from each day left in the current month. In case of negative errors from underutilization, the subtraction is actually an addition.

$$\begin{aligned} e &= R_{i-1} - P_{i-1} \\ e_d &= \frac{e}{d_{left}} \\ N_i &= P_i - e_d \ (i = 1, \ldots, d) \end{aligned} \qquad \begin{array}{lcl} i & - & \text{current day} \\ e & - & \text{error} \\ R & - & \text{actual traffic} \\ P & - & \text{predicted traffic} \\ e_d & - & \text{error per day} \\ d_{left} & - & \text{days left in current month} \\ N & - & \text{new scheduled traffic} \\ d & - & \text{days in month} \end{array} \tag{5.1}$$

As the scheduled consumption has changed, the factors should be adjusted accordingly. Again based on the assumption, that, at the end of the day, the errors will cancel one another out, the adjustment of the bandwidth factors was also done very lightly.

$$f_{n_i} = f_{p_i} \cdot \frac{N_i}{P_i} \qquad \begin{array}{lcl} i & - & \text{current day} \\ f_n & - & \text{new factor} \\ f_p & - & \text{previous/current factor} \\ P & - & \text{old predicted traffic} \\ N & - & \text{new scheduled traffic} \end{array} \tag{5.2}$$

The fraction in equation 5.1 results in proportional alterations. The difference between the factors is relative to the scheduled traffic while the deduction remains constant. A day with above average traffic will have its factor reduced less than a day with below average traffic.

Assessment

This control law has been abandoned completely because it has grave errors. On the positive side, the traffic update method offers some robustness. The errors are distributed equally over many days (15.5 on average), this results in relatively small changes on a given day. Even a capital error of 50 % is reduced to a 3 % change per day, given a standard day in both cases. However, the accumulated errors start to manifest themselves during the second half of the month, which leads to a last minute panic to maintain the quota.

The more important issue is the hesitant and slow reaction of the control to discrepancies. The relative multiplier has so little effect, that it would need a 50 % error in the traffic schedule to halve the factor. Any error above one percent is a potential threat to the quota, because one percent excess traffic every day will consume 99.99 % of the quota. The control lacks a lever to enhance the effect on the factor even for small errors.

Finally, the relative factor adjustment is counterproductive. High traffic days should receive a greater or at least the same reduction as low traffic days due to the higher savings potential. But the resulting fraction after the error deduction is the smaller, the higher the scheduled traffic is. The usefulness of this "control" is nonexistent, as it has only little more effect than no control at all. However, there is a remote applicability if the forecast is either perfect or exaggerated.

5.2.2 Error-driven Control With Penalty

In order to improve the effectiveness the first measure was to reduce the update interval of the traffic schedule. As the time series analysis in chapter 4 revealed, a week represents a full cycle. Therefore the interval was cut to the next seven days or, if less, the remaining days of the month.

$$
\begin{aligned}
e &= R_{i-1} - P_{i-1} \\
e_d &= \frac{e}{7 \parallel d_{left}} \\
N_i &= P_i - e_d \; (i = 1, \ldots, 7 \parallel d)
\end{aligned}
\tag{5.3}
$$

i	-	current day
e	-	error
R	-	actual traffic
P	-	predicted traffic
e_d	-	error per day
d_{left}	-	days left in current month
N	-	new scheduled traffic

This has two significant implications. First, it leads to greater changes in the scheduled traffic values, which, given the previous method, would provide some more leverage on the factors. Second, it delivers a more immediate feedback, because the discrepancies are no longer hidden until the last third of the month. Thus the control gains knowledge about the effectiveness of its measures on timely basis. However, the more substantial the changes, the more active the control must be. The forecast precalculates the factor for a given consumption, which in turn has to be constantly readjusted to match the updated schedule. As it is unfeasible to determine, which factor will correspond to a certain consumption (confer section 3.3), the error (ratio) represents the only means to develop a control. According to control theory this is also the right way for a closed-loop controller. Since the absolute error is irrelevant and unusable for control purpose as the errors relate to the size of the quota, the ratio between the actual and the scheduled traffic is used (equation 5.4).

$$u = \frac{R_{i-1}}{P_{i-1}} \qquad \begin{array}{lcl} i & - & \text{current day} \\ u & - & \text{utilization} \\ R & - & \text{actual traffic} \\ P & - & \text{predicted traffic} \end{array} \tag{5.4}$$

After calculating the utilization u, *all* days are divided by it. All days, because an error means there is a misprediction, which invalidates the supposed appropriateness of the scheduled factors f_i. The reasons for unfitting factors include improper group bandwidths not matching the quota, unraised quota with concurrent high demand, too heavily raised quota, moveable holidays (Easter) or a simple network outage. In any case the factors have to be reset to a value, which keeps the actual consumption close to the predicted one. Ideally the ratio would remain at one, once the initial error is mitigated and the factor has settled at its new value. Unfortunately the utilization ratio suffers from the same problem as the previous method. It lacks sufficient leverage on the factors for its expected range (0.6 to 1.4). Therefore the factor is enlarged by a power law, which linearizes the geometric sequence of the bandwidth of the groups (figure 5.1).

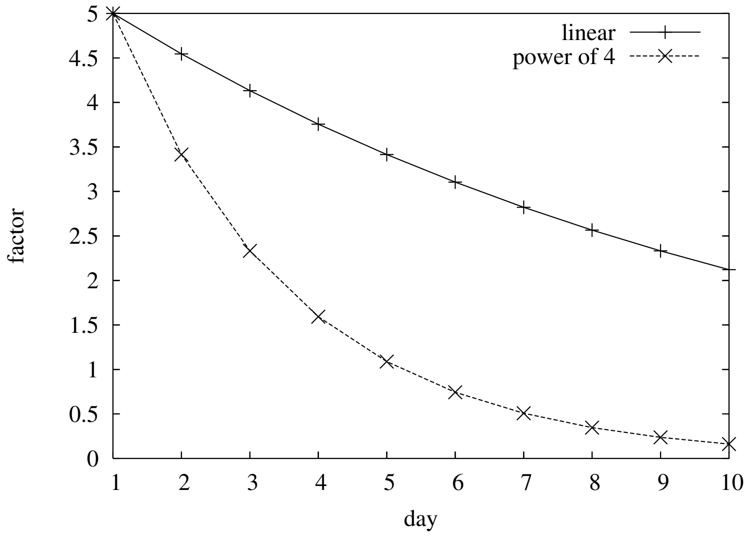

Figure 5.1: Comparison - Factor Reduction at 10% Error Rate

To prevent oscillation of the bandwidth factor, a threshold is added before the power law comes into effect. This also means, that the control accepts errors to a certain degree without losing control, which still progresses linearly. The threshold itself was empirically set to 1.1, which means that it takes effect, if the actual consumption is over 110 % of the scheduled one. The power of four was chosen, as it has proven to be effective in the former average-based control.

$$f_n = \frac{f_o}{u} \quad \text{if } u < 1.1 \qquad\qquad f_n \ \text{-} \ \text{new factor}$$
$$\qquad\qquad f_o \ \text{-} \ \text{old/current factor} \tag{5.5}$$
$$f_n = \frac{f_o}{u^4} \quad \text{if } u > 1.1 \qquad\qquad u \ \text{-} \ \text{utilization}$$

Assessment

After the disastrous results of the relative control (confer section 6.2), the error-driven control lived up to its expectations. The linear element adjusts the factor sufficiently, allowing the control to level the expected digressions from the schedule. In return, the threshold enabled power law provides enough force for dealing with any unexpected events.

The downside of this method is overshooting if behind schedule. It was faulty reasoning to assume that the factor would settle at the equilibrium position after the backlog had been cleared. The impulse to clear any backlog will raise the factor (to maximum) as long as no negative error is encountered. During this phase the bandwidth first rises slowly then exponentially, which in turn causes the backlog to clear suddenly. However, the bandwidth factor remains at this level and will most likely result in excessive consumption on the following days. After dwindling the factor with the power law, a vicious cycle ensues.

5.2.3 Dampened Error-driven Control with Penalty

The final control strategy summarizes on all previous issues and their solutions. It also addresses the final problem discovered in the previous section. The overshoot of the plain error-driven control must be reduced to prevent uncontrolled consumption, which might endanger the quota or cause a panic cutoff. The problem pertains to situations, when the control clears the backlog. The factor is linearly increased to allow more traffic, which should lead to a soft clearing. The change in consumption is not instantaneous. While the factor rises, more backlog can be accumulated until the factor is finally high enough to turn the tide. The problem at hand is to detect when the turning happens. This is similar to the average-based consumption, which must react quickly to short term increases in consumption by lowering the bandwidth to prevent greater damage. Its indication is the ratio between the seven day moving average and the fourteen day moving average (equation 5.6). As long as there is no day above the allowed daily consumption, it is not used. The indicator itself possesses the desired property of signalling the control, when the consumption changes. The trigger, on the other hand, causes the third problem of the average-based control, which is discussed in section 3.4.3. The desired behavior can be achieved after modifying the trigger to only fire when there is sufficient evidence to support it.

$$r = \frac{t_{avg_7}}{t_{avg_{14}}} \qquad \begin{array}{lcl} r & - & \text{rise} \\ t_{avg_7} & - & \text{7 day moving average} \\ t_{avg_{14}} & - & \text{14 day moving average} \end{array} \tag{5.6}$$

If the ratio between the two averages is above one, there has been a higher traffic consumption in the last seven days than in the last fourteen days. If the traffic is higher than expected in the schedule, the rules are applied as stated above (equations 5.4 and 5.5). A utilization less than 100 % creates a backlog and the factor rises. Given a large backlog, the factor increase should not be stopped until the utilization is close to 100 %, in which case the backlog is about to be cleared. The overshoot is only dangerous if the consumption is above average, thus the rise ratio would only be applied if the seven day average is above the allowed daily value. Should those constraints be met, the rise ratio is raised to the power of four and is multiplied with the utilization. The result is used as the divisor in the factor update calculation (equation 5.7).

$$f_n = \frac{f_o}{u \cdot r^4} \quad \text{if } r > 1.01 \text{ and } t_{avg_7} > t_d \qquad \begin{array}{lcl} f_n & - & \text{new factor} \\ f_o & - & \text{old/current factor} \\ u & - & \text{utilization} \\ r & - & \text{rise} \\ t_{avg_7} & - & \text{7 day moving average} \end{array} \tag{5.7}$$

If the utilization is low, the adjustment is only dampened slightly. At some point, both values cancel each other out, resulting in no adjustment at all. Figure 5.2 displays the product for various input values.

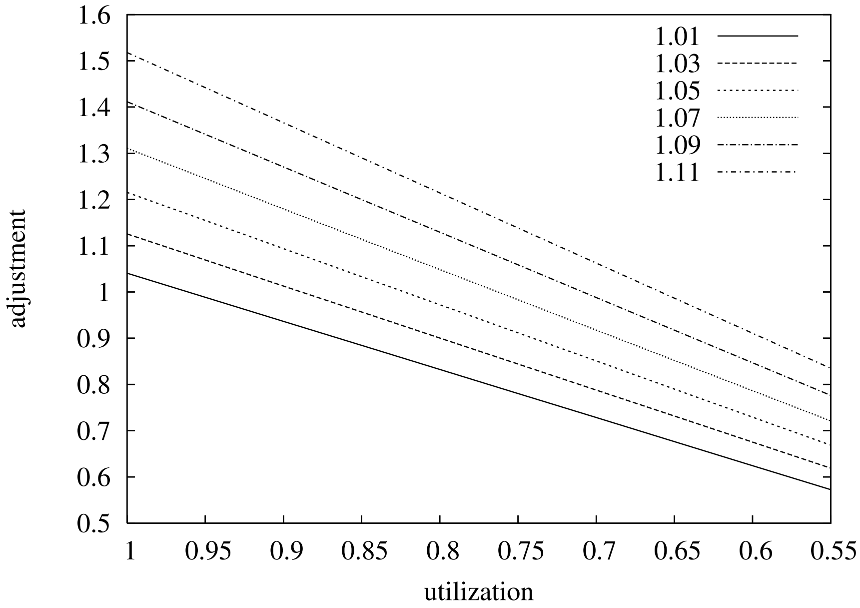

Figure 5.2: Adjustment at Different Rise Ratios

This allows the seven day average to climb above the allowed daily value and remain at this level, while assuring that once the backlog is about to be cleared, the controller is stopping the increases and regaining control.

Assessment

Besides the modified behavior in case of underutilization, the other rules remain unchanged. The traffic error is equally distributed over the next seven days (or the days left in month) and the factors are adjusted according to equation 5.5 if the consumption is higher than scheduled. The control reacts faster to those errors exceeding the 110 % threshold, which marks the end of the deadband. It is also more resistant to the creeping quota violation (section 3.4.1). Because the controller tracks the consumption, the overutilization would at first reduce the bandwidth slowly. After the continued overspending is apparent in the schedule, the utilization will inevitably exceed the threshold and the factor will receive the needed cutoff. On the other end, the controller no longer waits for the backlog to clear before returning to normal levels. Once the upward motion has gained sufficient momentum, the dampening comes into effect. First reducing the adjustment, later reversing its direction before the backlog is cleared.

5.3 Conclusion

In the conclusion of the previous chapter, the schedule was described as an educated guess, a guideline for the control. Undoubtedly there are plenty of reasons and scenarios which will invalidate a forecast. Control theory offers many models and strategies to meet the demands of almost any controllable system. When it comes to humans, any discovered pattern or behavior may become wrong. Fortunately this is rarely the case. The large number of people living in the same environment exhibits an almost predictable behavior. A fact that is exploited by the schedule and the control mechanism. The job of the control is to deal with the "almost" and the other unexpected events. The first strategy presented was very inflexible, it relied heavily on the correctness of the forecast. Any substantial change requires very strong impulses, which are not readily available.

Accepting the possibility of scheduling errors demanded a more agile control. A control that amplifies the relatively small errors to provide an immediate effect. This fast reaction is mandatory because the schedule only has a 1 % margin for errors. The overspending is not the only problem, which had to be faced. In spite of all efforts to perfect the forecast, the factors retain their uncertainty. Depending on whether the quota has changed or not and to what degree, the estimates can be completely off. In case the factors are too low, a backlog will accumulate until the controller has adjusted the factor. But as the backlog is hidden within the schedule, the adjustment is not sufficient. The higher traffic values force the factor to rise until the consumption matches these setpoints. Upon reaching this point or even earlier, the factor had to be lowered again because the elevated consumption was only required to clear the backlog. A high factor might be harmful for the upcoming days, which have not received additional traffic and may be at a much lower level.

The control system as a whole was initially considered redundant. It may as well be, if there was a way to foretell the future. Therefore the controller was restructured to be more robust to unexpected events and more agile in its reactions. The scenarios, which fostered these modifications, are presented in the following and final chapter.

6 Test Cases and Analysis of Results

This chapter examines the results of the researched scheduling and control algorithms in a real world environment. In order to verify the effectiveness, the algorithms were tested in an actual network during a three month period. Three tests were conducted, each one with the objective to control the quota of a month. At the end of each month the performance of the algorithm was evaluated. If it yielded poor results, the software was changed either at the beginning of a new test period or in case of apparent errors. Tests had to be delayed until October when the new semester started. The test cases were as follows:

I. The forecast uses linear scaling and the control operates with the relative proportional method.

II. In the second case, the forecast was altered to compensation scaling. It also introduces the completely altered error-driven control, which penalizes excessive consumption.

III. The final setup combines both previous scaling techniques in a hybrid scheduling method. The control strategy of the previous case is enhanced by a predictive element to preemptively lower the bandwidth in order to dampen the overshoot.

The first setup was used to test the new approach, while the second one is an engineered control based on the results of the first case. After optimizing the control, the software is tested in the December scenario, which is one of the problem cases of the current average-based control (3.4.2).

6.1 Test Environment

All tests were conducted in a real network with real people to receive true and immediate feedback on the performance of the algorithm. A simulation was not used as network simulators do not include the human element. The network of the CSN was used as proving ground for the software because it only required the new algorithms, while everything else (accounting, database, network, internet connectivity) was already set up and available. The CSN is a student operated network in the resident halls of the Studentenwerk Chemnitz-Zwickau at the Chemnitz University of Technology. There are approximately 1900 students connected to a modern network with Twisted Pair (TP) and 10 or 100 Mbit ports. The network infrastructure (figure 6.1) is a classic star shape with a single uplink to the university's campus network and through that to the Internet.

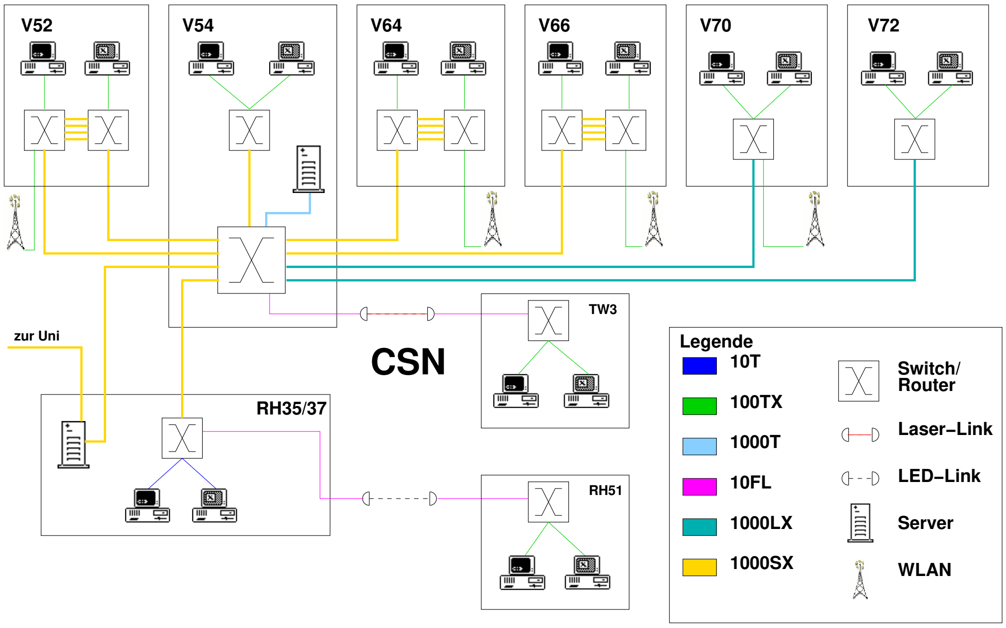

Figure 6.1: CSN Network Infrastructure (October 2006)

Besides wired access, the students may use a wireless LAN connection where available. All traffic is accounted by the core router and send to a dedicated machine for user-based processing. The accounting data is transferred to another machine and inserted into a PostgreSQL database, which provides the foundation for all shaping decisions. The periodically running DynShaper software uses this data to reassign users to their proper groups and changes the shaping rules to reflect these changes. The shaping rules are created on the same machine, which is also the only uplink and therefore able to control all incoming traffic. The software may also control outbound traffic if desired, although the accounting and database queries must be altered to provide this data. A more detailed description of the infrastructed and the associated processes can be found in [31] and [26].

6.2 Test Case I: Linear Scaling and Relative Proportional Control

The first test in October reflects the assumption that a good forecast would suffice and thus only require a minimum of control.

6.2.1 Objectives

The primary goal of the first test case was to gain experience. Using a simple scaling technique limited the complexity of the forecast and allowed to focus on whether the traffic pattern is repeated or not. It was expected, that the initial conjectures were faulty in some degree. To determine the extend of these errors in order to improve the control was the second objective.

6.2.2 Parameters

Except for a 1 % safety margin, the statistical data was scaled linearly according to the algorithm described in section 4.2 to match the current quota. The resulting schedule in figure 6.2) shows that every day receives the same contribution.

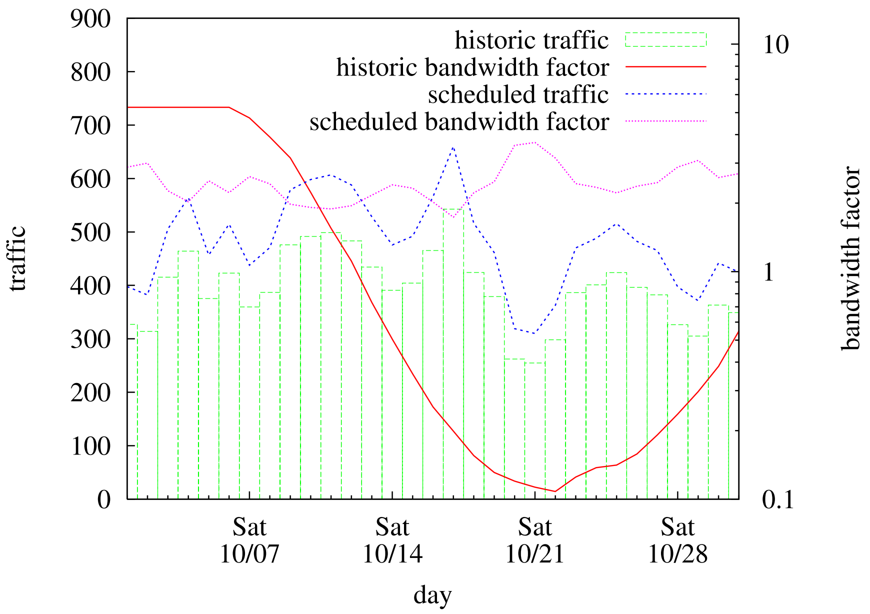

Figure 6.2: October Schedule

The factors are based on the average factor from the statistical data and preloaded according to equation 4.13. The controller operates with the relative method explained in section 5.2.1. Traffic schedule and factors are updated for the complete remaining month.

6.2.3 Results

The linear scaling was derived from the observation, that the demand for traffic increased by 30 % and more every year. An ample quota raise should therefore suffice to render the same results at approximately the same average shaping factor. The graph in figure 6.3 shows that this thesis is correct for the greater part of the month as the scheduled line remains close to the boxes displaying the actual traffic consumption. However, in the last third of the month, after the 18th, the errors began to increase up to a point, at which a correction was no longer possible.

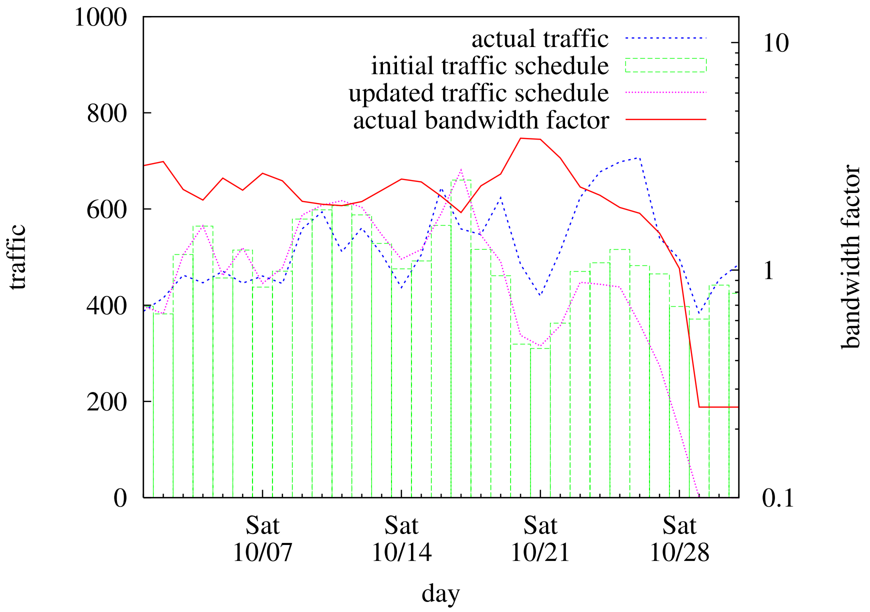

Figure 6.3: October Results

The test had to be aborted on October 29th after the quota was reached. For the remaining days the bandwidth factor was set to a reasonable but small value. In the statistical data from the previous year the factor had reached its lowest point approximately at the same time the errors started in the test case. Apparently the factor had attained a level at which the number of users with reduced bandwidth was sufficient enough to lower the consumption significantly. Although the factor recovered soon after, the traffic did not return to the same heights. It is clearly visible, that without accounting for the bandwidth factor, the forecast cannot provide good estimates. It also rules out any time series analysis, which is absolutely unable to deal with the fact that the analyzed data can be controlled, modified and shaped against any discovered trend. But even if the forecast is mistaken, the control has to be in a position to counteract those errors. An ability that the relative proportional control lacks, which was used in this test case.

Disregarding the errors, the basic assumptions could be verified. The actual consumption matched the predicted one very close in the first two thirds of the month. It is doubtable that this is the result of the shaping. The bandwidth reduction had been so minimal (factor two doubles the initial bandwidth), that the only affected groups where those for high volume users (9 and 10). Nevertheless, the preloading seemed to have the desired effect in the week of October as all days yielded approximately the same consumption. But the leveling did not continue on the following days, which posed the question if it had only been a coincidence. Thus the preloading had to be reexamined during the next test in November. In consequence of the failure to provide a good forecast and a matching control, both algorithms were substantially altered.

- Utilization of 99 % quota: 108.2 %
- Utilization of full quota: 107 %

6.3 Test Case II: Compensation Scaling and Error-driven Control

The results of the first test were promising, but required improvement. While the linear forecast technique is surprisingly accurate as long as the bandwidth is not seriously affected, the traffic schedule should be more concerned with reducing the lows due to low bandwidth before increasing the highs.

6.3.1 Objectives

After the previously used control method failed to react in due time to the continued overutilization, which caused the violation of the quota, the reaction time of the new control was under scrutiny. Since it was the forecast that put the controller in such a position, the performance of the new forecast algorithm is the second issue, which had to be observed.

6.3.2 Parameters

The statistical data is scaled to 99 % of the current quota, using the additional traffic to normalize all days which had a less than average bandwidth factor. Figure 6.4 displays the schedule derived by using the algorithm described in section 4.3.

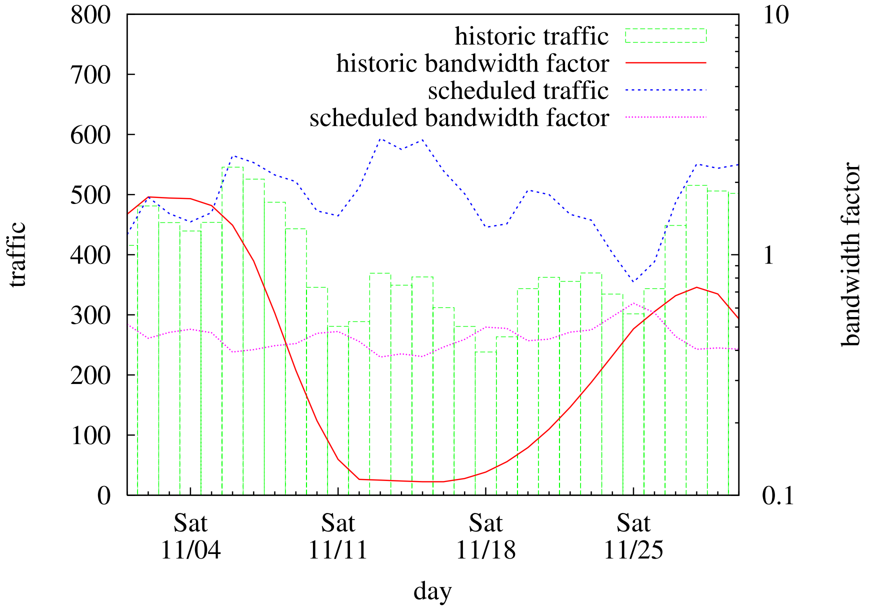

Figure 6.4: November Schedule

Trying a more conservative approach, the initial factors are derived from the median of the former factors. As for the controller, the error-driven method was first used only in case of overutilization, while the underutilization was handled by the relative proportional paradigma. After November 9th, the error-based control was used for both cases with the exception of the penalty rule, which was only applied if the actual consumption exceeded the schedule. The update of the traffic schedule was shortened to the next seven days in order to gain a more immediate feedback on the efforts of the controller, while the factors were adjusted for all remaining days. If a factor caused an error on a given day, it failed to meet its purpose. To mitigate the likelihood of recurrence, all factors had therefore to be altered accordingly.

6.3.3 Results

The failure to remain below the quota in the first test, demanded a more cautious approach, sensitive to errors, which could lead again to the worst of all possible failures. An unpredicted overutilization of 134 % on the first day of November lowered the factors for the following days drastically. However, the following days did not exhibit the same behavior. By November 9th, the schedule and the actual consumption had diverged substantially, which called for an immediate change in the controller in case of underutilization.

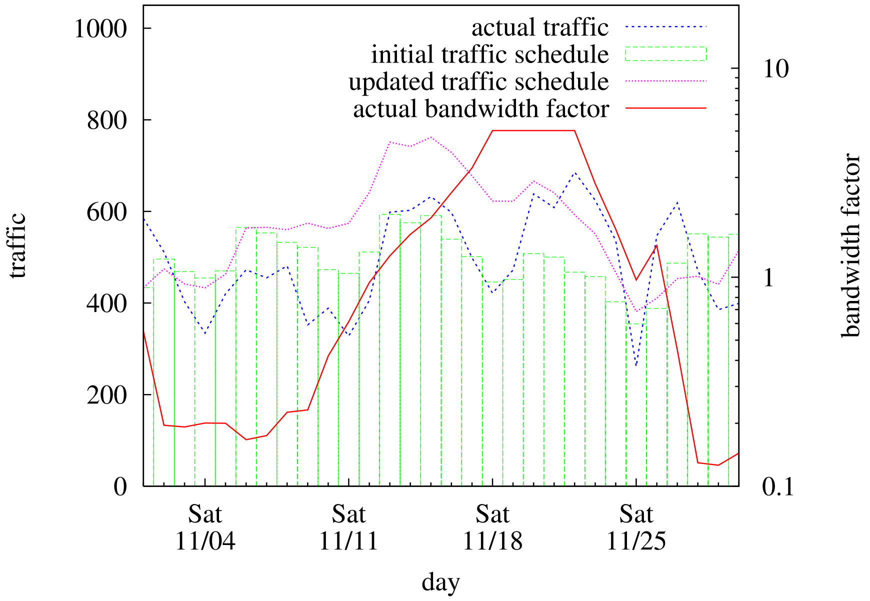

Figure 6.5: November Results

After changing the controller to use the error in both cases, the factors increased in the following days to clear the accumulated backlog and to bring the consumption to the level of the schedule. In figure 6.5 the graph shows that actual consumption returned to the initially scheduled values on the November 13th. However, as the updated schedule accounts for all past errors, its scheduled values remained above the actual traffic forcing the factors further up. After five days of unrestricted consumption, the traffic on the 22nd exceeded the schedule once again. The inability to detect the turning point and stop the factor increases, caused the controller to enforce a fast descent, which was only interrupted temporarily by a network outage on Saturday the 25th. In spite of the overshoot, the reduction was sufficient to avert the imminent danger to the quota. At the end the yield was the highest since the introduction of the DynShaper software in 2001.

- Utilization of 99 % quota: 99.2 %
- Utilization of full quota: 98.2 %

While imbalance of the control has shifted the traffic pattern, it still matches its general form. The method of scheduling is therefore not responsible for the discrepancies observed. There were issues, raised in section 4.3, which had to be addressed in order to use the forecast method in all conceivable future scenarios. The second problem is the upswing of the bandwidth factor, which did not cease until the consumption finally exceeded the schedule. Neither should it stop upon reaching the initial forecast value. A minor overshoot must be allowed to clear the backlog and return the schedule to its precalculated level.

6.4 Test Case III: Christmas Scenario

The displayed ability to control the traffic on short notice allowed to shift the focus on the improvement of the controller. Although the algorithm was functional, it lacked elegance and foresight, especially in the face of a schedule with forecast quality.

6.4.1 Objectives

While the first two tests were merely used to evaluate the quality of the forecast and the performance of the controller, the final one was the trial by fire. In the Christmas scenario the majority of any unused traffic is shifted to the first part of the month. This is potentially dangerous if the consumption during the Holidays is higher than expected. At this a comparatively low traffic level, the influence of the DynShaper is negligible because there are too few users to achieve the same effect as on regular business days. The users in group ten might not even be in the dormitory, but their moving average is still so high, that they are the first ones to be shaped with no effect at all. So this month relies heavily on a precise forecast-like schedule and an agile control.

6.4.2 Parameters

The schedule is calculated using the hybrid forecast method (section 4.4). Figure 6.6 shows how the algorithm allows more traffic in the first weeks than at the end of the month, but still raises all days to account for the continuously increasing demand.

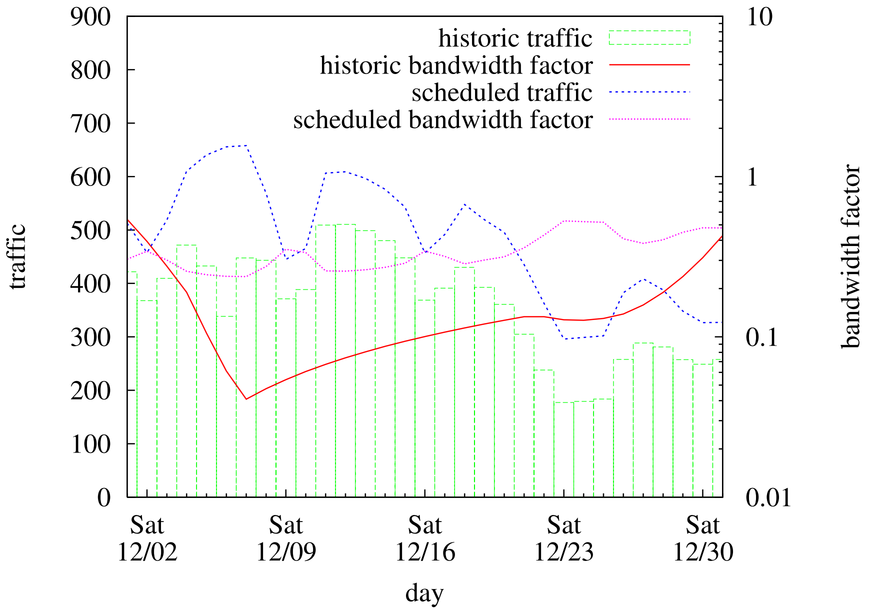

Figure 6.6: December Schedule

To verify that the median is indeed the false choice, the factors are again based on this value. The changes to the controller were only minor. Its algorithm now included the turning point detection and the dampening as described in 5.2.3. The second constraint, which requires the seven day moving average to exceed the allowed daily value, had not been added, yet. Thus any increased consumption will cause the trigger to fire and stop the factor from increasing further. As long as the backlog is small and the traffic is already close to the allowed maximum, this can be appropriate. The remaining parts are unaltered. Errors are propagated into the traffic schedule of the following seven days, while the factors are adjusted on all remaining days.

6.4.3 Results

Fulfilling all expectations, the results of the final test are almost exemplary. Figure 6.7 displays the exceptional accuracy of the forecast. Starting with too small factors, the consumption remains below its setpoint until the controller can adjust. The repeated underutilization as the result of using median as base for the factors reinforces the decision to return to the average as starting point for the control.

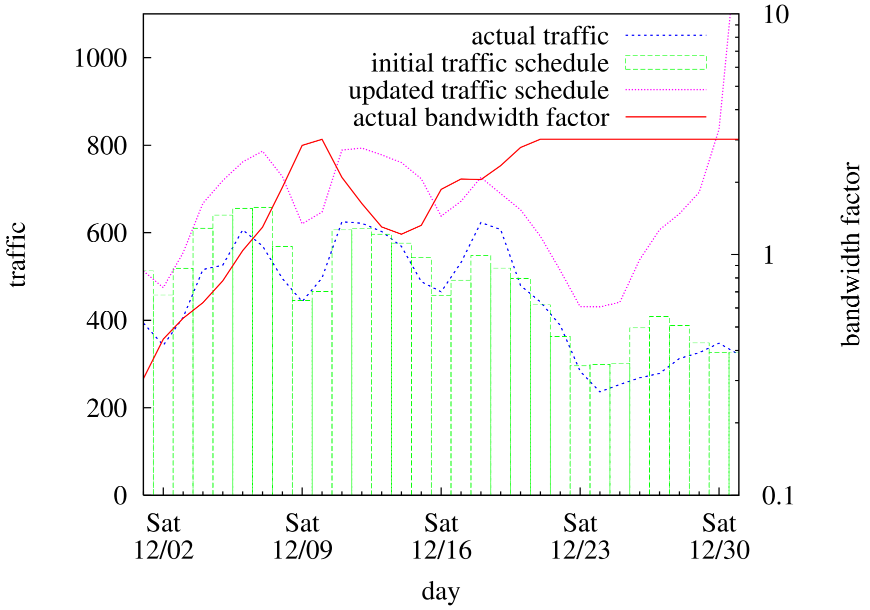

Figure 6.7: December Results

Having raised the factor to its maximum, the control's dampening feature is triggered on December 10th and in effect until December 18th. Being initially effective, its influence is diminished after the 14th. The moving average has shifted and the rise ratio is dropping, while underutilization remains unchanged, thus reversing the trend. However, the re-raise of the factor could no longer increase the consumption substantially to clear the accumulated backlog. As the factor was already at its maximum, there was nothing left to be done, when the consumption did not even match the initial schedule in the week after Christmas Eve.

In a more positive perspective, the actual influence of the shaping had been insignificant after December 9th. The quota, which had been in effect for all three tests, produces such high initial bandwidths, that at factor one only two groups are subject to shaping. At factor two only one group is still shaped, but at doubled bandwidth. So admittedly, the consumption matched the schedule for two-thirds of the month without the aid of shaping. Taking that into account, it might have been possible to run the December without any shaping at all as the scheduling had produced an exceptionally precise forecast. After all, the final yield was higher than any December before.

- Utilization of 99 % quota: 93.4 %
- Utilization of full quota: 92.4 %

It is very likely, that it would have been above 95 %, if the average had been used for the factors instead of the median.

7 Conclusion and Further Work

This thesis developed an improved control algorithm for managing a fixed non-deferrable traffic quota in shared networks. The key features include:

- predictive scheduling of traffic consumption within the bounds of a quota (based on statistical data)
- anticipatory bandwidth adjustment to minimize control effort (based on statistical data)
- adaptive feedback controller
- lossless re-scheduling
- improved utilization in standard and non-standard scenarios

Before improving utilization, the new control model had to match the good utilization provided by the average-based algorithm. The first attempt resulted in a disaster as the quota was exceeded. The two following tests produced excellent yields, which set new records. Previously, the maximum utilization had been between 93 % and 97 % in a shaping controlled standard month (May, June, July and November). The second test in November resulted in a 98 % utilization (99 % without the safety margin). Continued production use in the following months should render similar results.

Statistical analysis as part of the case-based reasoning process revealed repeated traffic patterns, which can be exploited to create a predictive schedule of the future consumption. The scheduling algorithm maximizes the estimate as far as the quota permits. By using a schedule, the utilization of the problematic December scenario could be raised to above 90 %, thus effectively eliminating the problem. The preliminary bandwidth schedule reduces the time the controller needs to reach its setpoint. It provides a head start in cases, where the consumption is expected to increase dramatically within a short period of time, especially across the monthly boundaries.

The use of this work is not limited to the CSN. The DynShaper software can be used in almost any shared network wishing to distributed its quota on a fair basis to its users. However, it requires a continuously operating server with a working accounting solution, database and firewall. In addition to the hardware, the user base has to be sufficiently large (> 100) for the shaping to operate as intended. Having a smaller number of users can lead to imbalanced shaping groups, where the amount of traffic consumed by each group is not equal, especially in those for power users. Finally the effort to set up the required infrastructure has to be weighted against the benefits gained. Thus, people sharing a flat with a DSL internet connection should rather invest in a flat rate than in understanding and deploying DynShaper.

7.1 Further Work

This thesis draws on the research of various field, such as control theory, artificial intelligence, statistics, and quality of service. Their combination in this work provides various starting points for further work. However, the complexity of the software already demands a seasoned system administrator to deploy and maintain. Further works should therefore consider the practical applicability first.

7.1.1 Control Theory

The controller developed in this thesis uses plain feedback and leaves the prediction mostly to the scheduling algorithm. Using the schedule and the encountered errors, the controller could calculate the necessary adjustment to match the schedule instantly. But because the system is time-variant, the controller would also require specific knowledge about the weekday and the month to differentiate between natural and artificial influences.

7.1.2 Artificial Intelligence

The case-based reasoning approach re-uses the data from one month to form a new solution. As the most recent case, it has the highest significance. If there is more data available, the algorithm could either generalize all of them and derive a new solution, or it could compute multiple separate solutions, which are weighted and combined to a final solution.

7.1.3 Quality of Service

Although no new algorithms have been published in the last years, the shaping solution could be extended to a more service oriented approach using a different algorithm such as Hierarchical

Fair Service Curve (HSFC) [32]. Currently, all applications are treated equal. With the emerging use of voice over IP instead of conventional phone lines, guaranteed bandwidth and low delay becomes an issue. The only algorithm in the Linux kernel, which can provide both is HSFC.

Bibliography

[1] Jan Horbach and Markus Schade: *Dynamic Traffic Shaper*, 2001-2007.
URL `http://rnvs.informatik.tu-chemnitz.de/twiki/bin/view/Dynshaper`

[2] Oregon State University Residence Halls: *Bandwidth, Security and Architecture*, 2006.
URL `http://oregonstate.edu/resnet/guides/security_architecture.php`

[3] WH-Netz Verein für Technologietransfer und Sicherheit e.V.: *Traffic Quota*, 2006.
URL `http://www.wh-netz.de/knowledgebase/TrafficQuota.de.htm`

[4] Packeteer Inc.: *PacketShaper*, 2006.
URL `http://www.packeteer.com/products/packetshaper/`

[5] Marina Fomenkov, Ken Keys, David Moore and K. Claffy: *Longitudinal study of Internet traffic in 1998-2003*.
URL `citeseer.ist.psu.edu/fomenkov03longitudinal.html`

[6] Jon Postel: *Transmission Control Protocol - DARPA Internet Program Protocol Specification (RFC793)*, DARPA, Sep. 1981.

[7] Jacobson, Braden and Borman: *TCP Extensions for High Performance (RFC1323)*, IETF, May 1992.

[8] Jon Postel: *Internet Protocol - DARPA Internet Program Protocol Specification (RFC791)*, DARPA, Sep. 1981.

[9] Jon Postel: *The TCP Maximum Segment Size and Related Topics (RFC879)*, IETF, Nov. 1983.

[10] V. Jacobson: *Congestion Avoidance and Control*, *ACM Computer Communication Review; Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988*, vol. 18, 4:pp. 314–329, 1988.
URL `citeseer.ist.psu.edu/article/jacobson88congestion.html`

[11] R. Braden: *Requirements for Internet Hosts – Communication Layers (RFC1122)*, IETF, Oct. 1989.

[12] M. Allman, V. Paxson and W. Stevens: *TCP Congestion Control (RFC2581)*, IETF, Apr. 1999.

[13] W. Stevens: *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms (RFC2001)*, IETF, Jan. 1997.

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

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

[16] Bert Hubert *et al.*: *Linux Advanced Routing and Traffic Control HOWTO*, 2004.
URL `http://www.lartc.org/howto/`

[17] Martin Devera and Don Cohen: *HTB Linux queuing discipline manual - user guide*, 2002.
URL `http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm`

[18] Wikipedia: *Controller (Control Theory)*, 2007, lookup date: 2007-01-26.
URL `http://en.wikipedia.org/wiki/Controller_(control_theory)`

[19] Benjamin C. Kuo: *Automatic Control Systems*, Prentice Hall, 6th edn., 1991.

[20] Shankar Sastry and Marc Bodson: *Adaptive Control: Stability, Convergence, and Robustness*, Prentice Hall, 1989-1994.
URL `http://www.ece.utah.edu/~bodson/acscr/`

[21] Wikipedia: *Adaptive Control*, 2007, lookup date: 2007-01-26.
URL `http://en.wikipedia.org/wiki/Adaptive_control`

[22] Wikipedia: *Case-based Reasoning*, 2007, lookup date: 2007-01-26.
URL `http://en.wikipedia.org/wiki/Case-based_reasoning`

[23] Aamodt, Agnar and Enric Plaza: *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches*, *Artificial Intelligence Communications*, vol. 7, 1:pp. 39–52, 1994.
URL `http://www.iiia.csic.es/People/enric/AICom.html`

[24] Roger Schank: *Dynamic memory; a theory of reminding and learning in computers and people*, Cambridge University Press, 1982.

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

[26] Markus Schade: *Dynamisches Bandbreitenmanagement im Chemnitzer StudentenNetz*, 2006.
URL `http://archiv.tu-chemnitz.de/pub/2006/0099/`

[27] Wikipedia: *Zeitreihenanalyse*, 2007, lookup date: 2007-01-26.
URL `http://de.wikipedia.org/wiki/Zeitreihenanalyse`

[28] George Box and G. M. Jenkins: *Time Series Analysis: Forecasting and Control*, Holden-Day, second edn., 1976.

[29] Wikipedia: *Linear Regression*, 2007, lookup date: 2007-01-26.
URL `http://en.wikipedia.org/wiki/Linear_Regression`

[30] Wikibooks: *Mathematik: Statistik - Zeitreihenanalyse*, 2006, lookup date: 2006-08-24.
URL `http://de.wikibooks.org/wiki/Mathematik:_Statistik:_Zeitreihenanalyse`

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

[32] Ion Stoica, Hui Zhang and T. S. Eugene Ng: *A hierarchical fair service curve algorithm for link-sharing, real-time, and priority services*, *IEEE/ACM Transactions on Networking*, vol. 8(2):pp. 185–199, 2000.
URL `citeseer.ist.psu.edu/stoica97hierarchical.html`

List of Figures

List of Tables

List of Abbreviations

AI artificial intelligence

ADSL Asynchronous Digital Subscriber Line

ATM Asynchronous Transfer Mode

CBQ Class Based Queuing

CBR Case-based Reasoning

CSN Chemnitzer StudentenNetz

DFN Deutsches ForschungsNetz

DHCP Dynamic Host Configuration Protocol

DNS Domain Name Service

DoS Denial of Service

DSL Digital Subscriber Line

dt. deutsch

FIFO First In First Out

HTB Hierarchical Token Bucket

HSFC Hierarchical Fair Service Curve

MTU Maximum Transfer Unit

MSS Maximum Segment Size

IP Internet Protocol

ISO International Standards Organization

ISP Internet Service Provider

LAN Local Area Network

qdisc queueing discipline

P2P Peer-to-Peer

PPP Point-to-Point Protocol

PPPoE Point-to-Point Protocol over Ethernet

QoS Quality of Service

SFQ Stochastic Fair Queueing

StuWe Studentenwerk Chemnitz-Zwickau

TBF Token Bucket Filter

TCP Transmission Control Protocol

TP Twisted Pair

UDP User Datagram Protocol

URZ Computing Center

VPN Virtual Private Network

WLAN Wireless LAN

WWW World Wide Web

Appendix

A CD-ROM Contents

The CD-ROM which accompanies this work comprises of a number of sections which are elaborated below.

doc/

This directory includes an electronic copy of the thesis in Adobe PDF and Postscript format.

img/

This directory contains all figures used in this work, both in encapsulated postscript (eps) and Adobe PDF format, as well as the data they were created from.

src/

The source code for the DynShaper software is contained in this directory.

- perl/ - source code distribution (Dynshaper/Evaluator/Predictive.pm features the main part of the add-on)
- sql/ - commands to create the required database entries

Thesis Declaration

I hereby declare that the whole of this diploma thesis is my own work, except where explicitly stated otherwise in the text or in the bibliography.

This work is submitted to Chemnitz University of Technology as a requirement for being awarded a diploma degree in Computer Science ("Diplom-Informatik"). I declare that it has not been submitted in whole, or in part, for any other degree.

Chemnitz, January 30, 2007

Markus Schade