Helsinki University of Technology
Department of Electrical and Communications Engineering
Networking Laboratory

Eeva Nyberg

# How to Achieve Fair Differentiation: Relating Flow Level QoS Requirements to DiffServ Packet Level Mechanisms

The thesis has been submitted for official examination for the degree of Licentiate of Science (Technology).

Supervisor: Professor Jorma Virtamo

| | |
|---|---|
| Author: | Eeva Nyberg |
| Title of the thesis: | How to achieve fair differentiation: Relating flow level QoS requirements to DiffServ packet level mechanisms |
| Language of the thesis: | English |
| Date: | 9.9.2002 |
| Pages: | 105 |
| Faculty: | Electrical and Communications Engineering |
| Chair: | Networking Technology, S-38 |
| Supervisor: | Professor Jorma Virtamo |
| Examiner: | PhD Samuli Aalto |



Quality of Service (QoS) is a general notion of introducing different quality levels to the best-effort Internet. Differentiation of traffic is needed both to guarantee quality requirements for real-time applications as well as to offer different levels of priority inside traffic classes. Differentiated Services (DiffServ) is a set of mechanisms aiming at providing scalable differentiation, where quality treatment inside the network is done to aggregates of flows not to individual flows. Quality guarantees can be divided into two service models. In assured services the flow should receive a rate at least equal to the contracted rate, while in relative services the rate of the flow should be proportional to the contracted rate.

The thesis models QoS mechanisms used and proposed for use in DiffServ networks. The modelling concentrates on the interaction between various differentiation mechanisms and traffic flows using a given transport protocol. This type of closed loop modelling of the dependency of the flow sending rate on the metering, marking, dropping and scheduling mechanisms has not been done before.

The mechanisms are modelled and analyzed on two levels: flow level and packet level. Furthermore detailed simulation of the packet level are performed. The correspondence between the models is good, and each modelling approach is able to illustrate a different aspect of mechanism design, either through the choice of parameter values or through the choice between two complementary mechanisms.

Based on the models, we study how well relative services can be achieved, i.e. in what proportion network bandwidth is divided, using DiffServ without admission control.


**Keywords:** Mathematical modelling, QoS, DiffServ, TCP, fairness

Contact information: Networking laboratory, P.O. Box 3000, 02015 HUT

The thesis is in electronical form at http://www.tct.hut.fi/julkaisut/lisurit/

| | |
|---|---|
| Kirjoittaja: | Eeva Nyberg |
| Työn nimi: | Kuinka eriyttää reilusti: Vuotason palvelunlaatuvaatimusten ja pakettitason DiffServ–mekanismien välinen yhteys |
| Työn kieli: | Englanti |
| Päiväys: | 9.9.2002 |
| Sivumäärä | 105 |
| Osasto: | Sähkö- ja tietoliikennetekniikan osasto |
| Professuuri: | Tietoverkkotekniikka, S-38 |
| Valvoja: | Professori Jorma Virtamo |
| Tarkastaja: | FT Samuli Aalto |



QoS eli palvelunlaatu on yleisnimitys laatuluokkien lisäämiselle best-effort Internetiin. Liikennettä eriytetään sekä reaaliaikaliikenteen laadun takaamiseksi että liikenneluokan sisällä olevien prioriteettitasojen tarjoamiseksi. Eriytetyt palvelut (DiffServ) koostuvat mekanismeista, joiden avulla pyritään toteuttamaan skaalautuvaa eriytystä verkossa. Ideana on, että verkon sisäsolmuissa laatuerottelua tehdään vuoryhmille, ei yksittäisille voille. Laatutakeet voidaan jakaa kahteen luokkaan. Taatussa palvelussa (assured services) vuon bittinopeus ei saisi olla pienempi kuin sopimusnopeus. Suhteellisessa palvelussa (relative services) vuon bittinopeuden tulisi olla verrannollinen sopimusnopeuteen.

Työssä mallinnetaan palvelunlaatumekanismeja, joita käytetään tai on ehdotettu käytettäväksi DiffServ–verkoissa. Tuloksena on mallinnettu eriytysmekanismien ja liikennevoiden välinen vuorovaikutus. Tämän tyyppistä suljettua mallia, joka huomioi mittaus-, merkkaus-, tiputus- ja skedulointimekanismien sekä voiden lähetysnopeuden välisen yhteyden, ei ole ennen toteutettu.

Mekanismeja mallinnetaan ja analysoidaan kahdella eri tasolla: vuo- ja pakettitasolla. Lisäksi pakettitasoa simuloidaan yksityiskohtaisemmalla simulointimallilla. Eri tasojen mallien yhtäläisyys on hyvä, ja jokainen malli ilmentää mekanismeista eri piirteitä, antaen joko osviittaa parametrien valinnalle tai kahden vaihtoehtoisen mekanismin valinnalle.

Mallien avulla voidaan tutkia, kuinka hyvin suhteellinen palvelu saavutetaan eli kuinka kaista jakautuu eri voiden välillä DiffServ–verkossa, jossa ei ole käytössä pääsynvalvontaa.


**Avainsanat:** Mallinnus, palvelunlaatu, DiffServ, TCP, reiluus

Yhteystiedot: Tietoverkkolaboratorio, PL 3000, 02015 TKK

Tämä työ on saatavilla sähköisessä muodossa WWW-osoitteesta
http://www.tct.hut.fi/julkaisut/lisurit/

# Preface

The title of the thesis is deliberately ambitious: How to achieve fair differentiation. The purpose of the work is not to give a final and concise answer to this question. However, reading through previous work conducted in this area, it has struck me that very few authors consider the question. Different mechanisms and proposals exist for bringing service differentiation into the Internet, but less effort has been put in evaluating these proposals and especially in comparing them in terms of the service they provide. By posing the question of how and by writing this thesis, I aimed at understanding better what Quality of Service (QoS) mechanisms are needed and why they are needed in the Internet.

Now as the thesis is ready and my understanding of QoS is better, I would like to thank my supervisor Jorma Virtamo for supporting me in my aim and letting me start this new research topic. My gratitude goes to my instructor Samuli Aalto. Without his insight, patience and true interest in my ideas this work would never have finished. Other people were also valuable especially in the beginning of my graduate studies; Kalevi Kilkki had a great influence on my work and topic through his two graduate level courses that I attended. He has also provided valuable help throughout the process.

I have also had the pleasure to work with summer trainees, who have each contributed to parts of the thesis. Many of the packet level mechanism modelling ideas of chapter 6 emerged through the work of Eemeli Kuumola on Assured Forwarding. Riikka Susitaival performed the simulations of chapter 7. With Laura Kneckt we polished up some of the buffer model equations of chapter 6. I thank each of you, and hope that I was able to offer you good supervision in return.

The work was carried out in the Networking laboratory and in the COST 257 and COST 279 projects funded by Nokia, Sonera, Nokia Research Center, Elisa Communications and Siemens. I have also received a personal grant from the Nokia foundation and have been a student of the graduate school GETA funded by the Academy of Finland since March 2001.

I thank my colleagues at the laboratory for useful discussions on work matters and on all those other topics. An especially great cheer for our laboratory's unofficial floorball team!

My friends and family have been the greatest as always. Many hugs and kisses. Olli I have the delight of thanking, hugging and kissing after yet another thesis.

Helsinki, September 9th, 2002

Eeva Nyberg

# Contents

# Chapter 1

# Introduction

New service concepts have been designed to overcome the problems of converged networks: delivering real time applications in the best-effort Internet. One of these concepts is Differentiated Services (DiffServ). DiffServ is termed a scalable architecture, as inside the network quality treatment is only offered to aggregate classes not to each individual traffic flow. The scalability is achieved at the price of losing strict quantitative guarantees and bounds on the level of service.

We can divide the differentiation into two categories: assured services and relative services. In assured services the flow should receive a rate at least equal to the contracted rate, while in relative services the rate of the flow should be in proportion to the contracted rate.

The approach taken in this work is to model the interaction between various differentiation mechanisms and traffic flows using a given transport protocol, namely Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). This type of closed loop modelling of the dependency of the flow sending rate on the metering, marking, dropping and scheduling mechanisms has to our knowledge not been done before.

The thesis is related to previous work by Sahu et al. [STK99], [SNT+00], May et al. [MBDM99] and Yeom and Reddy [YR01]. We also use results on TCP modelling by Floyd and Fall [FF99], Padhye et al. [PFTK98] and Kelly [Kel99], as well as results on flow level fairness studies by Kelly [Kel97], [KMT98], Massoulié et al. [MR99], [BM01] and others.

The new models presented in the thesis include a flow level model of differentiation among elastic flows, a packet level model of the interaction between differentiation mechanisms and flow congestion control as well as a simulation model of the corresponding network setting.

We study what kind of differentiation can be achieved using DiffServ without admission control. Specifically, previous work on DiffServ modelling and especially on the Assured Forwarding (AF) service proposal have shown that assured services cannot be achieved without admission control. We challenge this claim by stating that a more valid service requirement in DiffServ is relative services, where

the received rate does not have to be equal to the contracted rate, but where the link bandwidth should be divided in proportion to the contracted rates.

## 1.1 Outline of the thesis

The thesis is written to be as self contained as possible. In chapter 2 the fundamentals of the best-effort Internet relying on TCP congestion control are reviewed. This includes both a review on the congestion control mechanism as well as the packet level models and flow level models of TCP and its fairness in dividing bandwidth between flows. Chapter 3 reviews the Quality of Service mechanisms that have been proposed for the Internet, since it was noticed that the mechanisms of the best-effort Internet do not suffice in transporting both real time and elastic traffic. Chapter 3 also presents previous work related to DiffServ modelling.

Chapter 4 then presents our architectural model for a DiffServ Internet. We divide the mechanisms into flow level mechanisms at the boundary nodes and packet level mechanisms inside the DiffServ nodes. Based on this network model we study on the flow level, in chapter 5, on the packet level, in chapter 6, and using simulations, in chapter 7, how the division of bandwidth between elastic flows and between elastic and non-elastic flows depends on the DiffServ mechanisms of metering, marking, dropping and scheduling.

The flow level modelling of chapter 5 in essence extends the results of TCP fairness studies to include the effect that marks or priorities in a DiffServ network have on the fairness of bandwidth division of elastic flows. A more detailed model is given by the packet level model of chapter 6, where the TCP congestion control behavior, the priority mechanism and the forwarding treatment of packets is modelled. This results in a fixed point equation relating the TCP equilibrium sending rate to the priority dependent dropping rate of a scheduler. In chapter 7 the simulation model also demonstrates the effect that the time parameters of the metering mechanisms have on the achieved differentiation.

Chapter 8 then gathers the results of chapter 5, 6 and 7, in order to understand how the three levels of modelling relate to each other in terms of the results they deliver.

## 1.2 Related publications

Most of the results presented in the thesis have been published in conferences and technical documents. The first version of the results of chapters 5 and 6 were presented in [NAV01]. In [NAS02] the simulation results of chapter 7 were presented. In [NA02] the packet level modelling ideas of chapter 5 as well as the mechanism modelling approach of chapter 4 were refined to their present state, and in [AN02] the flow level modelling was strengthened.

# Chapter 2

# Best-effort TCP

Data delivery in IP has been designed to be undemanding, so that any underlying physical network can provide the service. The delivery is done through a connectionless datagram mode.

In datagram delivery, every packet contains the complete address to enable a switch to decide how the packet is forwarded to its destination. The packet can be sent anywhere at any time. When the packet is sent, there are no guarantees of delivery, packets can be forwarded independently and an alternate route can be found if a failure occurs at a switch.

The delivery is thus unreliable, as messages may be dropped, reordered, and delivered as duplicate copies. Other limitations of datagram delivery are that the messages are of some finite size and messages may be delivered after an arbitrarily long delay.

The transport protocol assures that the service is reliable, by guaranteeing message delivery, delivering the messages in the order they were sent, delivering at most one copy, supporting arbitrarily large messages, supporting synchronization between sender and receiver, allowing the receiver to apply flow control to the sender, and supporting multiple application processes on each host.

The transportation protocol together with flow control makes the datagram delivery service model reliable. By introducing congestion control or resource reservations, the now reliable best-effort service model can be made fairer and the scarce resources can be divided in a more optimal way. As TCP congestion control is the most prevailing way of introducing fairness to the network, the mechanism as well as its modelling will be discussed in this chapter and in this work instead of resource reservations.

## 2.1 The TCP mechanism

The TCP mechanism is described in RFC 793 [Pos81], dating back to 1981. The transport protocol is a connection oriented and end-to-end reliable protocol designed to fit into a layered hierarchy of protocols that support internetworking applications. The operative entities include a basic data transfer, reliability, flow control, multiplexing, connections, precedence and security.

During the connection establishment phase a TCP connection is formed with a three-way handshake mechanism with random initial sequence numbers sent by both sender and receiver to synchronize sequence numbers between the end-points. A TCP connection is uniquely identified by the 4-tuple of source and destination ports and addresses. The TCP header includes also the sequence number field for reliability, where the sequence number of the segment corresponds to the sequence number of the first byte.

The receiver advertised window (`rwnd`) and the acknowledgment (`ack`) fields are needed for flow control. The so called smart sender/dumb receiver rule means that the sender adjusts the size of its sliding window according to the size of the `rwnd`. The sender also paces the datagram transmission according to the `ack`s sent by the receiver. This is termed ack or self clocking.

To retransmit packets that are lost, adaptive retransmission is used. Segments are retransmitted if an ACK for the segment has not been received within a specific timeout. To this end, TCP has to have an effective and adaptive timeout as a function of both estimated round trip time (RTT) and its variance. The RTT is sampled once per RTT rather than once per packet using a coarse 500 ms clock. Furthermore each time TCP retransmits, the next timeout is set to twice the previous timeout, resulting in exponential backoff of the retransmit timer. Note that the original retransmission timeout algorithm did not have exponential backoff and did not take into account the variance of the estimation. These have been added later on. Karn and Partridge [KP91] have proposed the exponential backoff and Jacobson and Karels [V.88] the use of the variance of the RTT estimation.

## 2.2 Congestion control

### 2.2.1 Basic proposal

RFC 2581 [APS99] from 1999 defines the four congestion control algorithms to control traffic:

- slow start
- congestion avoidance
- fast retransmit

- fast recovery.

It also specifies how TCP should begin retransmission after a relatively long idle period as well as acknowledgment (`ack`) generation methods.

The TCP header field includes the sequence number and advertised window field (`rwnd`) for reliability and flow control. Adding congestion control requires adding a new variable to the TCP per-connection state, the congestion window (`cwnd`). The `rwnd` is the sender side limit and the `cwnd` the network limit on the amount of data the sender can transmit into the network before receiving an `ACK`.

The TCP source is not allowed to send faster than the slowest component – the network or the destination host – can accommodate. Thus the effective window of TCP is the minimum of these two variables (`cwnd` and `rwnd`).

#### 2.2.1.1 Slow Start and Congestion Avoidance

The purpose of slow start is to quickly probe the network to determine the available capacity in order to avoid congesting the network. Slow start is used in the beginning of a transfer or after repairing a loss detected by the retransmission timer. It is called slow start as opposed to sending an entire advertised window's worth of data at once.

Slow start ends either when a packet is lost or when a specific threshold is attained indicating that the algorithm should change to congestion avoidance mode. Congestion avoidance is thus used to slow down the sending rate, when the slow start mechanism has detected congestion.

For slow start and congestion avoidance, a state variable, the slow start threshold (`ssthresh`), is used to determine which mode, slow start or congestion avoidance takes place.

At the beginning of a connection, the sender only knows the advertised window size, not the congestion window size. Thus the initial congestion window is set to one segment.

The choice between using the slow start or the congestion avoidance algorithm is made based on the congestion state of the network indicated by the slow start threshold. When `cwnd` $<$ `ssthresh`, the slow start algorithm is used, otherwise when `cwnd` $>$ `ssthresh` congestion avoidance is used. Therefore, as slow start is always used in the beginning of the connection, the initial value of `ssthresh` may be arbitrarily high, e.g. equal to the advertised window. The threshold is then reduced in response to congestion.

Slow start is designed so that it approximates increasing the congestion window exponentially from a cold start by doubling the number of packets it has in transit every RTT. This is implemented so that during slow start TCP increments the congestion window by at most one sender maximum segment size (`SMSS`) worth of bytes for each received acknowledgment. Slow start continues until the `ssthresh`

is reached or until congestion is detected.

During congestion avoidance, the congestion window is incremented by 1 full sized segment per RTT, termed additive increase. It then continues until congestion is detected. TCP updates the congestion window per every received `ack`. There are two options in updating `cwnd`, both designed to approximate the increment of 1 full sized segment per RTT .

1. When a non-duplicate `ack` arrives, the update is

$$\texttt{cwnd} = \texttt{cwnd} + \texttt{SMSS}^2/\texttt{cwnd}.$$

2. This option requires an additional state variable to count number of bytes that have been acknowledged by `ack`s for new data. When `cwnd` bytes have been acknowledged, `cwnd` can be incremented by up to `SMSS` bytes. However, during congestion avoidance the increment may not be more than the maximum of 1 full sized segment per RTT or the increment given by option 1.

The difference between the options is in how the units of `cwnd` are calculated. If they are in bytes, option 1 is better suited. If units are in full sized segments, option 2 is easier to use.

When a loss is detected through the retransmission timer, the slow start threshold must be updated to

$$\texttt{ssthresh} = \max(\texttt{flightSize}/2, 2 * \texttt{SMSS}).$$

This corresponds to halving the congestion window, as `flightSize` is the amount of bytes in transit, i.e. the size of the congestion window.

Upon such a timeout the congestion window is bounded to 1 full sized segment. After retransmitting the dropped segment, the window is reopened at 1 full sized segment and slow start is used up to the halved value of `cwnd`. The additive increase mode of the congestion avoidance algorithm is employed once the threshold is attained.

Some alternatives to slow start in probing the network for congestion have been proposed. One interesting one is sending a packet pair, with no initial gap in between.The gap between the returning `ack`s, i.e. the delay, then indicates how much congestion is present.

#### 2.2.1.2 Fast Retransmission and Fast Recovery

When only slow start and congestion avoidance are used, losses are detected through timeouts. A faster way is to use the `ack`s sent by the receiver as an indication of loss and thus congestion.

Upon the receipt of a segment, the sender sends an acknowledgment. If the received segment is out-of order, the receiver cannot acknowledge that segment,

but sends a duplicate `ack`, re-acknowledging the last segment that has arrived successfully and in order. If the reason for an out-of order packet is a dropped segment, all subsequent segments will generate duplicate `acks`. Note that, duplicate acknowledgments may also be caused by reordering of data segments, or by replication of acknowledgments or of data segments by the network.

When the sender receives duplicate `acks`, it uses the fast retransmit algorithm to detect and repair the loss. The arrival of three duplicate `acks`, i.e. four identical `acks`, is used as an indication that a segment has been lost. The TCP then performs a retransmission of what is assumed to be the lost segment without waiting for the timer to expire. Fast retransmission thus gives an improvement in throughput as it eliminates half of the coarse grained timeouts on a typical TCP connection.

In the original TCP congestion control proposal [V.88], which did not include fast recovery, after three duplicate `ACK`s slow start was used. However, it was argued that the arrival of duplicate acknowledgments is an indication that packets are being transmitted and that these acknowledgments can also be used to clock the TCP transmission. This is called the fast recovery algorithm.

Following fast retransmission, the slow start phase is not entered, but fast recovery is used, where the congestion window is halved and transmission continues in congestion avoidance mode, until a non-duplicate `ACK` arrives.

The fast retransmission fast recovery algorithm has the following steps

1. Third duplicate `ACK` arrives
$$\texttt{ssthresh} = \max(\texttt{FlightSize}/2, 2 \cdot \texttt{SMSS})$$
2. Retransmit the lost segment and inflate the congestion window to
$$\texttt{cwnd} = \texttt{ssthresh} + 3 \cdot \texttt{SMSS}$$
3. For each duplicate `ACK` received, increment `cwnd` by `SMSS`
4. Transmit a segment if allowed by `cwnd` and `rwnd`
5. When the next `ACK` acknowledging the new data arrives, deflate the window by setting
$$\texttt{cwnd} = \texttt{ssthresh}$$

The algorithm does not recover very efficiently from multiple losses in a single flight of packets or for small window sizes, as there are not enough duplicate acknowledgments to clock the algorithm. Thus with multiple losses timeouts may occur and then the slow start phase is used to increase the window size. The modification to the algorithm when multiple losses occur is given in RFC 2582 [FH99].

With fast retransmit and fast recovery added to TCP congestion control, slow start is only used in the beginning of the connection and when coarse grained timeouts occur.

### 2.2.2 Algorithms in terms of implementations

The different versions of the TCP algorithm and especially the TCP congestion control algorithms have different names mainly based on the implementation version of TCP in the BSD Unix operating system. The original TCP with flow control and other transport related mechanisms was included in the 4.2 BSD Unix version in 1983.

As occurrences of congestion collapse were detected, congestion control was distributed in the 4.3 BSD Unix version. In 1988 the TCP congestion control algorithms included went by the name of TCP Tahoe. Tahoe included the original implementation of Jacobson's [V.88] congestion control mechanism: slow start, congestion avoidance, fast retransmit, but not fast recovery. Other improvements were better RTT variance estimation and exponential backoff of the retransmit timer.

In 1990 the TCP Reno was included in the 4.3 BSD Unix. TCP Reno included the fast recovery algorithm, header prediction to optimize for the common case that segments arrive in order, and delayed acknowledgments where only every second segment was acknowledged.

In 1993 the 4.4 BSD Unix operating system appeared with modifications to TCP for large bandwidth delay networks and gigabit throughput rates. It included window scaling, allowing a scaling factor for advertised window, Protect Against Wrapped Sequences, preventing sequence number wrapping on connections such as satellite links with long delays but high bandwidth capacity, and round trip time (RTT) measurement using time stamps.

In 1994 the TCP Vegas was included in the 4.4 BSD Unix. It was based upon the 4.3BSD Reno source. An improvement in throughput of up to 70% was achieved through the utilization of congestion anticipation. Congestion anticipation is done by calculating and comparing the expected throughput and achieved throughput every RTT. If the expected rate is clearly less than achieved rate TCP Vegas increases the congestion window linearly. If the expected rate is clearly more than the achieved rate, the congestion window is decreased linearly during the next RTT. In other cases, the congestion window is left unchanged. Though the congestion anticipation does increase the throughput in a network with only TCP Vegas, the acquiescent behavior of TCP Vegas may reduce its achieved throughput in a network with more aggressive TCP implementations such as Tahoe or Reno running concurrently.

### 2.2.3 Recent developments

One main focus of TCP congestion control research has been in developing loss recovery algorithms. As will be seen in the next section on TCP modelling, the target of TCP research has been in ensuring that after the initial setup using slow start, TCP works as if it were in congestion avoidance. Thus the focus has been on elimination of timeouts and the consequent re-entering of slow start and

to ensure that only actual packet losses are detected and only relevant packets retransmitted, i.e. fast recovery upon loss detection.

With the TCP selective acknowledgment (SACK) option (RFC 2018 [MMFR96]), the receiving TCP sends back SACK packets to the sender informing the sender of data that has been received. The sender can then retransmit only the missing data segments and it can retransmit more than one data segment per RTT.

Other developments in the area of retransmissions include increasing robustness when multiple packet losses occur without using SACK and adding more functionality to the ACK mechanism through ACK filtering and ACK congestion control.

Recent developments, outlined in [Flo01], to the TCP congestion control mechanism focus on reducing costs of unnecessary retransmissions. The article discusses in detail a limited transmit mechanism for transmitting new packets upon receipt of one or two duplicate ACKs and a SACK-based mechanism for detecting and responding to unnecessary fast retransmits or retransmit timeouts. Furthermore network changes such as explicit congestion notification and active queue management have also enhanced the principles of congestion control.

#### 2.2.3.1 Changes to algorithms

**Limited transmit** With fast retransmit and fast recovery, the sender has to wait only for three duplicate ACKs before transmitting again as opposed to waiting for a timeout to occur. In limited transmit, described in RFC 3042 [BF01] and first proposed in RFC 2760 [ADG+00], the sender transmits a new segment after receiving one or two duplicate ACKs, if allowed by the window size. Thus continuing transmission even though a possible loss may have occurred.

The first or second duplicate ACK is interpreted as evidence of a packet being delivered and not as an indication of a loss. Thus a new packet should be allowed into the network. This mechanism is robust to reordered packets, as it doesn't retransmit but sends a new packet. Loss is indicated only when the third duplicate ACK arrives. Then fast retransmit and fast recovery are used, with the window halved and the TCP congestion control mechanism working as described in the previous section.

**D-SACK** The SACK option was designed for acknowledging out-of-sequence data not covered by TCP's cumulative acknowledgement field. RFC 2883 [FMMP00] extends RFC 2018 [MMFR96] by specifying the use of the SACK option for acknowledging duplicate packets. When duplicate packets are received, the first block of the SACK option field can be used to report the sequence numbers in question. This extension, named D-SACK, gives the sender information to determine when an unnecessary retransmission has occurred. A TCP sender can then use this information if reordered packets, acknowledgment losses, packet replications, and/or early retransmit timeouts occur. The advantage is in a more

robust operation of the TCP congestion control algorithm, taking into account different reasons for duplicate ACKs.

If the sender has responded by an unnecessary congestion control action to reordered or delayed packets with D-SACK, the sender may, after 1 RTT time, determine if the retransmitted packet was necessary. If it was unnecessary, the sender can undo the halving of the congestion window by setting `ssthresh` to the previous value of the window, i.e. reentering slow start, or by restoring slow start. The old window is thus recovered in one RTT, not in half of a window size of RTTs. The sender may then also adjust the duplicate ACK threshold or retransmit timeout parameters.

#### 2.2.3.2 Changes to network

**Active Queue Management** Active queue management (AQM) algorithms, of which random early detection (RED) is the most popular example, are implicit congestion notification algorithms that drop packets in case of congestion at routers.

With every queue management algorithm there is a tradeoff in the network between low delay and high throughput. The main motivation for employing active queue management (AQM), instead of simple tail drop, is to control queueing delays, while at the same time preventing transient fluctuations in the queue size from causing unnecessary packet drops. For environments with the same worst case queueing delay for tail drop and AQM, the lower average queue size maintained by AQM can sometimes come at the cost of a higher packet drop rate. On the other hand, with highly bursty packet arrivals, tail drop results in unnecessary large packet drops. A higher drop rate wastes resources up to the congested link and introduces higher variance in transfer times.

In RED the queue management is based on two main features. First the dropping decision is made based on average queue length, essentially filtering out changes in queue length in shorter time scales. Secondly, the dropping of packets is done with a positive low probability before the queue is full, the probability of dropping packets increasing to one as the queue length increases.

Furthermore, through RED, dropping of packets are randomized, thus bringing the TCP algorithm closer to the ideal models of TCP introduced in the next section.

Further improvement is obtained by using AQM algorithms to explicitly notify of congestion through marking of packets. Then the indication of congestion is given in advance by AQM algorithms through marking of packets; drops and retransmissions are not necessarily required.

**Explicit Congestion Notification** Instead of implicitly notifying the end hosts of congestion by dropping packets, ECN explained in RFC 2481 [RF99], can be used to notify end hosts explicitly by marking packets as an indication of

congestion. For TCP sources ECN behaves in the same way, as the response to a mark is the same as the response to a packet loss. The advantage, however, is for real time or interactive traffic that do not use TCP because of losses or delays from retransmitting lost segments.

For TCP, the main difference is that no retransmissions are needed and there is no retransmission timeout. However, packets are still dropped when queues are full, and ECN can thus not eliminate packet losses altogether. Therefore, there is still need for the TCP extensions such as limited transmit and D-SACK for undoing the unnecessary halving of the congestion window.

## 2.3 TCP models

The TCP models in the literature are often based on the ideal and canonical form of the TCP congestion control algorithm [MSMO97]. It is assumed that

1. The sender is saturated, it has always a congestion window amount of traffic to send.
2. The receiver advertised window is always larger than the congestion window, i.e. the bottleneck is the network not the receiver.
3. All segments are of size maximum segment size (MSS).
4. Recovery from lost segments does not interact with congestion avoidance, i.e. SACK is implemented.
5. The connection does not contribute to delays and the RTT is independent of the window size.
6. Loss intervals are from a defined process, e.g. deterministic.

Some of these assumptions have been relaxed, as the research has advanced, namely assumptions 2 and 4. However, the focus of the design of TCP congestion control has at the same time been in pushing the algorithms towards the ideal model, which limited transmit and D-SACK, as well as the earlier development of fast recovery and retransmit algorithms are examples of.

Note further that the ideal model assumes that the connection is in congestion avoidance mode. The slow start phase is ignored as it is usually of short duration and with new developments to TCP congestion control should only occur at the beginning of a connection.

We consider three TCP congestion control models. The heuristic model by Floyd, the model by Padhye et al., which relaxes most of the ideal TCP assumptions, and the model by Kelly for a collection of TCP connections.

We then give a brief discussion on how different network conditions may affect the use of the listed assumptions and thus the use of the ideal TCP models.

### 2.3.1 Model by Floyd and Fall

In [Flo91] a steady state model for multiple TCP gateways using different rate increase algorithms and having different RTTs is given. In [FF99] a steady state model for the throughput of a TCP connection as a function of the packet drop rate is given. The model derivations are based on heuristics, but have been validated through simulations and other analytical models. We will concentrate on the latter model, which is able to capture the basics of the throughput of a TCP connection as a function of RTT and packet drop rate.

The idea is to calculate the throughput if a single packet is dropped from a TCP connection each time the congestion window has increased to $W$ packets. The average steady-state packet drop probability is $p$, when an individual TCP connection has at most one packet drop in a window of data when in steady-state. Each time a packet is dropped, the congestion window size has increased to $W$, and upon packet drop is halved to $W/2$. The cycle of the TCP connection is thus the time it takes for the sender to increase its window size from $W/2$ to $W$.

The TCP connection sends segments with maximum size $B$ bytes ($B = MSS$). The RTT is assumed to be fairly constant and it includes the queueing delays. The derivation of the equations is done in terms of a lower bound on RTT.

Under the given assumptions, the TCP sender thus transmits

$$\frac{W}{2} + (\frac{W}{2} + 1) + \cdots + W \approx \frac{3}{8}W^2$$

per each packet dropped. Thus

$$p \leq \frac{8}{3W^2}$$

and

$$W \leq \sqrt{\frac{8}{3p}}.$$

Now, there are $W/2$ round trip times between packet drops and as the TCP connection sends $\frac{3}{8}W^2$ packets between packet drops, the throughput $T$ (in terms of bits) is

$$\begin{aligned} T &= \frac{0.75WB}{RTT} \leq \frac{1.5\sqrt{2/3}B}{RTT\sqrt{p}}, \\ T &\leq \frac{\sqrt{3/2}B}{RTT\sqrt{p}}. \end{aligned}$$

If delayed ACKs are used, an acknowledgment is sent for every second packet, and the throughput is

$$T \leq \frac{\sqrt{3/4}B}{RTT\sqrt{p}}. \tag{2.1}$$

### 2.3.2 Model by Padhye et al.

The TCP model by Floyd, does not take into account TCP delays due to waiting for retransmit timers to timeout. The model in [PFTK98] is able to take into account retransmit timeouts and the congestion window size being limited by the receiver's advertised window. The model does not, however, include the fast recovery algorithm and is thus a model for TCP Tahoe.

The model assumes that the TCP sender is saturated, that the RTT is independent of window size and thus the throughput can be given in terms of average RTT. Here the cycle of a TCP connection starts with the back-to-back transmission of $W$ packets (current size of congestion window), and the first ACK marks the end of the cycle.

The model includes the possibility of delayed acknowledgments, with $b$ being the number of packets that are acknowledged by a received ACK. For the delayed ACK algorithm $b = 2$.

A packet loss occurs either through the arrival of a triple duplicate ACK or through a timeout. The packet losses are assumed to be independent from losses in other cycles, but inside a cycle, once a packet is lost all remaining packets transmitted in that cycle are also lost. The loss process can be characterized as deterministic [Bar01].

If the loss indications result from triple duplicate ACKs, the throughput (in terms of segments) is

$$T = \frac{1}{RTT}\sqrt{\frac{3}{2bp}} + o(1/\sqrt{p}). \tag{2.2}$$

If the loss indications are due to triple duplicate ACKs and timeouts, the throughput is

$$T \approx \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\frac{3bp}{8}\right)p(1+32p^2)},$$

where $T_0$ is the length of the first timeout period assuming that following a timeout the window is reduced to one and linear increase is resumed.

The derivation of the throughput for the impact of the congestion window limitation is also found in [PFTK98], taking into account the small windows advertised by current Internet receivers. However, assuming employment of window scaling and increasing the advertised window field, the model is not of such importance [Bar01].

### 2.3.3 Model by Kelly

In [Kel99] the behavior of TCP connections is studied through differential equations for the rate control of TCP. Differential equations and their steady state

solutions are given for a network with many links and routes. Here, we will present the results for one link.

Let $x_r$ be the rate allocated to the user $r \in \mathcal{R}$. For a collection of $m$ single TCP connections the differential equation of the so called MulTCP algorithm is derived as follows. The expected change in the congestion window `cwnd` per update step is

$$\frac{m}{\texttt{cwnd}}(1-p) - \frac{\texttt{cwnd}}{2m}p,$$

with $p$ being the probability of congestion indication. As the time between update steps is $RTT/\texttt{cwnd}$, the expected change in rate $x$ per unit time is approximately

$$\frac{(\frac{m}{\texttt{cwnd}}(1-p) - \frac{\texttt{cwnd}}{2m}p)/RTT}{\frac{RTT}{\texttt{cwnd}}} = \frac{m}{RTT^2}(1-p) - \frac{x^2}{2m}p.$$

In equilibrium, we have

$$x_r = \frac{m_r}{RTT_r}\sqrt{\frac{2(1-p_r)}{p_r}}. \tag{2.3}$$

The steady state result for the TCP congestion control of equation (2.3) has the same $\frac{1}{RTT\sqrt{(p)}}$ relation as the relations (2.1) and (2.2). The reason why (2.3) is adopted here is that the differential equation and rate control framework it was built from is best suited for our work.

### 2.3.4 TCP model assumptions

In [Bar01] TCP modelling is divided into two parts: the modelling of the TCP mechanisms and the modelling of the network reaction to congestion. In terms of the ideal model presented in the beginning of the section, we have

- Network assumptions
  1. The connection does not contribute to delays and the RTT is independent of window size.
  2. Loss intervals are from a defined process, e.g. deterministic.
- TCP mechanism assumptions
  1. The sender is saturated, it always has a congestion window amount of traffic to send.
  2. The receiver advertised window is always larger than the congestion window, i.e. the bottleneck is the network, not the receiver.
  3. All segments are of size MSS.
  4. Recovery from lost segments does not interact with congestion avoidance, i.e. SACK is implemented.

All the models presented in the previous section assume that the connection is in the congestion avoidance mode and therefore implicitly assume that the connection is in additive increase multiplicative decrease (AIMD) mode as a function of time.

The models further assume that the RTT is constant and independent of the connection's window size. If this assumption was not made, the increase would be linear only with respect to RTT, but not with respect to time. For window sizes small compared to the large bandwidth delay product of the network, linear increase with respect to RTT would be linear with respect to time. However, if the window sizes are large and thus contribute to the RTT, the simple models overestimate the throughput. At this point, no models taking the sub-linearity into account and giving closed form expressions exist for the TCP throughput.

The multiplicative decrease holds, if the congestion detection is done using the fast retransmit and fast recovery mechanisms. However if congestion is detected only after a timeout, TCP will go into slow start mode and the assumption of congestion avoidance mode continuing does not hold. Fortunately, this is an example where the TCP mechanism has been improved towards the ideal model and the assumption of multiplicative decrease upon congestion indication holds.

Modelling of window limitation was already done in [PFTK98]. Furthermore, its importance is diminishing as larger advertised window fields are added to the TCP protocol.

In modelling the TCP mechanisms a further assumption of continuous congestion window increases is often made. These fluid models are used as they simplify the analysis. The fluid models, however, overestimate the throughput. In [Bar01] a correction factor to the overestimation of throughput when using fluid models is given. The model of [PFTK98] is an example of a discrete model while the model by [Kel99] is an example of a continuous model.

On the network level, there are two main options. Either the network behavior is modelled, where as a result an indirect characterization of the occurrences of losses is obtained. An easier task is to model directly the loss occurrences as seen by the connection. As discussed, the models given in [PFTK98] assume a deterministic loss process. Poisson process is also used, e.g. in [Flo91]. In [AAB00] it is shown that the throughput is an increasing function of the variance of interloss times. Thus if a deterministic process is used instead of one with some variance, the throughput is underestimated.

## 2.4 Fairness

Linked with the study of TCP models and the resulting steady state throughputs is the study of how fair TCP is in dividing bandwidth between competing flows. The notion of fairness was set to assess this question. The study of pricing schemes is also strongly wrapped around the fairness concepts, mainly due to its economical heritage.

Fairness is a measure of the equal fulfilment of requirements of the network users in the same way as efficiency is a measure of the fulfilment of the technical requirements. As fairness measures the satisfaction of customer needs, it can be assessed using utility functions and social optima calculations. This has been the formulation in the work performed by Kelly et al. [Kel97] and [KMT98].

If a price is introduced that depends on the flow pattern $x$, we can formulate the measure of fairness as an optimization problem. Given a utility function $U_r(x_r)$, the optimization problem of maximizing overall utility results in charging and allocating network resources according to some fairness criteria. Besides the work by Kelly et al. different fairness concepts and TCP fairness have been studied by Massoulié et al. [MR99] and [BM01], by Mo and Walrand [MW00], by Ben Fredj et al. [FBAPR01] and by Vojnovic et al. [VBB00].

Ideally, assuming all customers are well behaving, the use of TCP should result in each flow receiving the same amount of service, e.g. capacity and packet loss. A natural extension is to design a mechanism that offers the same amount of service inside a class, while between classes the service offered differs by a given proportion.

### 2.4.1 Theoretical fairness concepts

Assume that a flow is related to the route $r \in \mathcal{R}$ and has utility function $U_r(x_r)$, where $x_r$ is the rate allocated to each flow on route $r$, and forms a vector $x = \{x_r, r \in \mathcal{R}\}$. Denote by $n_r$ the number of such flows. The objective is to maximize the overall utility of the system (network and flows), i.e. to find the social optimum.

$$\begin{array}{ll} \max & \sum_{r\in\mathcal{R}} n_r U_r(x_r) \\ \text{subject to} & Ax \leq C \\ \text{over} & x \geq 0. \end{array}$$

Here the matrix $A = \{a_{jr}, j \in \mathcal{J}, r \in \mathcal{R}\}$ has $a_{jr}$ equal to the number of flows $n_r$ on route $r \in \mathcal{R}$ that use link $j \in \mathcal{J}$, and $C$ is the vector of the link capacities $C = \{C_j, j \in \mathcal{J}\}$.

Let us consider the utility function $U_r(x_r)$ of the general form introduced in [MW00] and [BM01],

$$U_r(x_r) = \frac{x_r^{1-\alpha}}{1-\alpha}.$$

If a price or weight $w_r$ is introduced to each route, the optimization problem can be written in a more general way, with the utility function

$$U_r(x_r) = w_r \frac{x_r^{1-\alpha}}{1-\alpha}.$$

Table 2.1 summarizes the resulting optimization problem for four fairness types assuming weights $w_r = 1$. Note that the constraints stay unchanged. The four

types, given at the limit when $\alpha \to 0$, 1, 2, $\infty$, are: maximizing overall throughput of the network, dividing the bandwidth in a proportionally fair manner, minimizing the potential delay and maximizing the minimum allocation.

Table 2.1: Relation between $\alpha$ and four fairness concepts [BM01]

| $\alpha \to$ | concept | $\max_x \sum_{\mathcal{R}} n_r U_r(x_r)$ |
|---|---|---|
| 0 | maximize overall throughput | $\max_x \sum_{\mathcal{R}} n_r x_r$ |
| 1 | proportional fairness | $\max_x \sum_{\mathcal{R}} n_r \log x_r$ |
| 2 | minimize potential delay | $\min_x \sum_{\mathcal{R}} \frac{n_r}{x_r}$ |
| $\infty$ | max min fairness | $\max_x \min_{\mathcal{R}} n_r x_r$ |

Note that the limit, as $\alpha \to 1$, is

$$\begin{aligned} & \lim_{\alpha \to 1} \max_x \sum_{\mathcal{R}} n_r \frac{x_r^{1-\alpha}}{1-\alpha} \\ = \; & \max_x \sum_{\mathcal{R}} n_r \lim_{\alpha \to 1} \frac{x_r^{1-\alpha}}{1-\alpha} \\ = \; & \max_x \sum_{\mathcal{R}} n_r \log x_r. \end{aligned}$$

Given that

$$\lim_{\beta \to 0} \frac{x^\beta}{\beta} \;=\; \log x + \lim_{\beta \to 0} \frac{1}{\beta}.$$

And the limit, as $\alpha \to \infty$, is

$$\begin{aligned} & \lim_{\alpha \to \infty} \max \sum_{\mathcal{R}} n_r \frac{x_r^{1-\alpha}}{1-\alpha} \\ = \; & \min_x \lim_{\alpha \to \infty} \sum_{\mathcal{R}} \frac{n_r}{(\alpha-1) x_r^{\alpha-1}} \\ = \; & \max_x \min_{\mathcal{R}} n_r x_r, \end{aligned}$$

as the minimum sum is achieved when the smallest $x_r$ is as large as possible.

In the case of only one bottleneck link, each of the fairness objectives results in dividing the bandwidth equally among the flows on that link. For a more general network, the topology of the network affects the way bandwidth is divided. For example, in [BM01], the topologies considered are a linear network, a cyclic network and a grid network.

As shown by examples in [MR99] and [BM01], in a simple linear network depicted in figure 2.1 with routes either passing through only one link or routes passing

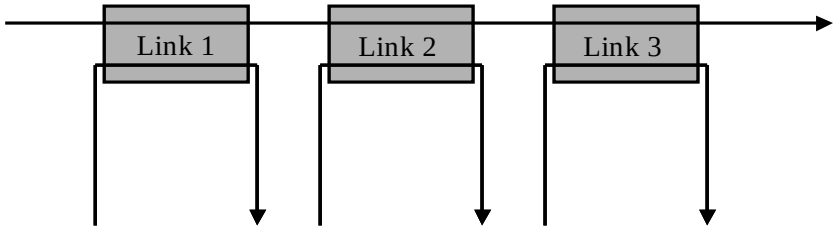



Figure 2.1: The example linear network

through all the links, maximizing throughput results in giving no bandwidth to the long route and all the bandwidth on each link to the routes using only that link. For the other fairness definitions table 2.2 summarizes the resulting bandwidth allocations. Here $r = 0$ represents a long route passing through all links and $r = j$ a route passing through only link $j$. The capacity constraint is

$$n_0 x_0 + n_j x_j = 1, j \in \mathcal{J}.$$

| $\alpha \rightarrow$ | concept | $n_0 x_0$ | $n_j x_j$ |
|---|---|---|---|
| 0 | maximize overall throughput | 0 | 1 |
| 1 | proportional fairness | $\frac{n_0}{n_0 + \sum_{\mathcal{J}} n_j}$ | $\frac{\sum_{\mathcal{J}} n_j}{n_0 + \sum_{\mathcal{J}} n_j}$ |
| 2 | minimize potential delay | $\frac{n_0}{n_0 + \sqrt{\sum_{\mathcal{J}} n_j^2}}$ | $\frac{\sqrt{\sum_{\mathcal{J}} n_j^2}}{n_0 + \sqrt{\sum_{\mathcal{J}} n_j^2}}$ |
| $\infty$ | max min fairness | $\frac{n_0}{n_0 + \max_{j \geq 1} n_j}$ | $\frac{\max_{j \geq 1} n_j}{n_0 + \max_{j \geq 1} n_j}$ |

Table 2.2: Bandwidth allocations $n_r x_r$ in a linear network for the four different fairness criteria, $r = 0$ represents a long route passing all links and $r = j$ a route passing through link $j$ [MR99].

As $\alpha$ increases from 0 to $\infty$, the different allocations give relatively more bandwidth to long routes.

### 2.4.2 Fairness of TCP

The fairness of TCP can be evaluated in two ways. Ideal TCP congestion control can be modelled to be of form general additive increase general multiplicative decrease and based on this assumption the type of fairness, e.g. max-min or proportional can be deduced. On the other hand the fairness of TCP can be evaluated based on how well actual TCP implementations are able to divide the capacity in a non-ideal network.

Kelly et al. [KMT98] and Massoulié and Roberts [MR99] show that additive increase multiplicative decrease (AIMD) is proportionally fair. Based on the

ideal model of TCP, which states that TCP is an AIMD mechanism, it can be thus deduced that best-effort TCP using congestion control is proportionally fair.

Veciana et al. [dVLK01] employed these results and studied the stability of the network with dynamic number of flows assuming that flows share bandwidth according to a given fairness criteria.

Ben Fredj et al. [FBAPR01] have studied the statistics of the realized bandwidth share for elastic document transfers, including short lived flows also. They conclude that with similar RTTs and one bottleneck link TCP tends to share bandwidth equally among all flows. Furthermore, they demonstrate that if the bandwidth allocated to a flow $r$ on a link $j$ is given by the TCP equation, where $K$ is a constant, e.g. $\sqrt{3/2}$,

$$x_r = \frac{K}{RTT_r\sqrt{\sum_{j\in r} p}},$$

then this is a unique solution to the optimization problem

$$\begin{array}{ll} \max & \sum_{r\in\mathcal{R}} w_r n_r \frac{x_r^{1-\alpha}}{1-\alpha}, \\ \text{subject to} & Ax \leq C, \\ \text{over} & x \geq 0. \end{array}$$

for $\alpha = 2$ and $w_r = 1/RTT_r^2$, demonstrating, contrary to Kelly et al. [KMT98] and Massoulié and Roberts [MR99], that TCP divides bandwidth such that potential delay is minimized according to weights proportional to the square of RTT.

As demonstrated above, due to the different level of modelling TCP, the fairness of TCP is either max min fair ($\alpha \to \infty$), proportionally fair ($\alpha \to 1$) or something in between e.g. minimizing potential delay. Vojnovic et al. [VBB00] give a TCP fairness model assuming AIMD, heterogeneous RTTs and rare rate proportional congestion feedback. They term the resulting fairness, which is between max min fairness and proportional fairness, as $F_a^h$-fairness.

Several factors affect the ideal TCP model based on AIMD and affect thus the modelling of its fairness. The main factor is the bias of TCP against connections with long round trip times, which results from the linear increase of one unit per RTT. Other factors are: the implementation differences, especially in terms of aggressiveness, between different versions of TCP, the difference in the TCP algorithm for short lived flows compared to long lived flows, the topology and routes of the network and the number of bottleneck links on the route and finally the amount of non-TCP traffic in the network. TCP friendliness assesses these issues by testing for the different biases.

## 2.5 TCP friendliness

TCP congestion control was designed as a response to the congestion collapse observed in the Internet in the late 1980s. However, not all forms of traffic

want to use TCP as the transport mechanism. Namely, TCP together with its congestion control mechanism is suited for traffic that prefers delays to packet losses, i.e. one where a lost packet is always retransmitted. Real time traffic such as voice and video can sustain losses but a change in the delay or jitter of the transmission results in an apparent decrease in quality. Therefore these sources rather use the other popular transport protocol of the Internet: UDP, where no flow control nor congestion control is employed.

Because the UDP sources have no sending rate backoff mechanism, they may capture all of the network capacity from the TCP flows, when the TCP flows have reduced their sending rate as a response to congestion notification.

It would be desirable to have a way to control the sending rate of non-TCP traffic so that the capacity is shared in a fair manner by all flows in the case of congestion in the network. Furthermore, even flows using TCP do not share bandwidth equally due to the biases discussed in the previous section. The tests to control the sending rate of non-TCP flows can also be used to eliminate the bias between TCP flows. These are the ideas behind the notion of TCP friendliness.

### 2.5.1 UDP traffic

User Datagram Protocol (UDP) is a transport protocol that adds demultiplexing functionality allowing multiple application processes on each host to share the network. Processes indirectly identify each other using an abstract locater, e.g. the port. The UDP header includes the source and the destination ports. The UDP thus provides a connectionless datagram service, as opposed to TCP, which is connection oriented. UDP is also unreliable, as it does not use flow control, i.e. acknowledgments to clock its sending rate.

### 2.5.2 Tests for fair sharing of bandwidth

A flow is *not TCP friendly* if its long term arrival rate exceeds that of any conforming TCP flow in the same circumstances. A flow is *unresponsive* if it fails to reduce its sending rate at a router in response to an increased packet drop rate. A flow is a *disproportionate-bandwidth* flow if it uses considerably more bandwidth than other flows in a time of congestion. These definitions are given in [FF99]. We define a flow to be *non-conforming* if it is *not TCP friendly*, *unresponsive* and if it is a *disproportionate-bandwidth* flow.

In [FF99] it is argued that end-to-end congestion control through router mechanisms are needed to identify and restrict the bandwidth of *all* flows in times of congestion. Other mechanisms proposed are pricing mechanisms and per flow scheduling mechanisms.

UDP flows are non-conforming flows as opposed to TCP flows, which are conforming. The effect of UDP flows is twofold. They can starve the bandwidth from the TCP flows, resulting in unfair division of capacity. The UDP flows may

also, due to their unresponsive nature, bring to pass congestion collapse, where the network is in a state of transmitting packets that will be discarded before reaching the destination.

The approach taken in [FF99] is to build such incentives into the network that flows find responding to congestion signals in a TCP friendly way as their most appealing choice. Floyd and Fall found the incentives on the notion of cooperation; cooperation in sharing bandwidth in times of congestion.

In order to be sure that all flows are conforming, the routers must identify the flows that do not cooperate, i.e. those that are *not TCP-friendly*, *unresponsive* or using *disproportionate bandwidth*, and regulate them.

#### 2.5.2.1 Test for TCP friendliness

A flow sending $B$ bytes is *TCP friendly* if its maximum sending rate is that of a TCP flow. Any equation modelling TCP throughput could also be used, but let us define TCP friendliness using equation (2.2),

$$T \leq \frac{1.5\sqrt{2/(3b)}B}{RTT\sqrt{p}}.$$

The test can only be applied to a flow at the level of granularity of a single TCP connection. Furthermore, equation (2.2) is suited for networks and routers with large bandwidth delay products, but not for routers with attached links having smaller propagation delays. Thus the TCP friendliness test by itself is not enough to guarantee cooperative behavior.

#### 2.5.2.2 Test for unresponsiveness

A more general test is to verify that a high-bandwidth flow does actually respond to a congestion notification by decreasing its arrival rate. Equation (2.2) can again be used. If the drop rate is increased by a factor of $x$ a flow should decrease its rate by a factor of $\sqrt{x}$ or more. The same principle can be applied to an aggregate of flows. The test for unresponsive flows would be applied to high bandwidth flows.

Both the test for TCP friendliness and response to congestion have limitations. The main limitation is the way bandwidth is shared. As already shown in section 2.4, TCP does not necessarily divide bandwidth in equal shares in more complicated network scenarios. Furthermore, as the actual type of fairness of TCP traffic is questionable, and there is a bias in the relationship of the throughput to the RTT, the above tests do not remove the unwanted bias of the bandwidth share as a function of RTT.

#### 2.5.2.3 Test for disproportionate bandwidth

The authors of [FF99] wish to test for flows using *disproportionate bandwidth* in times of high congestion. Due to difference in RTTs, use of window scaling or difference between short and persistent TCP flows, a TCP flow may have a *disproportionate share of bandwidth*. Flows can be forced to share bandwidth in a max min fair manner by checking if a flow's rate is more than $1/n$th of the aggregate rate. The authors have chosen a cut off rate of $\frac{\log 3n}{n}$, as it is close to one for $n = 2$ and grows slowly as a multiple of $1/n$. Because there is a relationship between the arrival rate and the packet drop rate, the second step is to test the rate relative to the level of congestion. The rate should be less than $\frac{c}{\sqrt{p}}$, for some constant $c$.

#### 2.5.2.4 Comparison of the tests

Of these tests, the third one is clearly the most questionable, as it attempts to change the resulting bandwidth share of TCP. This should rather be done at the design level of the TCP congestion control mechanism, not by adding tests to be performed at the routers.

The first and second tests aim at giving incentives for flows to adjust their sending rate to the steady state of the TCP flows, without having to do it in the "TCP way". In the sections to come it is especially the *TCP friendliness* of a non-TCP flow, i.e. the steady state behavior of the flow, which will be considered.

Furthermore as discussed in section 2.3 and in [Bar01] the closed form TCP throughput formulae have to be good enough in order to be usable as the basis for the tests. If the closed form expressions used underestimate or overestimate the throughput, the tests are not accurate enough and cannot be used in the networks.

# Chapter 3

# Quality of Service: guarantees and differentiation

As the section on TCP friendliness showed, problems arise when real-time applications are delivered in the best-effort Internet. As mentioned, real-time applications suffer from variable queueing delays and large losses due to congestion. On the other hand, aggressive real-time applications may starve bandwidth from TCP traffic. Furthermore, real-time applications would often desire to take advantage of the multicasting possibilities of the Internet.

To this end the IntServ service model was designed. Integrated Services aimed at providing control over end-to-end packet delays and was designed primarily for multicasting. Coupled with this aim, the problem of sharing bandwidth fairly was also addressed by IntServ by introducing controlled link sharing between and among real-time and best-effort traffic through per flow reservation.

The Differentiated Services (DiffServ) architecture [BBC+98] was designed to provide service in a scalable way. Service is defined as a set of significant characteristics of packet transmission in one direction across a set of one or more paths within a network. The characteristics may be in terms of relative priority in accessing network resources or in terms of quantitative or statistical variables such as throughput, delay, jitter or loss.

Scalability of DiffServ, compared to the less scalable requirement of per flow state in IntServ routers, is achieved by performing complex classification and conditioning functions at network boundary nodes to aggregate traffic into forwarding classes. Simple forwarding and discarding functions are then performed on the aggregates inside the network.

It is appropriate to distinguish two main DiffServ aims: relative services and assured services. The former does not make any quantitative nor statistical guarantees on the absolute level of service, but only guarantees that the higher priority class receives better service than the lower priority class. Furthermore, relative services may guarantee that the service is better by a given factor, e.g. proportional to the weight. Assured services, on the other hand, aim at guaranteeing

either statistically or quantitatively, some minimum level of service, while the leftover capacity is then divided equally or in some other way among the flows.

Per flow state and IntServ have better tools in realizing assured services especially if the core network is the bottleneck. With the use of aggregate scheduling instead of per flow scheduling, relative differentiation of aggregates is achieved, but it is not clear how differentiation is achieved inside the aggregates. The same applies to assured services, since if the contracted rate of a flow or aggregate cannot be achieved, it is not clear how the capacity is then divided between the flows or the aggregates. As the classification and conditioning of the aggregates is done on the boundary of the DiffServ node, it is the design of those mechanisms that strongly affects how the aggregates are formed and thus how the bandwidth inside an aggregate is divided among the flows.

## 3.1 IntServ

RFC 1633 [BCS94] gives an overview of Integrated Services (IntServ) as seen in 1994. Since then the work has been refined in [Wro97b], [Wro97a] and [SPG97]. The objective of IntServ, as given in [BCS94], is to offer guaranteed and predictive services in addition to best-effort service and thus implicitly change the Internet best-effort service model. The new model uses controlled link sharing, implemented by resource reservations and admission control. The QoS requirements of the admitted flows are met through an explicit resource reservation setup mechanism, per flow state in the routers and advanced packet scheduling mechanisms. The measures used are latency and fidelity, a measure of loss, reordering or delay of packets.

### 3.1.1 Integrated services model

The service model of IntServ was created to integrate real-time services with existing elastic services of the Internet. The service model is thus concerned exclusively with the time of delivery of packets, i.e. the per packet delay. Quantitative service commitments are given as bounds on the maximum and minimum delays.

Intolerant applications need perfectly reliable upper bounds on delay and are termed *guaranteed services*. They are designed for playback type of applications and mission control systems with hard real-time requirements. These applications need guaranteed bandwidth and strict and a priori information on the end-to-end maximum delay.

Tolerant applications can sustain some late packets, but require fairly reliable delay bounds and are named *predictive or controlled load services*. Predictive services can also be termed better than best-effort services, as the objective is to emulate a lightly loaded best-effort network. Thus statistical multiplexing is allowed and admission control is local and implementation specific. As an

extension, predictive services can be divided into many levels of elastic services.

### 3.1.2 Elements of the architecture

To calculate the delay bounds and make the admission control decision, traffic needs to be characterized and the service requirements need to be specified. Reservations are then setup based on the QoS requirements and the given traffic profiles and using the routing decision and the required resources given by the admission control. The per flow packet scheduling functions ensure that the requirements of the admitted traffic are met to the required precision. Packet scheduling and admission control are also used for resource sharing between entities of many flows.

#### 3.1.2.1 Traffic characterization

The flow specification is the service contract of IntServ. It is a list of traffic parameters used to characterize the traffic and a list of parameters used to specify the service requirements. At the edge of the network, traffic can be policed to check that the flow conforms to the traffic parameters.

The actual parameters included in the flow specification depend on the service type required by the flow i.e. it depends on the admission control and packet scheduling that is used to forward the flow.

The flow specification of guaranteed services is divided into the traffic descriptor TSpec and service specification RSpec, while controlled load services only specify the TSpec parameters.

Traffic is characterized by token bucket parameters, thus the TSpec parameters include:

- Token bucket rate or average rate
- Peak rate
- Bucket depth or maximum burst size
- Minimum policed unit
- Maximum packet size

The RSpec parameters are

- Service rate, i.e. bandwidth requirement
- Slack term, the extra amount of sustainable delay

The flow specification is carried by the reservation setup protocol and is passed to the admission control and if accepted, is used to parameterize the packet scheduling mechanisms.

#### 3.1.2.2 Traffic control for delay bounds

Once the traffic is characterized, traffic control is used to offer guarantees and predictable bounds on performance measures. Traffic control is implemented using admission control, a flow classifier or identifier and a packet scheduler.

Admission control makes the decision of whether a new flow can be granted the requested QoS without impacting earlier guarantees. Admission control may also be used for accounting, administration and authentication of resource reservations.

In order to make an admission control decision, the routers must measure the resource usage and based on the history data, make computations using the worst case bounds on the services, i.e. assess the impact of the new reservation. Another option is for the admission control to calculate the required resources based on the TSpec parameter characterization. However, how to deal with admission control demands is a local matter and is left as an open implementation issue for routers.

The classifier maps flows to appropriate classes. Inside a class packets of a flow receive the same treatment at the packet scheduler. A class may also be local to a particular router. Packets are often classified as flows based on the five-tuple or parts of it: source and destination IP address, protocol ID, and source and destination port. Another possibility would be a virtual circuit identifier, but it would require setup of virtual circuits as in ATM, or introducing a flow ID field to the header. The design of the process of classifying flows always amounts to a speed versus space tradeoff.

The router can select the route for the packet, forward or drop the packet, reorder or hold the packet. Dropping of packets and forwarding must be coordinated. Instead of the traditional tail-drop mechanism of dropping a packet when the router buffer is full, it would be more efficient to pick a packet to drop. As already mentioned, for loss tolerating real-time applications dropping may reduce delay, while for elastic traffic using for example TCP congestion control, a packet drop signals the source to reduce the sending rate and will require a retransmission of the packet.

One way to control dropping is to indicate through a mark which packets of a flow can be dropped in order to achieve the negotiated service level. The packets that can be dropped can be marked as pre-emptiable or expendable. Pre-emptiable packets are packets that are included in the admission control decision, but can be dropped in order to meet the service commitments. Expendable packets, on the other hand, are not included in the admission control decision, and are delivered only if the network can accommodate them and meet the service commitments.

#### 3.1.2.3 Resource reservations

For traffic control to function and for delay bounds to be met, resource reservations must be done before traffic can be transported by the network. A reservation setup protocol RSVP (resource reservation protocol) was created for IntServ to create and maintain the flow specific state, i.e. reservation along the path of the flow.

#### 3.1.2.4 Resource sharing

Fulfilment of QoS requirements is achieved per flow and through traffic control and resource reservations. Link sharing, on the other hand, is applied to aggregate entities.

Resource sharing has the aggregate bandwidth quantity as the primary interest. Sharing is done to meet a set of specified shares organized in a hierarchical tree order. The resource sharing is then implemented using packet schedulers given that the admission control keeps the cumulative guaranteed and predictive traffic from exceeding the assigned link share.

On the highest level there is *multi-entity link-sharing*. A link may be purchased and used jointly by a group of organizations. In case of overload, the entities wish to share the link according to predetermined shares. In the case of underload, any of the entities can utilize the idle bandwidth.

The next level is sharing between protocol families named *multi-protocol link-sharing*. During congestion, different protocol families react to congestion signals in a different way. By sharing the bandwidth it is guaranteed that flows from a protocol family that do not backoff during congestion do not hog the capacity from the protocols that backoff during congestion. When the link has idle bandwidth no predetermined sharing is needed.

Inside a protocol family *multi-service sharing* may be used. As an example inside IP there are shares between service classes, so that real-time traffic does not preempt elastic traffic.

Resource sharing is performed using packet schedulers. They determine how the network resources are allocated to individual flows by reordering appropriately the output queue. The FIFO scheduler used in the traditional Internet does not give sufficient guarantees to meet the delay bounds, therefore more sophisticated scheduling mechanisms are needed in IntServ. Fair queueing schedulers are able to provide both end-to-end bounds and link sharing. An example of a fair queuing scheduling mechanism is weighted fair queueing (WFQ) [PG93]. It is a generalized processor sharing scheduler with weights for different classes.

The priority queue orders the output stream according to priority while round robin or WFQ order according to the share of the link capacity given to the flow or class. An even more sophisticated reordering is according to the delivery deadline or the required output rate.

In hierarchical packet scheduling, for example, to provide controlled link sharing by limiting overload shares on a link, both priority queuing and WFQ or just WFQ at different levels can be used.

## 3.2 DiffServ

Differentiated Services is designed to achieve assured or relative service between flows without per flow scheduling. In general, this is done through assigning only a contracted or assured rate to the flows or aggregates. With this contracted rate a charge may be associated.

The main elements of DiffServ are traffic classification and conditioning at the boundary nodes and traffic forwarding through scheduling and discarding at the DiffServ interior nodes. In addition, congestion control mechanisms designed for the Internet, such as TCP, and active queue management algorithms, such as RED, may be used to achieve Quality of Service in the Internet.

The traffic classification and conditioning, i.e. the division of flows into per hop behaviors (PHB) and drop precedences inside PHBs, is done at the boundary nodes. After the packets of the flows have been classified, the traffic is conditioned, more specifically, traffic is metered and packets marked into drop precedence levels.

Inside a DiffServ node all traffic handling functions are performed on aggregates based on the given PHB. Inside each PHB, a number of precedence levels exist based on the marking done at the boundary node. In the scheduling unit two main elements affect the service experienced by the PHBs, the scheduling policy of the buffers and the realization and relationship between the discarding thresholds of the buffers.

The DiffServ proposals considered here are Expedited Forwarding (EF), Assured Forwarding (AF) and the Simple Integrated Media Access (SIMA) proposal. Following our distinction of assured services and relative services, we can categorize EF and AF to be of type assured services and SIMA offering relative services. AF can, however, due to its broad definition, be implemented to be of relative services type also. Relative services are easier to realize, in the sense that in overload situations the capacity has to be divided in shares without having to give the contracted rates to the flows. With assured services and conditioning only at the edge of the network, it may happen that at a core link the total capacity is less than the sum of assured rates, and then without relative differentiation, no differentiation may result.

EF and AF have the status of an IETF RFC and give a conceptual service model of the PHB groups that could be implemented. SIMA has not gained RFC status, but gives a more thorough implementation and end-to-end view on how to achieve service differentiation. The proposals are therefore treated in different length in the following sections.

### 3.2.1 EF

Expedited Forwarding (EF), as explained in RFC 2598 [JNP99], aims at providing a point to point or virtual leased line like connection offering a low loss, low latency, low jitter and assured bandwidth end-to-end service. Achieving such service requires that queueing at the nodes is minimal. For the buffers to be empty at the routers, the maximum arrival rate of an aggregate has to be conditioned so that it is less than the minimum departure rate.

Implementing EF requires that the nodes forward packets by at least a contracted rate. This can for example be implemented using priority queues, where EF traffic preempts other traffic. Other traffic with higher priority than EF cannot preempt the EF for more than a packet time at the contracted rate. On the other hand, the EF traffic must be conditioned or policed so that the aggregate arrival rate of packets is not more than the contracted rate and so that the EF traffic does not starve the lower priority traffic. Scheduling could also be implemented using WFQ with weights for EF in accordance with the contracted rate.

Amendments to RFC 2598 have been done in specifying the definition of EF. However, as the main idea has not been changed, they are not discussed in further detail here.

EF is of type assured services. It is an example of a very strict service, in the sense that anything above the contracted rate is normally not admitted into the network. Therefore traffic of EF type do not have the flexibility of sending more traffic at lower priority in times of low load. However, this is not the main aim of EF, as the use of EF is designed for a small set of flows requiring stringent delay bounds.

### 3.2.2 AF

Assured Forwarding (AF) outlined in RFC 2597 [HBWW99] is designed to give assured service to flows. It does not concern quantifiable delay or jitter requirements, but instead requires that a certain amount of forwarding resources are allocated to AF classes.

AF is a per hop behavior (PHB) group with four forwarding classes and three drop precedence levels. AF is designed to provide assured service to the traffic that is in profile, i.e. does not exceed the subscribed rate. Traffic may be conditioned to in profile or out of profile using a token bucket or leaky bucket mechanism. AF also requires that packets belonging to the same class are not reordered, for example, according to precedence level.

Different AF classes are used to offer different levels of assurance and each class is allocated a certain amount of buffer space and bandwidth. Inside an AF class packets are assigned a drop precedence, with the packets with the highest drop precedence discarded before the packets with lower drop precedence.

Each AF class should be serviced so that at worst it achieves its contracted rate. An AF class may receive more forwarding resources if excess resources are available from other AF classes or PHBs. The marking into precedence levels can be done using token buckets, e.g. a single rate three color marker [HG99a] or a two rate three color marker [HG99b].

In order to minimize long term congestion, active queue management to drop packets, e.g. Random Early Detection (RED), should be implemented at the buffers. Short term congestion can be handled by queueing packets.

AF is an example of an assured service, where a flow sending above its contracted rate may utilize the excess bandwidth of the network. However, as is the case with all assured services, the relationship between the assured rate and the actual received rate is not straightforward. As the flow is conditioned at the boundary, the total amount of traffic at an interior link may overload the link even though each flow is sending at its contracted rate. How the capacity should be divided in such cases is therefore not a trivial problem.

### 3.2.3 SIMA

The Simple Integrated Media Access (SIMA) proposal [Kil97] does not aim at assuring a certain contracted rate for each flow, but aims at assuring that the bandwidth is divided relative to the contracted rate.

SIMA thus concentrates on mechanisms to determine how network capacity should be divided during overload situations. The contracted rate, which influences both the charging and division of bandwidth, is named the nominal bit rate (NBR). The priority of a flow is determined relative to the NBR, seven priority levels are proposed in SIMA. If a flow is sending at its contracted rate, it has middle priority. If its sending rate is less than its NBR, it has higher priority, and conversely. Ideally, this means that in cases of overload, only the highest priority levels receive capacity in the network. Then each flow in the highest priority level divides the capacity in proportion to the NBR, but the absolute sending rate is only a fraction of the contracted rate.

The NBR is permanent, it has a charge associated with it, and is related to an organization (e.g. network interface), a user (e.g. IP address), or a flow (e.g. IP address and port number). The simplest approach is to assign an NBR to each interface, while the most useful approach in terms of performance is to have an NBR associated with each flow. In the following, we assume that the NBR entity is the flow.

The user can also make a distinction in service according to application or delay requirement by choosing to label the traffic as real-time (rt) or non-real-time (nrt) traffic. The real-time class is designated for flows requiring low delay and jitter. This is achieved by having small real-time buffers serviced prior to non-real-time buffers and favoring smooth traffic with small, i.e. less than 0.1 ms, traffic variations.

#### 3.2.3.1 Implementation example of SIMA

Figure 3.1 depicts the functional entities of SIMA.

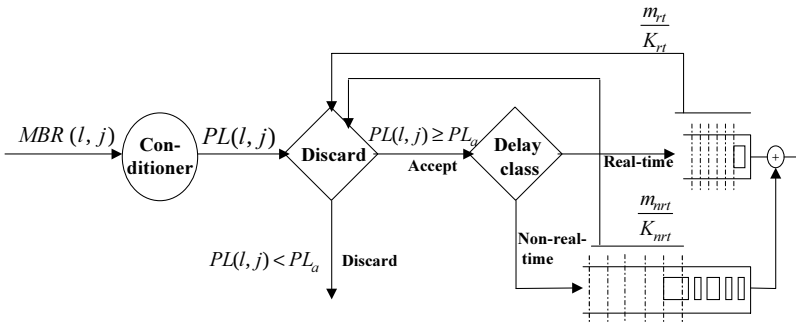



Figure 3.1: The SIMA specification

Once NBR is purchased and the delay class is chosen, the flow is sent to the network. The conditioner at the access node assigns priorities per flow, thus a corresponding priority level is associated with the packets sent. Note that the priority level of SIMA and the drop precedence of AF are conceptually the same thing, except that the highest priority corresponds to the lowest drop precedence.

The measurement of the sending rate, resulting in the momentary bit rate, is done by averaging the traffic sent by the flow. A proposed measuring principle is the exponential weighted moving average (EWMA), with different parameters for the rt and nrt traffic. Non-real time applications that can have variations in time scales of over 10 ms would not benefit from marking flows as real-time, as the bit rate measurements for real-time class traffic is more sensitive to traffic variations, giving thus worse priorities during peak rates.

Assuming fixed packet sizes, scaled to one, the momentary bit rate (MBR) of flow $l$ at time instant of the $j$:th packet is as given by [Kil97]

$$\begin{aligned} MBR(l,j) &= C\frac{\ln(1-\alpha)}{\ln(1-(\alpha/\rho(l,j)))} \\ \rho(l,j) &= \alpha + \rho(l,j-1)(1-\alpha)^{N(l,j)}. \end{aligned} \tag{3.1}$$

If $N(l,j) > 10/\alpha$, then

$$MBR(l,j) = \frac{C}{N(l,j)}.$$

Here $\alpha$ defines the time scale of the measurement, and depends on the buffer capacity $K_n$ of the delay class, e.g. $\alpha = 5/K_n$. Other variables are $N(l,j)$, the distance between the $j$:th and the $j-1$:th packets in time slots, $\rho(l,j)$ is the measured load generated by the $l$:th flow at the instant of the $j$:th packet and $C$ is the link capacity.

With the momentary bit rate determined, the conditioner at the access node assigns the priority level at the arrival of the $j$:th packet of $l$:th flow to

$$PL(l,j) = \max\left[\min\left[\left\lfloor 4.5 - \frac{\ln\frac{MBR(l,j)}{NBR(l)}}{\ln 2}\right\rfloor, 6\right], 0\right], \tag{3.2}$$

The above equation is derived so that when the ratio of MBR to NBR is one, the flow is given medium priority. As the MBR of a flow doubles (halves) the priority decreases (increases) by one unit.

Once the momentary bit rate of the flow is measured and the flow is assigned a priority level at the access node, the packets are marked with the priority level. In the core nodes, each packet is handled based on the DiffServ code point (DSCP) information storing the priority level and rt/nrt classification information. The delay class of the packet determines which of the two queues the packet is directed to in the scheduling unit.

Before this, however, there is a packet discarding system, which determines if the packet is admitted to the scheduling unit. The discarding is solely based on the priority level of the packet, the delay class does not affect the decision. The discarding system maintains an accepted level of priority, $PL_a$, calculated from the current buffer contents of both the scheduling unit's queues. Some of the possible expressions for calculating $PL_a$ based on the buffer contents $m_{rt}$ and $m_{nrt}$ of the real time and non-real-time queues, respectively, given in the proposal are,

$$\begin{aligned} PL_a &= a + b \cdot \left(\frac{m_{rt}}{K_{rt}} + \frac{m_{nrt}}{K_{nrt}}\right), \\ PL_a &= a + b \cdot \max\left(\frac{m_{rt}}{K_{rt}}, \frac{m_{nrt}}{K_{nrt}}\right), \\ PL_a &= a + b \cdot \sqrt{\left(\frac{m_{rt}}{K_{rt}}\right)^2 + \left(\frac{m_{nrt}}{K_{nrt}}\right)^2}, \end{aligned} \tag{3.3}$$

where $K_{nrt}$ and $K_{rt}$ are the sizes of the non real-time and real-time buffers, respectively, and $a$ and $b$ are constants to be determined by the implementors.

If the packet priority level is equal to or higher than $PL_a$, the packet is accepted to the scheduling unit and placed in the appropriate queue. Otherwise, the packet is discarded. In the case of a TCP flow, the packet discarding signals the TCP source that it should drop (halve) its sending rate, which in terms of the priority level would mean an increase by one, thus decreasing the probability of packets being discarded.

Note that the accepted level of priority, $PL_a$, is not a fixed value, but varies in time according to variations in the buffer contents and thus in response to congestion build up. This variation also affects the TCP mechanism through the loss probability feedback. Stability questions of the threshold mechanism have been considered in [Sch99].

## 3.3 DiffServ research

The AF and EF proposals having an IETF RFC status have been studied both analytically and through simulations. Studies on how to provide differentiated services in other ways have also been under consideration, with such proposals

as congestion pricing [GK99] and proportional delay differentiation [DSR99] to name but a few.

The DiffServ research reviewed below includes analytical and simulation studies of AF and simulation and test implementations of SIMA. In reviewing the previous work, we emphasize the difference between providing assured and relative services. In the analytical work, attention is paid to the loss process characterization, especially in the studies where TCP modelling is enhanced to include differentiated services. The last section gives a short review on the problems in providing the services and motivation for our modelling setup of SIMA and AF.

### 3.3.1 Analytical studies on AF

The papers by Sahu et al. [SNT+00] and Yeom and Reddy [YR01] are analytical studies on the relationship between the contracted rate and the transmission rate of TCP flows. The works include a model for TCP and a model for the priority marker, under the assumption of deterministic loss rates for different priorities, without an explicit buffer model for how the loss probabilities depend on the TCP rates.

Other analytical studies on differentiation include the work by May et al. [MBDM99] for assured and premium service. The model for assured service includes a buffer model with different discarding levels for different priorities, but does not include a TCP model nor a marking model for the traffic. The traffic is assumed to be from a Poisson process, with given probabilities for streams to be marked in profile or out of profile. The work thus concentrates on comparing different dropping schemes such as hard thresholds, pushout mechanism or RIO.

May et al. also considered a model for premium service. They studied the effect of having two buffers, where one is served with strict priority over the other. This can be though of as a model for Expedited Forwarding (EF) PHB class. Packets arriving with Poisson intensity are marked to priority levels, as in the simple case of one buffer, but are then directed to a queue depending on the marking. In-packets are serviced at a finite size $M/M/1/K$ queue and out-packets are directed to an $M/M/1$-queue. The out of profile packets receive service only when the queue for in profile packets is empty.

Another paper by Sahu et al. [STK99] studies a model similar to the one by May et al., with some considerations on modelling TCP flows. The papers [STK99], [SNT+00] and [YR01] are discussed in more detail below.

#### 3.3.1.1 Sahu et al. (2000)

The work by Sahu et al. [SNT+00] models how the token bucket marking of flows based on their assured rate affects the received rate. As this is a DiffServ study, the marking of the flows is performed at the edge of the network while aggregates are forwarded inside the DiffServ node. The paper makes the following

assumptions: two priorities, given loss probabilities for the two priority aggregates assuming that multiRED is used, and one single type of TCP flow at a time. The sending rate is thus derived as a function of the loss probabilities, with no explicit model of how the loss probabilities are affected by the sending rate. Furthermore, the derivation for the TCP sending rate is done assuming over and under subscription cases, but the equations are only derived assuming a given assured rate. Thus the differentiation is evaluated based on the relationship between the assured rate and the received rate. The system model does not provide insight into how the ratio of received rates depends on the share of assured rates of different types of TCP flows. The model and the results are therefore applicable only in determining how assured differentiated services are achieved.

The main result of the paper is that assured rate may not be achieved. If the assured rates are low, the link may be in under subscription state and the received rate is larger than the assured rate. Similarly, if the assured rates are too high, the link may be in over subscription state and less than the assured rate is received.

More specifically, the TCP process under consideration is of type Reno and is modelled as a renewal process, where the window is either reduced to half upon a triple duplicate ACK or to one upon a timeout. The sending rate of this TCP flow is then determined for two cases: under subscription and over subscription. In the under subscription case, only the lower priority traffic is discarded. In the over subscribed case, the lower priority traffic is discarded with probability one, and the higher priority traffic with a positive probability.

The result for the over subscribed case illustrates the problem of dividing capacity according to assured rate. The received rate $r$ in terms of assured rate $A$, bucket size $B$, round trip time $T$ and loss probability $p_1$ is

$$r = \min\left(A, \frac{3(A + \sqrt{2B}/T)}{4}, \frac{1}{T}\sqrt{\frac{3}{2p_1}}\right).$$

The result shows that under an assured services discipline a TCP flow never receives more than its assured rate. Furthermore, when the received rate is below the assured rate, the received rate can be that of a TCP flow under no differentiation mechanism.

In the under subscription case, the result is not so easy to interpret, but if the assured rate is small enough, the extra capacity of the link is divided equally according to the TCP equations. Thus the TCP flow receives its assured rate plus an equal share of the link capacity.

The reason for this kind of division is in the token bucket marking scheme and in using two priorities. If the bottleneck link is congested enough not to be able to provide the assured rates of the flows, the capacity is divided as if no marking was used. More priorities give the possibility of having many assured rates, thus introducing more leverage into the differentiation.

#### 3.3.1.2 Yeom and Reddy

The paper by Yeom and Reddy [YR01] gives a better model for assured services. They model the interaction of many TCP flows and have two or three priority levels in their model, but do not study the effect of the token bucket size on the bandwidth share.

The authors make an interesting claim saying that it is the congestion avoidance technique in TCP that makes it difficult for flows to achieve their performance goal. In view of the study by Sahu et al. the claim refers to the case when the congestion avoidance in TCP reduces the sending rate below the assured rate, as then no differentiation is achieved. They also propose, and the models are based on, a two window TCP, where the congestion window equals the sum of the reserved window and the excess window.

The system model is again one with token bucket marking and given loss rates for the different priorities assuming that RED is used. The division of the model into under and over subscription cases was first proposed by these authors.

The results are quite the same as given by Sahu et al., with the exception that the received rate is not given in terms of token bucket size, only in terms of assured rate. In cases of over subscription, the assured rate may not be reached and the capacity is divided equally, with the received rate being the rate for a TCP flow when no differentiation is present. Even in the under subscription case, those flows with small assured rates may have a received rate almost equal to a flow with a higher assured rate.

Under subscription case is defined as the case when only out of profile packets are dropped from the network. However, in the case of many TCP flows with different assured rates, the network would be over subscribed if all flows had the highest assured rate. Then those flows with the highest assured rate are only able to receive their assured rate up to the threshold $A = \frac{3}{T}\sqrt{\frac{2}{p_{out}}}$, corresponding to $\sqrt{3}$ times the best-effort share, when no reservations are used. Below this assured rate, the TCP flows receive their assured rate and some excess rate.

Note that, as in both the models by Sahu et al. and Yeom and Reddy, the loss probabilities were assumed to be given, and it is not clear what mix of assured rates results in an under subscription case, e.g. if all flows have an assured rate of $A = \frac{3}{T}\sqrt{\frac{2}{p_{out}}}$, they will not receive this rate, as the maximum rate is given by the best-effort rate of $r = \frac{1}{T}\sqrt{\frac{3}{2p}}$.

#### 3.3.1.3 Sahu et al. (1999)

In an earlier paper by Sahu et al. [STK99], the authors study both a model related to that of May et al. [MBDM99] and a model for TCP under different dropping probabilities for in and out of profile traffic.

In the first part of the paper threshold dropping in one buffer is compared to priority queueing between two buffers. Thus the models also assume that traffic is divided in predefined shares to two priority classes and these priority classes are either serviced by one buffer with two thresholds or by two buffers with tail dropping. The results of the paper show that priority scheduling is needed to support delay sensitive applications.

However, the paper also gives a TCP model similar to [SNT+00] under the assumption that two levels of dropping probabilities exist in the network. The results of the paper are similar to [SNT+00].

Though the two models are presented in the same paper, the paper still lacks combining both a TCP model and a buffer model. Thus it does not take into account how the sending rate of TCP flows affects the dropping probabilities and vice versa.

### 3.3.2 Simulation studies on AF

The analytical studies demonstrated the relationship between assured rate and received rate of TCP flows. Simulation studies have studied the effect of having both TCP and UDP flows in the network.

#### 3.3.2.1 Goyal et al.

The simulation study by Goyal et al. [GDJL] study different factors related to differentiation, namely related to AF. These include: number of priority levels, percentage of highest priority traffic, buffer management and traffic types. The simulations assume many TCP flows, but a fixed number of flows. The system model is similar to the ones of assured services presented in the analytical studies. Based on assured rates, two cascaded token bucket markers mark the packets to three priorities: green, yellow or red.

As a result they show the importance of three priority levels in distinguishing between congestion sensitive and insensitive flows. The marking into priorities is done so that in the case of over subscription of the network the priorities are redundant, and thus the case of over subscription is not studied. This is always the case when assured services are used, as highest priority is given to a flow conforming to assured rate. Differentiation would occur in the oversubscribed case if middle priority would be given to a flow conforming to its assured rate. Then a flow would reach higher priorities by dropping the rate below the assured rate.

An interesting conclusion of the paper is to mark excess insensitive flows, e.g. UDP traffic to lowest priority, while excess congestion sensitive flows, e.g. TCP traffic receives middle priority. Thus UDP traffic is given less priority levels than TCP traffic.

Elloumi et al. [ECP] give similar results, showing that TCP flows can be protected from UDP flows by marking out of profile UDP flows to lowest priority, but out of profile TCP flows to middle priority.

#### 3.3.2.2 Pieda et al.

Pieda et al. [PSN99] study the claim made by Goyal et al. and Elloumi et al. that TCP flows can be protected from UDP flows by marking out of profile UDP flows to lowest priority, but out of profile TCP flows to middle priority. They divide the study of priority levels into six scenarios. In each scenario three priority levels are used, but packets of a flow are only marked in or out of profile. They also compare the effect of using RED with overlapping discarding thresholds and different dropping probabilities compared to using RED with non-overlapping thresholds and different dropping probabilities.

As a result they show that when TCP and UDP are marked to the same AF class, three priorities are not enough to achieve the three targets: that in an over provisioned network both UDP and TCP target rates are achieved, that UDP and TCP out of profile packets should have a reasonable share of excess bandwidth, and that in an under provisioned network TCP and UDP flows experience degradation in proportion to the assured rate.

The scenario where TCP in profile packets have highest priority, UDP in profile packets have middle priority, and all out of profile packets have lowest priority is able to guarantee assured rates, and the relative deviation from assured rate is the smallest. However none of the scenarios is able to meet all three differentiation targets.

### 3.3.3 Studies on SIMA

SIMA with TCP flows has been studied with the help of simulations in [KR98] and using a test network in [LSLH00] and [HKL+00]. All studies show that SIMA is able to achieve differentiation and that different flows with ascending contracted rates (NBRs) receive different and ascending quality. The authors of [LSLH00] and [HKL+00] also demonstrate that the discarding and measuring mechanisms of SIMA are easy to implement.

### 3.3.4 Further research on SIMA and AF

The simulation and analytical studies on AF showed that when assured services are used, it is difficult, even impossible, to guarantee division of excess or division of limited capacity in ratio of assured rate or weight. Furthermore, Pieda et al. concluded that the best way of protecting UDP and TCP traffic from each other is separating them into different AF classes.

In the SIMA proposal, two delay classes are proposed and the assured or nominal bit rate is specified so that relative services could be achieved. Given the results of the earlier SIMA research it is thus valid to develop an analytical model of SIMA to compare the results on AF studies in achieving assured services to the SIMA proposal aiming at guaranteeing relative services.

Furthermore, the analytical models for AF lacked combining a TCP model and a packet buffer model to explicitly model the loss process and its effect on TCP and differentiation. Thus a more comprehensive model of the interaction between differentiation mechanisms and TCP is needed.

The analytical and simulation studies presented in the following chapters will quantify how the received quality depends on the contracted rate and compare the SIMA differentiation to the one achieved by AF.

# Chapter 4

# Underlying mechanisms in differentiating traffic

The purpose of the subsequent chapters is to identify how delay and priority aggregates can be used to achieve relative differentiation of flows. To this end we formulate a generic DiffServ architecture based on the proposals considered in the previous chapter, i.e. EF, AF, and SIMA. From these proposals, we identify the main QoS mechanisms used to achieve differentiation.

All the mechanisms are based on assigning a contracted or assured rate to the flows. With this contracted rate a charge may be associated. Based on the assured rate of the flow it is marked to a priority level. If the service model is of type assured services, then as long as the flow sends at a rate less than the assured rate, the flow is marked in profile and thus only two or three priority levels are needed. If the service model is of type relative services, then the priority can be determined based on the ratio of the sending rate to the assured rate. As will be shown later, it is reasonable to have many priorities. The more priority levels, the more flexible the marking is in terms of covering a wider range of sending rates. The main difference between these service concepts is the division of excess capacity and the division of capacity in overload situations.

We divide the DiffServ mechanisms into two categories: classification and conditioning of flows at the boundary node and forwarding of flow aggregates through packet level mechanisms inside the DiffServ node.

Our aim is to model both the TCP sources and the network buffers to study how the sending rate of TCP flows affect the dropping probabilities of the network and vice versa. Our models also include the effect of UDP flows in the network. We then evaluate the differentiation mechanisms based on the relationship between the contracted rate, which will be called the weight, of the flow and the share of bandwidth achieved by the flow belonging to a specific aggregate.

Figure 4.1 summarizes the components, each discussed separately below.

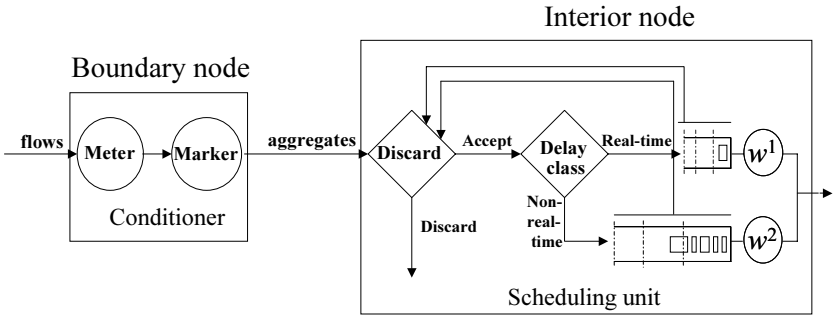



Figure 4.1: Components of a DiffServ network

## 4.1 Network model

Consider a DiffServ network with a single bottleneck link, which is loaded by a fixed number of flows. Assume two delay classes, $d = 1, 2$, and $I$ priority levels, $i = 1, \ldots, I$. Level $I$ refers to the highest priority, i.e. flows at that level encounter the smallest packet loss probability, and level 1 to the lowest priority. Note that this is just opposite to, e.g., the definition of precedence level given in [HBWW99]. Therefore, we rather use the term priority level here. Each flow is given a weight $\phi$ that reflects the value of the flow or the contracted rate of the flow.

Assume that the flows are grouped according to the weight $\phi$. There are $L^1$ different groups of flows of delay class 1, each group $l$ with a characteristic packet sending rate $\nu(l)$ and weight $\phi(l)$. Let $\mathcal{L}^1$ denote the set of such flow groups. Furthermore, assume that there are $L^2$ different groups of flows of delay class 2, each group $l$ with a characteristic packet sending rate $\nu(l)$ and weight $\phi(l)$. Let $\mathcal{L}^2$ denote the set of such flow groups. Finally, let $n(l)$ denote the number of flows in any group $l$.

Note that we have chosen to make the restrictive assumption of a single bottleneck link and a fixed number of flows at this point, although the mechanisms and their models presented in this chapter can be generalized to any network with an arbitrary number of flows. However, as solving the models, which will be done in the chapters to follow, require these assumptions, we have chosen to state them already in this chapter. The extension of the models for general topologies and number of flows is left for further research.

## 4.2 Flow level mechanisms at the boundary nodes

At the conditioner, the packets of a flow are marked and aggregated to priority levels. We adopt the proposal in [Kil97], where the priority level $pr$ of the flow

depends on $\nu$ and $\phi$ as follows:

$$pr = \max\left[\min\left[\left\lfloor I/2 + 0.5 - \log_2 \frac{\nu}{\phi}\right\rfloor, I\right], 1\right]. \tag{4.1}$$

Thus, the priority level is decreased by one as the traffic rate doubles. Note also that we adopt here the relative services approach by defining that a flow sending at its contracted rate receives the middle priority and not the highest priority.

### 4.2.1 Metering and marking mechanisms

Consider two **metering and marking alternatives** to mark flows to priority levels :

- *Token bucket*: Packets are marked in-profile if the bucket holds enough tokens upon arrival and out-of-profile otherwise.
- *Exponential weighted moving average*: The measured bit rate of previous time instants are exponentially dampened according to a time parameter $\alpha$ and the time interval that has passed since the measurement was done.

The token bucket or leaky bucket principle is a popular metering principle, referred to, e.g. in the AF specification [HG99a] and [HG99b]. Arriving packets use up the tokens in the token bucket, which has capacity $c$ and is filled with tokens at a constant rate of $r$. If the bucket holds enough tokens to match the bit size of the arriving packet the packet is marked to the highest priority level, otherwise the packet has low priority.

For three priority levels the metering and marking may be performed with two token buckets for each group $l$, with rates $r(l,1) > r(l,2)$ and capacities $c$, as shown in figure 4.2. If the first bucket does not have enough tokens at the arrival of a packet, the packet is out-of-profile and is marked to the lowest priority level; in other cases the state of the second bucket determines the priority level. If the second bucket does not have enough tokens at the arrival of the packet, the packet is out-of-profile and is marked to middle priority, and if there are enough tokens both in the first and second bucket, i.e. the packet is in-profile for both buckets, the packet is marked to the highest priority level. Note that a packet uses up the tokens in all the buckets where it is in-profile.

Exponential weighted moving average (EWMA) is another traditional metering principle. Its use for metering flow arrival rates was proposed, for example, in the SIMA specification. The measurement results of previous time instants are taken into account, but exponentially dampened according to a time parameter $\alpha$ and the time interval that has elapsed since the measurement was done. The marking is then performed based on predefined thresholds on the resulting measured arrival rate. Parameter $\alpha$ describes the time scale of the bit rate measurement.

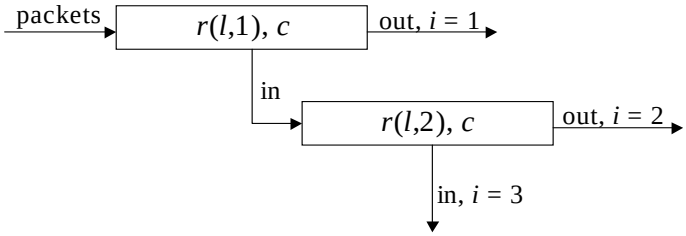



Figure 4.2: Token bucket scheme for marking packets to three priority levels

In [Kil97], a discrete version of EWMA is used. The measured bit rate of a flow $k$ at the moment of transmission of the $j$:th packet is

$$mbr(k,j) = C\frac{\ln(1-\alpha)}{\ln(1-(\alpha/\rho(k,j)))}, \tag{4.2}$$

$$\rho(k,j) = \alpha + \rho(l,j-1)(1-\alpha)^{N(k,j)}. \tag{4.3}$$

If $N(k,j) > 10/\alpha$, then

$$mbr(k,j) = \frac{C}{N(k,j)},$$

where $N_{k,j}$ is the distance between the $j$:th and $j-1$:th packet in time slots and $C$ is the capacity of the link.

Note that this result can be derived from the traditional discrete EWMA updated at fixed intervals of length $1/\delta$ time slots, with time scale $\alpha/\delta$ and assuming that $C = 1$,

$$\rho(k,j) = \alpha/\delta + \rho(k,j-1)(1-\alpha/\delta).$$

When the update is done only each time a packet arrives the discrete update becomes

$$\rho(k,j) = \alpha/\delta + \rho(k,j-1)(1-\alpha/\delta)^{\delta N(k,j)}.$$

Thus the rate per intervals of length 1 is

$$\rho_\delta(k,j) = \alpha + \rho_\delta(k,j-1)(1-\alpha/\delta)^{\delta N(k,j)}.$$

Which gives the continuous update as $\delta \to \infty$

$$\rho(k,j) = \alpha + \rho(k,j-1)e^{-\alpha N(k,j)}.$$

The first order approximation of the above is then

$$\rho(k,j) = \alpha + \rho(k,j-1)(1-\alpha)^{N(k,j)}.$$

The momentary bit rate assuming stationarity, i.e. $N(k,j) = N$ and $\rho(k,j) = \rho(k,j-1) = \rho$, can then be solved from the above equation.

$$\begin{aligned} \rho &= \alpha + \rho(1-\alpha)^N \\ 1/N &= \frac{\ln(1-\alpha)}{\ln(1-(\alpha/\rho))} \\ mbr(k,j) &= \frac{\ln(1-\alpha)}{\ln(1-(\alpha/\rho(k,j)))}. \end{aligned}$$

The priorities are determined based on thresholds. We allow the thresholds to depend on the group $l$, which the flow $k$ belongs to, as follows,

$$\begin{aligned} t(l,0) &= \infty, \\ t(l,i) &= \phi(l)a(i),\ i = 1, ..., I-1, \\ t(l,I) &= 0. \end{aligned}$$

The function $a(i)$ could, for example, be defined as in SIMA (equation (4.1)),

$$a(i) = 2^{I/2-i-0.5}.$$

Note that $a(i-1)/a(i) = 2$ for all $i$. The $j$:th packet of flow $k \in l$ has priority $i$, if

$$t(l,i) \leq mbr(k,j) < t(l,i-1).$$

The metering principles and their time parameters $\alpha$ and $c$ have an effect on the resulting differentiation, as both the memory or time span of the measurement as well as the dampening of bursts affect the way traffic is divided into priority levels. Note, furthermore that $\alpha$ and $c$ may depend on the forwarding class, i.e. the delay class of the flow.

### 4.2.2 Metering and marking model

Assume that a metering principle exists. The resulting traffic intensity then determines which priority level the packets are marked to. We can model the resulting two **marking alternatives** as follows:

- *Per packet marking:* Only those packets of a flow that exceed the marking threshold are marked to the lower priority level.
- *Per flow marking:* Once the measured load of a flow exceeds a marking threshold, all packets of the flow are marked to the same priority level.

As an example, consider the case of three priority levels and thresholds $0 = t(l,3) < t(l,2) < t(l,1)$. Denote by $\nu$ the bit rate of flow $k \in l$. With *per packet* marking, the ratio of packets

$$\frac{\min[\nu, t(l,i-1)] - \min[\nu, t(l,i)]}{\nu} \tag{4.4}$$

have priority $i$. With *per flow* marking, all packets of the flow have the same priority, corresponding to

$$pr = \arg\min_i [\nu \geq t(l,i)]. \tag{4.5}$$

Independent of the marking scheme we say that a flow has priority $i = pr$ as given by equations (4.1) and (4.5).

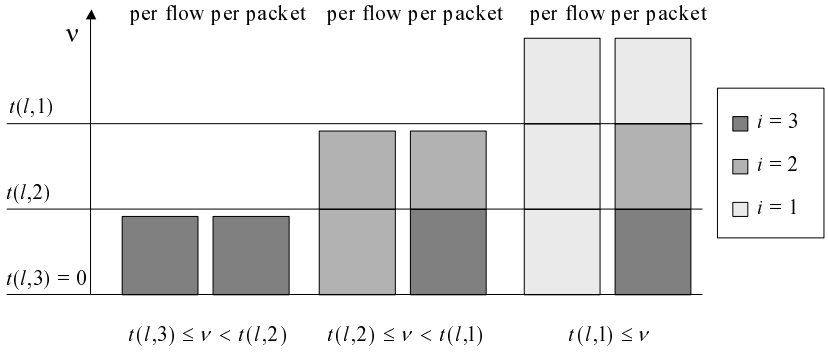



Figure 4.3: Differences in marking, for three priority levels

Figure 4.3 depicts the resulting marks given to the packets of the flow. The token bucket mechanism can be modelled as a *per packet* marking method, while EWMA results in *per flow* marking. This conjecture will be used in the chapters to follow and shown to hold through simulations in chapter 7.

An intuitive proof can be given assuming a CBR stream of packets and assuming that all the rate and threshold values are rational numbers. Packets are marked in cycles as a CBR stream passes cascaded token buckets. Consider, for example, a flow with reference rate $\phi(l) = 1/2$ and sending rate $\nu = 1$. If $I = 3$, then the threshold rates are $t(l,2) = 1/4$ and $t(l,1) = 1/2$. Assume further that the bucket size corresponds to the constant packet size. Then the sequence of packet marks will be as follows: $3, 1, 2, 1, 3, 1, 2, 1, \ldots$. In this case, the length of the cycle is 4, and the proportions of packets with marks $i = 1, 2, 3$ are

$$1/2 = \nu - t(l,1), \quad 1/4 = t(l,1) - t(l,2), \quad 1/4 = t(l,2),$$

respectively. Cascaded token buckets split the flow into sub-streams $i = pr, pr + 1, \ldots, I$ with rates

$$\nu(i) = \min[\nu, t(l,i-1)] - \min[\nu, t(l,i)].$$

All the packets of sub-stream $i$ have the same mark $i$ corresponding to the ratio given in equation (4.4).

Using EWMA marking, the measured rate will approach the constant sending rate $\nu$. Assuming that this convergence happens in a negligible time interval, which corresponds to the assumption that the time constant in averaging is small compared to the lifetime of the flow, we conclude that the measured rate equals the sending rate $\nu$ in each measurement. All the packets of the flow have the same mark $pr$ given by equation (4.5).

## 4.3 Packet level mechanisms inside the DiffServ nodes

If per flow queuing could be possible, bandwidth would be divided between flows according to the weights assigned to each flow. In DiffServ the forwarding is done

to aggregates divided, in our case, into two classes based on the delay requirements and into $I$ priority levels based on the weight of the flow.

Bandwidth between delay aggregates must be divided in terms of delay requirements, with emphasis on protecting streaming traffic from elastic traffic and vice versa. The class that requires low delays, i.e. streaming traffic, should be given enough capacity and short buffers in order for the queuing delays to stay low.

Bandwidth between priority aggregates must be divided in terms of packet loss probabilities. It should result in a division according to flow weights assuming that the relative services approach is adopted. Furthermore, this must be done across delay classes. Low latency classes should not starve bandwidth from the other classes and the elastic delay aggregate with high priority should not be discarded before the low latency aggregate with lower priority. Discarding of packets should be based on the priority levels and corresponding buffer thresholds.

### 4.3.1 Discarding

We have a system with two delay classes, serviced by two separate buffers, where the buffer sizes are chosen according to the delay requirements of the delay aggregates. Both buffers have $I$ discarding thresholds, one for each priority class.

Consider two different **discarding mechanisms**:

- *Independent discarding:* Each buffer acts locally as a separate buffer, discarding appropriate priority levels according to its buffer content.
- *Dependent discarding:* The content of both buffers determines which priority level is discarded, in both buffers.

Let $m^d$ denote the number of packets in the buffer of delay class $d$. The *independent discarding* is implemented by giving, separately for each delay class $d$, thresholds $K^d(i)$, $K^d(I) = K^d$ . The minimum priority level accepted is then $pr_a^d = f_d(\frac{m^d}{K^d})$. The dependent discarding, proposed in [Kil97], is implemented by giving a two-dimensional monotonic function

$$pr_a = f(\frac{m^1}{K^1}, \frac{m^2}{K^2}) \tag{4.6}$$

that determines the minimum priority level accepted when in state $(m^1, m^2)$.

The *dependent discarding*, introduced in the SIMA proposal, can be implemented using different discarding functions

$$pr_a = a + b \cdot \Big(\frac{m^1}{K^1} + \frac{m^2}{K^2}\Big), \tag{4.7}$$

$$pr_a = a + b \cdot \max\Big(\frac{m^1}{K^1}, \frac{m^2}{K^2}\Big),$$

$$pr_a = a + b \cdot \sqrt{\Big(\frac{m^1}{K^1}\Big)^2 + \Big(\frac{m^2}{K^2}\Big)^2}. \tag{4.8}$$

We use equation (4.8) as a basis for the discarding function. Note that equation (4.7) is similar to the discarding in a system with only one buffer.

These two mechanisms differ most when one buffer is heavily loaded and the other is almost empty. *Independent discarding* would discard traffic belonging to different priority levels in different buffers, while *dependent discarding* would discard traffic of high priority also from the buffer that is almost empty.

Figure 4.4 shows the proposed discarding functions under two possible choices of the constants $a$ and $b$. The figure on the left has

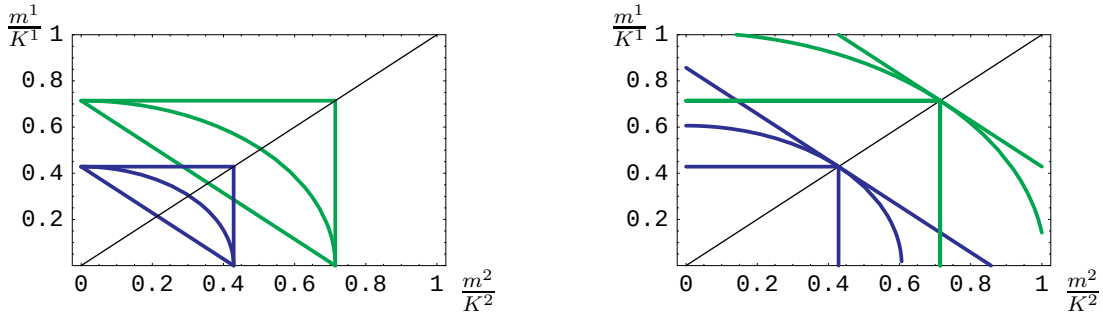
$$\frac{K^1(i)}{K^1} = \frac{K^2(i)}{K^2},$$

i.e. the mappings fixed to work identically, when either buffer is empty. The second proposed mapping, the maximum function, would then be reasonable in the two buffer setting. However, the first and third mapping would in this case result in mapping over half of the area of the state space to service only the highest priority level, resulting in under utilization of the buffers.

In figure 4.4, the figure on the right shows how the areas of priority level acceptance are kept approximately similar with each function by fixing the functions to be identical in the case that both buffers are equally loaded, that is

$$\frac{m^1}{K^1} = \frac{m^2}{K^2}.$$

Then the third mapping, square root of the sum of buffer contents is in terms of threshold areas in between the linear and maximum functions.




Mappings fixed to $\frac{K^1(i)}{K^1} = \frac{K^2(i)}{K^2}$

Mappings fixed to line $\frac{m^1}{K^1} = \frac{m^2}{K^2}$

Figure 4.4: Examples of discarding functions for three priority levels

### 4.3.2 Scheduling

The traffic that is not discarded is placed in either one of the two buffers. The scheduling principle affects the division of the link capacity, and can thus be

used to divide resources between delay aggregates. Furthermore where discarding affects the packet loss of the flow, scheduling affects the delay experienced by the flow.

We restrict our analysis of scheduling mechanisms to the different weights possible in the Weighted Fair Queuing (WFQ) scheduling principle. Whenever one of the buffers is empty, the other buffer has use of total link capacity. Otherwise the capacity of the link is divided according to predetermined weights $w^1$ and $w^2$, with $w^1 + w^2 = 1$.

We consider three different **scheduling scenarios**:

- *Priority queuing:* WFQ with weights $(w^1 = 1, w^2 = 0)$.
- *Unequal sharing:* WFQ with weights $(w^1 = 0.75, w^2 = 0.25)$.
- *Equal sharing:* WFQ, with weights $(w^1 = w^2 = 0.5)$.

# Chapter 5

# Flow level differentiation model for greedy TCP flows

Let us consider the network model introduced in section 4.1. We consider a single link network, with the link capacity scaled to one. The network is used by greedy TCP sources that tend to maximize the minimum bandwidth allocation. We know from the best-effort flow level studies presented in section 2.4.2 that in such a setting and with no priorities ($I = 1$) it is optimal for greedy TCP flows to divide bandwidth equally among themselves.

Let us now study the fairness of such greedy TCP flows when a priority mechanism based on relative services is added to the network. A flow with weight $\phi$ is classified to priority level $i \in \mathcal{I} = \{1, ..., I\}$. A natural objective of any traffic control algorithm is to allocate bandwidth as fairly as possible. Here fairness refers to weighted fairness or, equivalently, to relative services in a single link, i.e. achieved bit rate $\beta$ of any flow should be proportional to its weight $\phi$. Our goal is to study what kind of weighted fairness is achieved in our example network between TCP flows.

## 5.1 Marking model for TCP flows

Assume that the flows are divided to the priority levels based on the metering and marking mechanism presented in section 4.2.1, i.e. flows are conditioned to priority levels depending on the ratio of their sending rate to their weight $\phi$. The more traffic a flow sends, the worse the overall priority.

Assume only one delay class, and set $d = 2$ for elastic TCP flows. We have $n$ flows divided into groups $l \in \mathcal{L}^2 = \{1, ..., L^2\}$ according to the weight $\phi(l)$. Each group consists of $n(l)$ identical flows. Flows are classified to priority levels using either *per packet* or *per flow* marking. Denote here the actual bit rate of a flow in group $l$ by $\beta(l)$.

In such a setting the TCP flows tend to maximize the sending rate, but the condi-

tioner may penalize an increase in sending rate by a drop in priority level. Flows in lower priorities may then have a small share of bandwidth compared to their high sending rate. The TCP mechanism would then react and lower the sending rate until equilibrium rate $\beta$ is reached. Thus the interaction between TCP and DiffServ traffic conditioning makes the flows to maximize their bandwidth share individually.

We aim at studying to what magnitude the weight of flows, determining the priority level, affect the bandwidth allocation of greedy TCP sources under this individual optimization principle.

Following the ideal TCP model, we assume that in the same priority level bandwidth is divided equally between flows. In addition, we model priority as strict priority, where a higher priority level has strict priority over the lower class. Note that this does not cause a flow of the higher class to use up all the bandwidth and starve the flows of the lower class, as the assignment to priority level depends both on the contracted rate $\phi(l)$ and the bit rate $\beta(l)$ of the flow $l$.

The evaluation of fairness is made based on the resulting bandwidth allocation and division into priority levels. Note that, we consider differentiation as a function of the number of flows, opposed to as the function of the load of the network. This is due to the fact that we are considering elastic sources that adjust their sending rate, and thus adjust the load, according to the congestion of the network.

Consider two flow groups, $L^2 = 2$ that have a choice between $I$ priority levels. Group 2 flows have $k$ times more weight than group 1 flows

$$\phi(2) = k \cdot \phi(1).$$

In section 4.2.1 we defined the common thresholds of the marking schemes for the priority levels as

$$t(l, i) = \phi(l) \cdot a(i), \ i = 1, ..., I - 1, \tag{5.1}$$

with $t(l, 0) = \infty$ and $t(l, I) = 0$ and

$$a(i) = 2^{(-i+I/2-0.5)}.$$

The flows in group $l = 2$ are allowed to send $k$ times more traffic until classified to the same priority level as the flows in group $l = 1$, since the boundary rate for them is

$$t(2, i) = \phi(2) \cdot a(i) = k \cdot \phi(1) \cdot a(i) = k \cdot t(1, i).$$

The resulting optimal bandwidth allocation depends on the marking scheme used, as the marking scheme determines the overall priority of a flow.

### 5.1.1 Per flow marking

In *per flow* marking, all packets of a flow are marked to lower priority, once the corresponding threshold is exceeded. Flows in group $l$ are in priority level $i$ if

$$t(l,i) < \beta(l) \leq t(l,i-1).$$

Given that $t(1,1) = b$, we have $t(2,1) = kb$, and the resulting priority levels for the two flow groups, when $I = 2$, are given in table 5.1.

| | $l=1$ | $l=2$ |
|---|---|---|
| $i=1$ | $\beta(1) \geq b$ | $\beta(2) \geq kb$ |
| $i=2$ | $\beta(1) < b$ | $\beta(2) < kb$ |

Table 5.1: Priority level assignment conditions for $I = 2$

As a result of the above relation, if a flow in the higher priority level increases its bit rate above the boundary $b$ or $kb$, it falls to the lower priority level, and divides the remaining capacity equally with the other flows in the lower class. The coupling of sending rate and assignment of priority level thus prevents starvation of bandwidth typical to normal priority mechanisms. The weight of the flow, on the other hand, gives the flows the right to control the boundary at which point the priority level changes.

#### 5.1.1.1 Bandwidth allocation model

Let us look closer at the resulting bandwidth shares of the flows. Under *per flow* marking, flows from the two groups, $l = 1$ or $l = 2$, may be in the same priority level or they may be in different priority levels.

Define $\beta(l,i)$ as the actual bit rate of a flow in group $l$ and in priority level $i$ and $n(l,i)$ as the number of flows in the corresponding state, with

$$n_i = \sum_{\mathcal{L}} n(l,i),\ i \in \mathcal{I}.$$

In general the strict priority means that flows in the highest priority level divide the bandwidth equally, but only up to their boundary rate $t(l, I-1)$. Flows in the next priority level divide equally, among themselves, the remaining bandwidth up to their boundary rate $t(l, I-2)$, and as long there is bandwidth left subsequent priority levels divide the bandwidth left over from higher priority levels, with the maximum share determined by the threshold $t(l, i-1)$.

However, as the flows have different boundary rates, the bandwidth share for flows in group 1 in priority $I$ is

$$\beta(1, I) = \min\Big(\frac{1}{n_I}, t(1, I-1)\Big),$$

while the bandwidth share for flows in group 2 is

$$\beta(2, I) = \min\Big(\max\big(\frac{1}{n_I}, \frac{1 - n(1, I)t(1, I-1)}{n(2, I)}\big), t(2, I-1)\Big).$$

The remaining capacity is then

$$C_{I-1} = \max\Big(0, 1 - n(1, I)t(1, I-1) - n(2, I)t(2, I-1)\Big).$$

Thus, if after the higher priority flows have reached their bandwidth thresholds there is still capacity left, the subsequent priority levels divide the capacity in the same fashion:

$$\beta(1, i) = \min\Big(\frac{C_i}{n_i}, t(1, i-1)\Big) \tag{5.2}$$

$$\beta(2, i) = \min\Big(\max\big(\frac{C_i}{n_i}, \frac{C_i - n(1, i)t(1, i-1)}{n(2, i)}\big), t(2, i-1)\Big). \tag{5.3}$$

The capacity remaining for priority level $i$ is

$$C_i = \max\Big(0, C_{i+1} - n(1, i+1)t(1, i) - n(2, i+1)t(2, i)\Big),$$

assuming that $C_I = 1$.

All flows in the lowest priority receive the same share

$$\beta(1, 1) = \beta(2, 1) = \frac{C_1}{n(1)},$$

as $t(l, 0) = \infty$. Thus at the lowest priority level no differentiation occurs.

#### 5.1.1.2 Optimal bandwidth allocations $I = 2$

Assume now that $I = 2$ and that all the flows inside a flow group behave in the same manner, i.e. inside a group all flows belong to the same priority level. Therefore we have $n(l, i) = n(l)$ when the flows are in priority level $i$ and zero otherwise. From this it follows that the system may be in exactly four different states.

The optimal allocations follow then directly from the equations (5.2) and (5.3) under the condition that all flows of a group are in the same priority level. For the four different states possible for the link, the actual bit rates $\beta(l)$ received by the flows in group $l$ are shown in table 5.2.

| | $l=1$ | |
|---|---|---|
| $l=2$ | $i=1$ | $i=2$ |
| $i=1$ | $\beta(1)=\frac{1}{n(1)+n(2)}$<br>$\beta(2)=\frac{1}{n(1)+n(2)}$ | $\beta(1)=\min(\frac{1}{n(1)},b)$<br>$\beta(2)=\max(\frac{1-n(1)b}{n(2)},0)$ |
| $i=2$ | $\beta(1)=\max(\frac{1-n(2)kb}{n(1)},0)$<br>$\beta(2)=\min(\frac{1}{n(2)},kb)$ | $\beta(1)=\min(\frac{1}{n(1)+n(2)},b)$<br>$\beta(2)=\min(\max(\frac{1}{n(1)+n(2)},\frac{1-n(1)b}{n(2)}),kb)$ |

Table 5.2: Bandwidth allocation in the four possible network states, *per flow* marking, $I=2$

Assume now that the individual optimization principle of TCP flows holds, i.e. the flows optimize the bandwidth share measured by actual bit rate $\beta$ of the flow and not the sending rate. The following scenarios can be observed as a function of the number of flows $n(1)$ and $n(2)$.

1. In times of low load, when the following condition holds

$$n(1)kb+n(2)kb<1,$$

   there is no advantage in being in the higher priority level, where the bit rate would be limited. For both groups, the bit rate achieved in the lower priority level, $\beta(l)>kb$ for all $l$. This is more than the bit rate in the higher priority level. Thus all flows are in the lowest priority level sharing equally the bandwidth of the link.

2. As the number of flows increases the low load condition does not hold

$$n(1)kb+n(2)kb>1,$$

   and there is not enough bandwidth for all flows to send at rate $kb$. If, however, the condition

$$n(1)b+n(2)kb<1$$

   holds, then it follows that $1/n(2)>kb$. Thus flows in group $l=2$ move up to priority level $i=2$, as there they always get the boundary rate $kb$, which is more than if they stayed in the lower priority level. As a result flows in group $l=1$ stay in priority level $i=1$, as they can continue sending at rate higher than their boundary rate $b$. Moving up a priority level would require them to reduce their sending rate to or below $b$.

3. As the number of flows continues to increase and the link is further congested the previous condition does not hold, and

$$n(1)b+n(2)kb>1.$$

However, when the load is still such that

$$n(1)b + n(2)b < 1,$$

flows in group 1 move up to priority $i = 2$. Otherwise their sending rate would be reduced to less than $b$. The flows in group 2 are, however, still sending more than $b$, as there is still enough bandwidth in the link. Therefore, though the two groups of flows are in the same priority level, the bandwidth is not divided equally between the two classes.

4. As the congestion deepens, and more flows are introduced to the link, the flows in group 2 have to also reduce their sending rate, and when condition

$$n(1)b + n(2)b > 1$$

holds, all flows are in the highest priority level with equal bit rates of less than $b$ units.

The actual bit rates of each flow as a function of the number of flows is therefore equal when

$$n(1)kb + n(2)kb < 1 \quad \text{or} \quad n(1)b + n(2)b > 1.$$

These correspond to times of low load, when there is no need to differentiate between flows, as there is enough bandwidth for everyone, and times of very high load, which as a condition should be very rare. In all other cases there is a difference in the bandwidth received by the flows in group $l = 1$ and flows in group $l = 2$, the ratio of bandwidth being at most equal to the nominal bit rate ratio $k$. The differences occur in the middle area, in cases of moderate congestion.

Figure 5.1 summarizes the areas described. We note from the picture that the area of differentiation is increased by decreasing $b$ and increasing $k$. As $b = \phi(1)a(1)$, this means decreasing $\phi(1)$ or increasing the ratio $\frac{a(i-1)}{a(i)}$. Therefore, assuming that $\frac{a(i-1)}{a(i)} = 2$, we have a large differentiation if we increase $k$.

#### 5.1.1.3 Optimal bandwidth allocations $I > 2$

Let us next study the effect of increasing the number of priority levels. The previous section showed that at some point when all flows are in the highest priority level, the bandwidth is divided equally. Differentiation is achieved, when the user group with more weight is able to move up in priority. We can thus increase the area of differentiation by increasing the number of priority levels $I$. Table 5.3 shows the resulting thresholds for the case $I = 3$, assuming again that $t(1, 1) = b$.

Again if each group of flows individually optimizes its actual bit rate $\beta$, the following scenarios can be observed as a function of the number of flows $n(1)$ and $n(2)$ when three priorities are given.

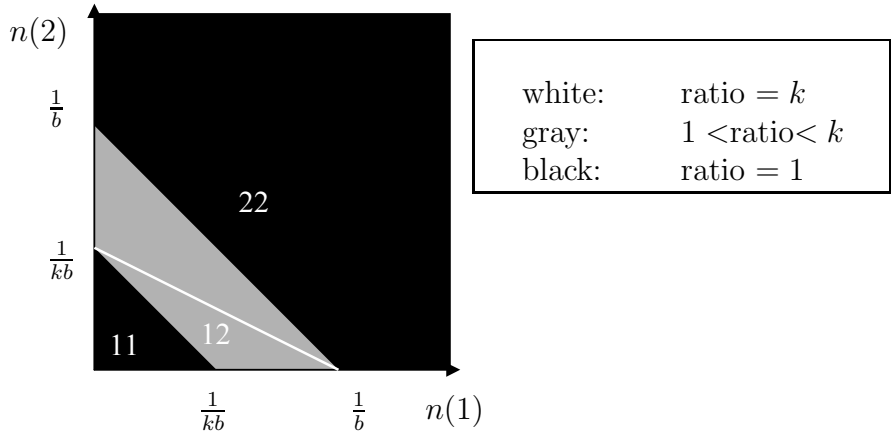



Figure 5.1: The optimal areas for two elastic flow groups competing for bandwidth under *per flow* marking, $I = 2$.

| | $l = 1$ | $l = 2$ |
|---|---|---|
| $i = 1$ | $\beta(1) \geq b$ | $\beta(2) \geq kb$ |
| $i = 2$ | $b/2 \leq \beta(1) < b$ | $kb/2 \leq \beta(2) < kb$ |
| $i = 3$ | $\beta(1) < b/2$ | $\beta(2) < kb/2$ |

Table 5.3: Priority level assignment conditions for $I = 3$

1. As was the case with $I = 2$, in times of low load, when the following condition holds
$$n(1)kb + n(2)kb < 1,$$
there is no advantage in being in a higher priority level, where the bit rate would be limited. For both groups, the bit rate achieved in the lower priority level, $\beta(l) > kb$ for all $l$. This is more than the bit rate in a higher priority level. Thus all flows are in the lowest priority level sharing equally the bandwidth of the link.

2. As the number of flows increases the low load condition does not hold
$$n(1)kb + n(2)kb > 1,$$
and there is not enough bandwidth for all flows to send at rate $kb$. If, however, the condition
$$n(1)b + n(2)kb < 1$$
holds, then it follows that $1/n(2) > kb$. Thus flows in group $l = 2$ move up to priority level $i = 2$, as there they always get the boundary rate $kb$, which is more than if they stayed in the lower priority level. As a result flows in group $l = 1$ stay in priority level $i = 1$, as they can continue sending at rate higher than their boundary rate $b$. Moving up a priority level would require them to reduce their sending rate to or below $b$.

3. As the number of flows continues to increase and the link is further congested the previous condition does not hold,

$$n(1)b + n(2)kb > 1.$$

However, when the load is still such that

$$n(1)b + n(2)b < 1,$$

flows in group 1 move up to priority $i = 2$. Otherwise their sending rate would be reduced to less than $b$. The flows in group 2 are, however, still sending more than $b$, as there is still enough bandwidth in the link. Therefore, though the two groups of flows are in the same priority level, the bandwidth is not divided equally between the two classes.

4. Now assume that $k < 2$.

- When the previous condition does not hold,

$$n(1)b + n(2)b > 1,$$

but we still have such a load that

$$n(1)\frac{kb}{2} + n(2)\frac{kb}{2} < 1,$$

both flow groups are in the middle priority $i = 2$ and divide the bandwidth equally.

- When the condition

$$n(1)\frac{kb}{2} + n(2)\frac{kb}{2} > 1$$

holds, there is not enough bandwidth for all flows to send at rate $kb/2$. If, however, the condition

$$n(1)\frac{b}{2} + n(2)\frac{kb}{2} < 1$$

holds, then it follows that $1/n(2) > kb/2$. Thus flows in group $l = 2$ move up to priority level $i = 3$, as there they always get the boundary rate $kb/2$, which is more than if they stayed in the lower priority level. As a result flows in group $l = 1$ stay in priority level $i = 2$, as they can continue sending at rate higher than their boundary rate $b/2$. Moving up a priority level would require them to reduce their sending rate to or below $b/2$.

- As the number of flows continues to increase and the link is further congested the previous condition does not hold,

$$n(1)\frac{b}{2} + n(2)\frac{kb}{2} > 1.$$

However, when the load is still such that

$$n(1)\frac{b}{2} + n(2)\frac{b}{2} < 1,$$

flows in group 1 move up to priority $i = 3$. Otherwise their sending rate would be reduced to less than $b/2$. The flows in group 2 are, however, still sending more than $b/2$, as there is still enough bandwidth in the link. Therefore, though the two groups of flows are in the same priority level, the bandwidth is not divided equally between the two classes.

5. Now assume that $k \geq 2$.

   - When the condition
   $$n(1)b + n(2)b < 1$$
   still holds, but
   $$n(1)b + n(2)\frac{kb}{2} > 1,$$
   there is not enough bandwidth for all flows to send at rate $kb/2 \geq b$. If, however, the condition
   $$n(1)\frac{b}{2} + n(2)\frac{kb}{2} < 1$$
   holds, then it follows that $1/n(2) > kb/2$. Thus flows in group $l = 2$ move up to priority level $i = 3$, as there they always get the boundary rate $kb/2$, which is more than if they stayed in the lower priority level. As a result flows in group $l = 1$ stay in priority level $i = 2$, as they can continue sending at rate higher than their boundary rate $b/2$. Moving up a priority level would require them to reduce their sending rate to or below $b/2$.
   - As the number of flows continues to increase and the link is further congested the previous condition does not hold,
   $$n(1)b + n(2)\frac{kb}{2} > 1.$$
   However, when the load is still such that
   $$n(1)\frac{b}{2} + n(2)\frac{b}{2} < 1,$$
   flows in group 1 move up to priority $i = 3$. Otherwise their sending rate would be reduced to less than $b/2$. The flows in group 2 are, however, still sending more than $b/2$, as there is still enough bandwidth in the link. Therefore, though the two groups of flows are in the same priority level, the bandwidth is not divided equally between the two classes.

6. As the congestion deepens, and more flows are introduced to the link, the flows in group 2 have to also reduce their sending rate, and when condition
   $$n(1)\frac{b}{2} + n(2)\frac{b}{2} > 1$$
   holds, all flows are in the highest priority level with equal bit rates of less than $b/2$ units.

The actual bit rates of each flow as a function of the number of flows is therefore equal when

$$n(1)kb + n(2)kb < 1 \quad \text{or} \quad n(1)\frac{b}{2} + n(2)\frac{b}{2} > 1.$$

In addition when $k < 2$, the actual bit rates equal in the middle area, where

$$n(1)\frac{kb}{2} + n(2)\frac{kb}{2} < 1 \quad \text{and} \quad n(1)b + n(2)b > 1.$$

With three priorities, the resulting optimal allocation can be divided into the areas summarized in figure 5.2. Note that, with more than two priorities, there is a difference between $k < 2$ and $k \geq 2$.

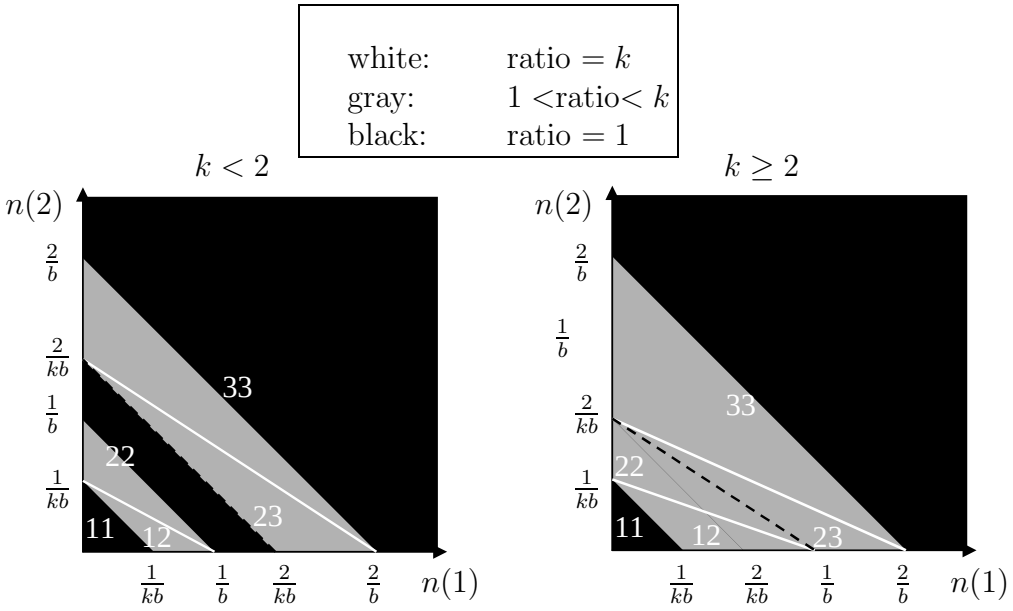



Figure 5.2: The optimal areas for two elastic flow groups competing for bandwidth under *per flow* marking and three priorities

## 5.1.2 Per packet marking

When flows are marked to priorities per packet, all packets of a flow do not necessarily have the same priority. When a flow in group $l$ exceeds the threshold $t(l, i)$ only the packets above the threshold are assigned to priority $pr(l) = i$ while the rest of the packets have $pr(l) > i$. This changes the equations presented earlier.

### 5.1.2.1 Bandwidth allocation model

With *per packet* marking, part of the flows packets are always marked to the highest priority $pr(l) = I$. Therefore, all flows have sub-streams in the highest priority

level $I$. In general, these sub-streams each receive equal amount of bandwidth and the sub-streams of flows in the lower priority level divide equally, among themselves, the remaining bandwidth. Again the rate received by a sub-stream in a priority level cannot exceed the boundary rate.

In the highest priority level where all the $n$ flows have a sub-stream whose rate is bounded by $t(l, I-1)$, the bandwidth share for flows in group 1 in priority level $I$ is

$$\beta(1, I) = \min\Big(\frac{1}{n}, t(1, I-1)\Big)$$

while the bandwidth share for flows in group 2 is

$$\beta(2, I) = \min\Big(\max\big(\frac{1}{n}, \frac{1 - n(1)t(1, I-1)}{n(2)}\big), t(2, I-1)\Big).$$

The remaining capacity is then

$$C_{I-1} = \max\Big(0, 1 - n(1)t(1, I-1) - n(2)t(2, I-1)\Big).$$

Thus if there is still capacity left after the higher priority flows have reached their bandwidth thresholds, the sub-streams in the subsequent priority levels divide the remaining capacity.

Define $s(l, i)$ as the number of flows whose priority level is less than or equal to $i$. Define also $\delta(l, i) = t(l, i-1) - t(l, i)$. Then the share of bandwidth for a flow with priority $i$ is

$$\beta(1, i) = \min\Big(\beta(1, i+1) + \frac{C_i}{s_i}, t(1, i-1)\Big), \tag{5.4}$$

$$\beta(2, i) = \min\Big(\beta(2, i+1) + \max\big(\frac{C_i}{s_i}, \frac{C_i - s(1, i)\delta(1, i)}{s(2, i)}\big), t(2, i-1)\Big). \tag{5.5}$$

The capacity remaining for priority level $i$ is

$$C_i = \max\Big(0, C_{i+1} - s(1, i+1)\delta(1, i+1) - s(2, i+1)\delta(2, i+1)\Big),$$

assuming that $\beta(l, I+1) = 0$ and $C(I) = 1$.

The lowest priority traffic then receive the share

$$\beta(1, 1) = \beta(1, 2) + \frac{C_1}{n_1},$$

$$\beta(2, 1) = \beta(2, 2) + \frac{C_1}{n_1},$$

as $t(l, 0) = \infty$. Thus with *per packet* marking differentiation does occur at the lowest priority level, due to the fact that some of the sub-streams of the flow are in the higher priority levels.

#### 5.1.2.2 Optimal bandwidth allocations $I = 2$

We assume that $I = 2$ and that flows inside a group behave in the same manner and that inside a group the flows have the same ratio of sub-streams in the priority classes. We can again divide our model into four different possibilities of actual bit rates $\beta(l)$ presented in table 5.4.

| | $l = 1$ | |
|---|---|---|
| $l = 2$ | $i = 1$ | $i = 2$ |
| $i = 1$ | $\beta(1) = b + \max(\frac{1-n(1)b-n(2)kb}{n(11)+n(21)}, 0)$<br>$\beta(2) = kb + \max(\frac{1-n(1)b-n(2)kb}{n(11)+n(21)}, 0)$ | $\beta(1) = \min(\frac{1}{n(1)+n(2)}, b)$<br>$\beta(2) = kb + \max(\frac{1-n(1)b-n(2)kb}{n(21)}, 0)$ |
| $i = 2$ | $\beta(1) = b + \max(\frac{1-n(1)b-n(2)kb}{n(11)}, 0)$<br>$\beta(2) = \min(\max(\frac{1}{n(1)+n(2)}, \frac{1-n(1)b}{n(2)}), kb)$ | $\beta(1) = \min(\frac{1}{n(1)+n(2)}, b)$<br>$\beta(2) = \min(\max(\frac{1}{n(1)+n(2)}, \frac{1-n(1)b}{n(2)}), kb)$ |

Table 5.4: Bandwidth allocation in the four possible network states, *per packet* marking, $I = 2$

If each group of flows individually optimizes its actual bit rate $\beta$, the following scenarios can be observed as a function of the number of flows $n(1)$ and $n(2)$.

1. In times of very low load, when the following condition holds

   $$(n(1), n(2)) \to 0,$$

   the flows divide the bandwidth equally. As the number of flows increases, the flows reduce their sending rate, and the portion of flows which receive bandwidth $\beta(l) = t(l, 1)$ become dominant.

2. When

   $$n(1)b + n(2)kb = 1,$$

   both flows are sending exactly at their thresholds $\beta(l) = t(l, 1)$ and bandwidth is divided in ratio of $k$.

3. As the congestion deepens, and more flows are introduced to the link, the flows in group 2 have to reduce their sending rate, and when condition

   $$n(1)b + n(2)kb > 1$$

   holds, but

   $$n(1)b + n(2)b < 1,$$

   all flows are in the highest priority class with $\beta(1) = b$ and $b < \beta(2) < kb$, as was the case with *per flow* marking.

4. As with *per flow* marking, when condition

$$n(1)b + n(2)b > 1$$

holds, all flows are in the highest priority class with equal bit rates of less than $b$ units.

The actual bit rates of each flow as a function of the number of flows is therefore equal when

$$(n(1), n(2)) \rightarrow 0, \text{ or} \tag{5.6}$$
$$n(1)b + n(2)b > 1. \tag{5.7}$$

These correspond to times of very low load, when there is no need to differentiate between flows, as there is enough bandwidth for everyone, and times of very high load, which as a condition should be very rare. In all other cases there is a difference in the bandwidth received by the flows in group $l = 1$ and flows in group $l = 2$, the ratio of bandwidth being at most equal to the nominal bit rate ratio $k$. The differences occur in the middle area, in cases of moderate congestion. Note, that in *per packet* marking the differentiation area is upper bounded by the same line, equation (5.7), as in *per flow* marking, but the area of differentiation is larger, as the lower bound equation (5.6) is lower then in *per flow* marking.

Figure 5.3 summarizes the areas described. We note from the picture that the area of differentiation is again increased by increasing $k$ and decreasing $b$.

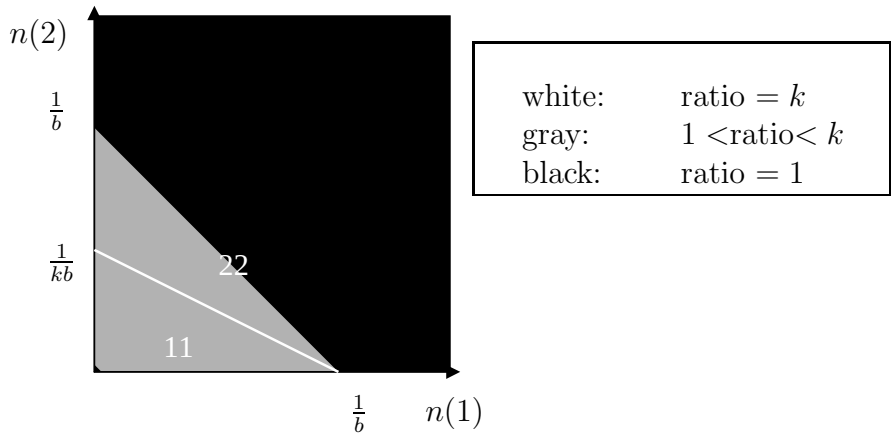



Figure 5.3: The optimal areas for two elastic flow groups competing for bandwidth under *per packet* marking, $I = 2$

#### 5.1.2.3 Optimal bandwidth allocations $I > 2$

The results for packet marking when $I > 2$ can be obtained analogous to the case $I = 2$ and are similar to the corresponding results $I > 2$ for *per flow* marking. We will thus not go through the resulting scenarios. They will be presented in terms of numerical examples in the next section.

## 5.2 Numerical results

To illustrate the optimization described earlier, let us consider few scenarios. We have flows with $(\phi(1), \phi(2)) = (0.04, 0.08)$ and $(\phi(1), \phi(2)) = (0.02, 0.08)$, i.e. ratio $k = 2$ and $k = 4$, respectively. We have

$$b = \phi(1) \cdot a(1) = \phi(1) \cdot 2^{I/2-1.5}.$$

Table 5.5 shows the different values for $b$ under the various scenarios.

| I | $b$ |
|---|---|
| 2 | $\phi(1)/\sqrt{(2)}$ |
| 3 | $\phi(1)$ |

Table 5.5: Different values for $b$ in the numerical examples

### 5.2.1 Per flow marking

Figure 5.4 shows the numerical results of the optimization for $I = 2$ and *per flow* marking. The number of flows of group 1 and 2 are on the $x$- and $y$-axis respectively. The black areas correspond to equal bandwidth allocation, the white areas correspond to bandwidth allocation equal to $k$ and the grey areas to bandwidth allocation between 1 and $k$. The figure clearly shows that the middle area, where differentiation is achieved, increases as $k$ increases.

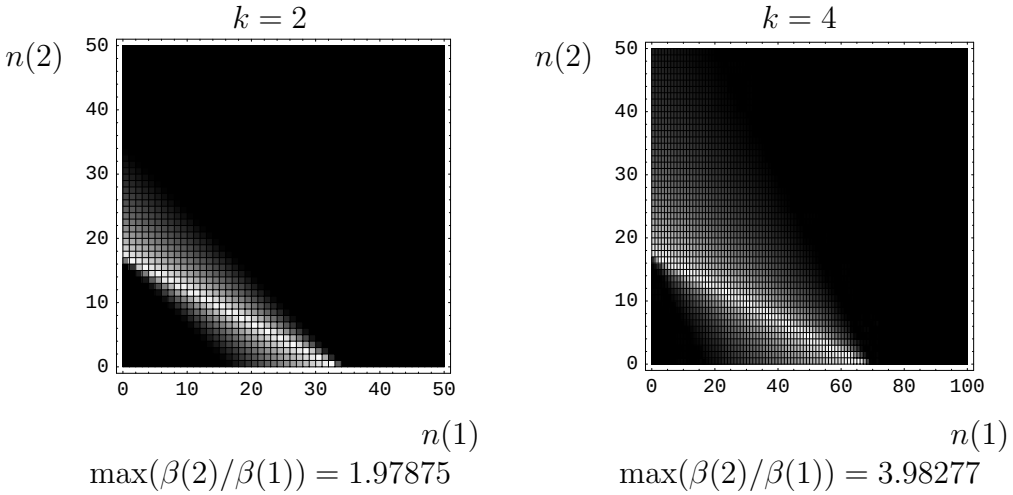



Figure 5.4: Bandwidth allocation $\beta(2)/\beta(1)$ for flows with *per flow* marking, $I = 2$ as a function of number of flows, $n(1)$ ($x$-axis) and $n(2)$ ($y$-axis).

We have also studied the model for three priority classes. As the number of priority classes, i.e. differentiation classes is increased the middle area becomes

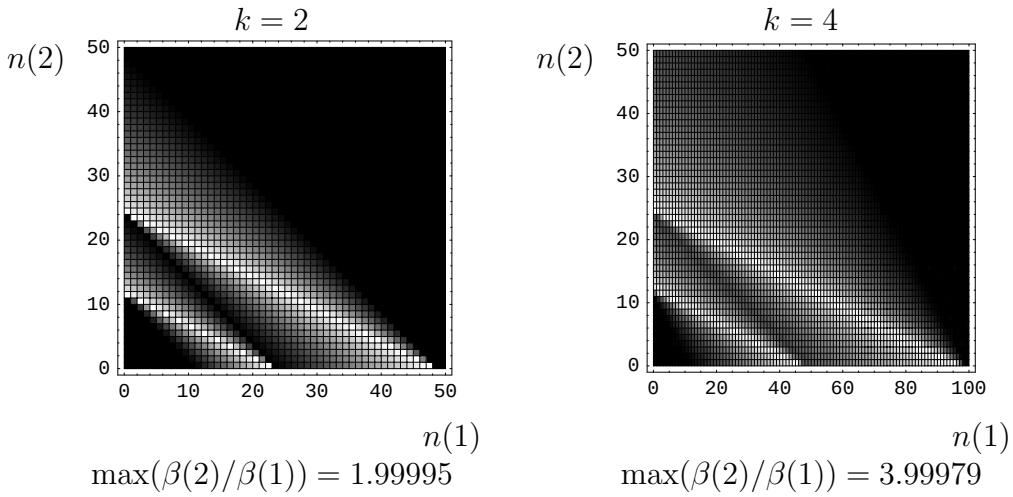



Figure 5.5: Bandwidth allocation $\beta(2)/\beta(1)$ for flows with *per flow* marking, $I = 3$ as a function of number of flows, $n(1)$ ($x$-axis) and $n(2)$ ($y$-axis).

larger and only rarely do all flows receive the same bandwidth. The middle area grows also as the ratio of weights, $k$, grows. Thus with many priority classes and large differences in weights, differentiation is at its best, as shown in figure 5.5.

### 5.2.2 Per packet marking

The effect that changing the marking scheme has on the optimal bandwidth allocations is illustrated in figures 5.6 and 5.7. The effect of increasing priority levels is the same as with *per flow* marking. The difference between the marking schemes, is that with *per packet* marking differentiation is also achieved in the lightly loaded area and with more than two priority levels the bandwidth share of flows is also greater than one in the middle areas of the graphs.

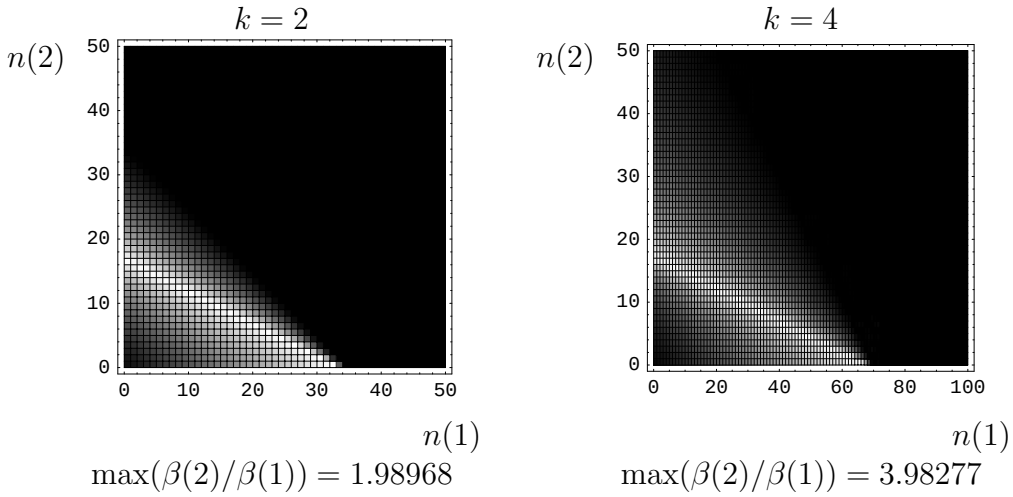



Figure 5.6: Bandwidth allocation $\beta(2)/\beta(1)$ for flows with *per packet* marking, $I = 2$ as a function of number of flows, $n(1)$ ($x$-axis) and $n(2)$ ($y$-axis).

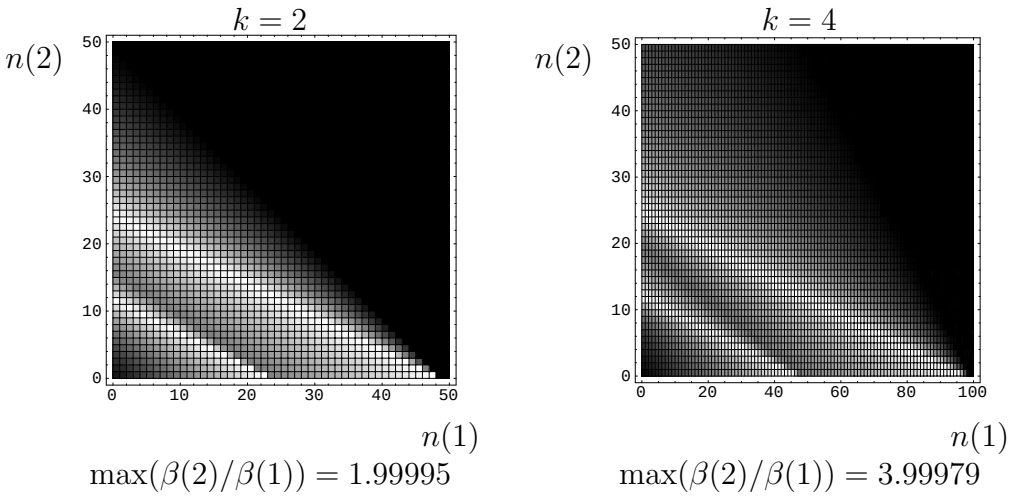



Figure 5.7: Bandwidth allocation $\beta(2)/\beta(1)$ for flows with *per packet* marking, $I = 3$ as a function of number of flows, $n(1)$ ($x$-axis) and $n(2)$ ($y$-axis).

## 5.3 Effect of marking

We have demonstrated the type of differentiation that results from marking packets to priorities according to thresholds based on price paid by or weight of greedy TCP flows.

In a single link network, the bandwidth share received by competing TCP flows depends on the weight purchased by the flows. We assumed a fixed number of flows and studied how the weighted bandwidth allocation depends on the congestion of the link. Using *per packet* marking results in a larger area of differentiation, but the difference compared to *per flow* marking occurs only when the network has a low load. The main drawback of the resulting division of bandwidth is the lack of differentiation when all flows are in the highest priority. The only way to guarantee differentiation, is to have many or even an infinite number of priority classes, so that the network offers an incentive for the flows to halve their sending rate.

Neither *per flow* marking nor *per packet* marking allocates bandwidth in proportion to fixed weights, rather the weights depends on the state of the network. The share of bandwidth is, however, at least equal and at most in proportion to the fixed weights. From the results, one can deduce that as the number of priority levels is increased the differentiation approaches that of relative services.

In the same way as was proposed in section 3.3.2 in the AF simulation studies, flows can also be differentiated in the highly loaded areas by allowing some flows the choice of more priorities than other flows. Then differentiation is always achieved, as flows are in different priority levels.

# Chapter 6

# Packet level differentiation model

The previous chapter showed the resulting bandwidth allocation of greedy TCP sources optimizing their share of bandwidth subject to conditioning at the flow level. In this chapter, we model more closely the mechanisms inside the DiffServ node to see how bandwidth is divided among the flows. As the mechanisms inside the nodes are on the packet level, we need to develop appropriate metering and marking models, TCP models, and buffer models, in order to study the resulting end-to-end differentiation. Furthermore, we also include non-elastic, i.e. non-TCP, traffic into our models.

Let us consider the same single link network as in the previous chapter, with the link capacity scaled to one. We have a scheduling unit consisting of at most two buffers. The buffers, one for each delay class $d$, use the WFQ principle with weights $w^d$, $d = 1, 2$. The buffer space is divided by $I$ discarding thresholds $K^d(i)$.

The network is loaded by a fixed number, $n$, of flows. Now flows can be greedy elastic flows or unelastic real-time flows. The flows choose a weight $\phi$ as a reference level, i.e. a flow sending at most its weight has its packets marked at least to middle priority. The user model is the same as in the previous chapters. The set of all flows is denoted by $\mathcal{L}$. It can be divided according to type of flow $\mathcal{L} = \mathcal{L}^{\mathrm{UDP}} \cup \mathcal{L}^{\mathrm{TCP}}$ or according to delay class $\mathcal{L} = \mathcal{L}^{\mathrm{rt}} \cup \mathcal{L}^{\mathrm{nrt}}$. For the delay class we will use the shorthand notation $\mathcal{L}^d$, where $d = 1$ corresponds to the real-time and $d = 2$ to the non-real time delay class. The flows are divided into groups based on type or delay class and weight $\phi$ of the flow. Each group then consists of $n(l)$ identical flows. Packets of a flow are marked to priority levels $i \in \mathcal{I} = \{1, ..., I\}$ by one of the two marking mechanisms of chapter 4. Marking is done according to the measured bit rate compared to the weight $\phi(l)$ of the flow.

We first show the main framework of our model, depicted in figure 6.1, by assuming that only non-TCP traffic is present. We then demonstrate how the TCP equilibrium sending rate of our system can be solved through parameterization of the system equations. Finally we present our system model for a general traffic mix and corresponding numerical examples.

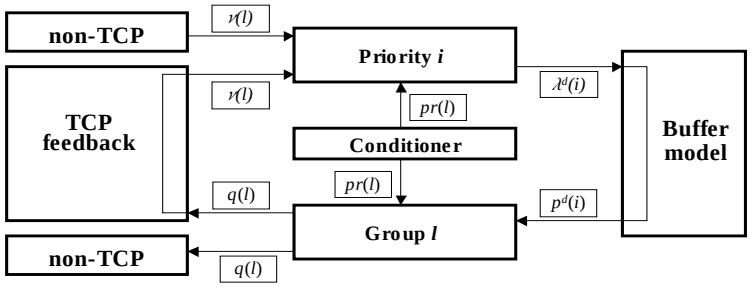



Figure 6.1: The modelling approach used on the packet level.

## 6.1 Non-TCP flows

Let us first look at unresponsive flows, which we interchangeably call non-TCP or UDP flows. These flows have a fixed sending rate, which from the perspective of our system can also be called packet arrival intensity, denoted by $\nu(l)$. At this point we do not need to make a distinction between delay classes. The flows inside one delay class could be elastic flows, non-elastic flows or a mixture of the two.

### 6.1.1 Packet arrival model

The marking mechanism determines how the packets of a flow are distributed among the possible priority levels. We study the marking options described earlier, *per flow* and *per packet* marking.

The metering result gives the packet arrival intensity $\nu(l)$ corresponding to a long term average of the bit rates of individual flows belonging to group $l$. The flows of group $l$ are then assigned priority $pr(l)$

$$pr(l) = \max\left[\min\left[\left\lfloor I/2 + 0.5 - \log_2 \frac{\nu(l)}{\phi(l)} \right\rfloor, I\right], 1\right], \tag{6.1}$$

with the corresponding thresholds

$$\begin{aligned} t(l,0) &= \infty, \\ t(l,i) &= \phi(l) \cdot 2^{-i+I/2-0.5}, \ i = 1, ..., I-1, \\ t(l,I) &= 0. \end{aligned} \tag{6.2}$$

Now the marking scheme used determines how the flows are grouped into priority aggregates. However, we can assume independent of the marking scheme that the resulting aggregates constitute a Poisson process with arrival intensity $\lambda(i)$ of priority level $i$. This stems from the assumption that the priority aggregate is composed of many flows.

#### 6.1.1.1 Per flow marking

Marking all packets of a flow to the same priority level gives the aggregate arrival intensity $\lambda(i)$ of priority level $i$ as

$$\lambda^d(i) = \sum_{l \in \mathcal{L}^d : pr(l)=i} n(l)\nu(l), \tag{6.3}$$

for all $i = 1, \ldots, I$.

#### 6.1.1.2 Per packet marking

Marking only overflow packets to the lower priority level gives the aggregate arrival intensity $\lambda(i)$ of priority level $i$ as

$$\lambda^d(i) = \sum_{l \in \mathcal{L}^d : pr(l) \leq i} n(l)(\min[\nu(l), t(l, i-1)] - \min[\nu(l), t(l, i)]),$$

for all $i = 1, \ldots, I$.

### 6.1.2 Buffer models

By modelling the scheduling unit, we are able to determine the loss probability and delay for each flow aggregate in terms of the packet arrival intensity to the scheduling unit. We consider two cases: one queue servicing only traffic of one delay class and two queues with one for each delay class.

#### 6.1.2.1 Buffer model for one delay class

We first present the analytical model for the one buffer system servicing only one delay class. A similar model has been discussed in [MBDM99] and [STK99], but only for the special case of two priority levels.

Denote the discarding threshold of priority level $i$ by $K(i)$, $i \in \mathcal{I}$, with $K(I) = K$, the size of the buffer and define $K(0) = 0$. The packet transmission time with full link rate is assumed to be exponentially distributed with mean $1/\mu$ time units, and let $\lambda(i)$ denote the arrival rate of packets in priority level $i$. Define the cumulative sum of arrival intensities of those priority levels accepted into the system as $\lambda_i = \sum_{k=i}^{I} \lambda(k)$. The corresponding load is $\rho_i = \frac{\lambda_i}{\mu}$. The buffer can then be modelled as an $M/M/1/K$ queue with state dependent arrival intensities as depicted in figure 6.2.

The stationary distribution of the system is

$$\pi_m = \rho_1^m \pi_0,$$

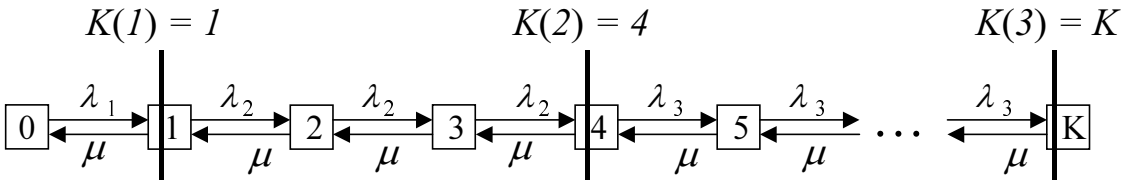



Figure 6.2: State transition diagram for one buffer modelled as an $M/M/1/K$ queue, when $I = 3$.

for $m = 1, ..., K(1) - 1$. Correspondingly,

$$\pi_{K(i)+m} = \left( \prod_{j=1}^{i} \rho_j^{K(j)-K(j-1)} \right) \cdot \rho_{i+1}^{m} \pi_0,$$

for $i = 1, ..., I - 1$ and $m = 0, ..., K(i + 1) - K(i) - 1$. Finally, the probability that the buffer is full is

$$\pi_K = \prod_{j=1}^{I} \rho_j^{K(j)-K(j-1)} \pi_0.$$

The probability that the buffer is empty $\pi_0$ is determined from the normalization condition $\sum_{i=0}^{K} \pi_i = 1$.

Using the shorthand notation $b(0) = 1$ and

$$b(i) = \rho_i^{K(i)-K(i-1)},$$

for $i \in \mathcal{I}$, the stationary probability is

$$\pi_{K(i)+m} = \left( \prod_{j=1}^{i} b(j) \right) \cdot \rho_{i+1}^{m} \pi_0, \tag{6.4}$$

for $i = 0, ..., I - 1$ and $m = 1, ..., K(i + 1) - K(i)$.

Now, the probability $p(i)$ that packets belonging to priority level $i$ will be lost is simply

$$p(i) = \sum_{j=K(i)}^{K} \pi_j = p(i + 1) + \sum_{m=0}^{K(i+1)-K(i)-1} \pi_{K(i)+m},$$

for $i = 1, ..., I-1$, with $\pi_{K(i)+m}$ defined in equation (6.4). For the highest priority level, the loss probability is

$$p(I) = \pi_K. \tag{6.5}$$

Using again a shorthand notation,

$$\begin{aligned} a(i) &= \sum_{m=0}^{K(i)-K(i-1)-1} \rho_i^m \\ &= \frac{1 - \rho_i^{K(i)-K(i-1)}}{1 - \rho_i}, \end{aligned}$$

the loss probability is

$$p(i) = (\sum_{j=i+1}^{I} a(j) \prod_{h=1}^{j-1} b(h) + \prod_{h=1}^{I} b(h))\pi_0, \quad (6.6)$$

where

$$\pi_0^{-1} = \sum_{j=1}^{I} a(j) \cdot \prod_{h=0}^{j-1} b(h) + \prod_{h=1}^{I} b(h).$$

The throughput of each priority level is defined as the net rate at which packets leave the system, i.e.

$$\lambda_{\text{eff}}(i) = \lambda(i)(1 - p(i)).$$

The expected delay seen by the packet belonging to priority level $i$ is

$$\bar{D}(i) = \frac{\sum_{m=0}^{K(i)-1} (m+1)\pi_m}{\mu \sum_{m=0}^{K(i)-1} \pi_m}.$$

Introducing a third shorthand notation

$$\tilde{a}(i) = \sum_{m=0}^{K(i)-K(i-1)-1} (K(i-1)+m+1)\rho_i^m,$$

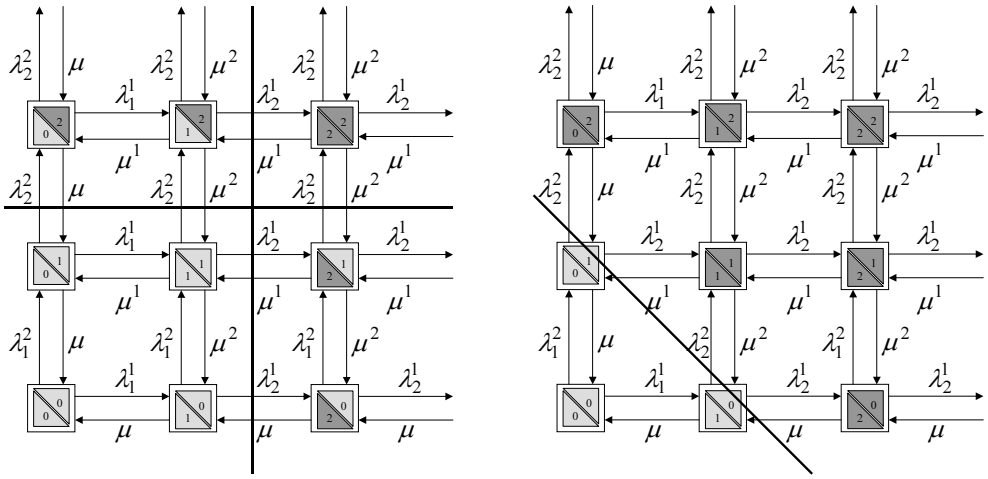
gives the expected delay as

$$D(i) = \frac{\sum_{j=1}^{i} \tilde{a}(j) \cdot \prod_{h=1}^{j-1} b(h)}{\mu \sum_{j=1}^{i} a(j) \cdot \prod_{h=1}^{j-1} b(h)}. \quad (6.7)$$

#### 6.1.2.2 Buffer model for two delay classes

Now assume that our system has two delay classes. This can be modelled as two dependent $M/M/1/K$ queues with state dependent arrival intensities. The packet transmission time is assumed to be exponentially distributed with mean $1/\mu$ time units. Thus, if both buffers are non-empty, packet service rates are $w^d\mu = \mu^d$, for $d = 1$ and $d = 2$, i.e. for real time and non-real time traffic, respectively. The arrival intensities are state dependent, according to the discard threshold function used. Let $\lambda^1(i)$ and $\lambda^2(i)$ denote the arrival rate of packets of priority level $i$ for rt and nrt delay classes respectively. Define the cumulative sum of arrival intensities of those priority levels accepted to the system as $\lambda_i^d = \sum_{k=i}^{I} \lambda^d(k)$.

The state transition diagram is illustrated in figure 6.3. The figure on the left depicts the transition diagram in the case of independent discarding and the figure on the right in the case of dependent discarding. The resulting stationary distribution and loss probabilities $p^d(i)$ can only be solved numerically.



Independent discarding Dependent discarding

Figure 6.3: State transition diagrams for a two buffer scheduling unit.

### 6.1.3 Flow group loss probability

From the buffer model of section 6.1.2.2 we have the loss probabilities $p^d(i)$ in terms of priority levels. We are however mainly interested in the loss probability of flow groups as opposed to loss probability of priority levels.

We define the throughput of a flow as

$$\theta(l) = \nu(l) \cdot (1 - q(l)),$$

where $q(l)$ refers to the loss probability of flow group $l$ given by equation (6.8) or (6.9) below.

#### 6.1.3.1 Per flow marking

Under the *per flow* marking mechanism, the packet loss probability, $q(l)$, for a flow belonging to group $l$ and for $d = 1, 2$, is

$$q(l) = p^d(pr(l)),\ l \in \mathcal{L}^d. \tag{6.8}$$

#### 6.1.3.2 Per packet marking

Under the *per packet* marking scheme, the packet loss probability, $q(l)$, for a flow belonging to group $l$ and for $d = 1, 2$, is

$$q(l) = \sum_{j=1}^{I} p^d(j) \frac{\min\left[\nu(l), t(l, j-1)\right] - \min\left[\nu(l), t(l, j)\right]}{\nu(l)}, \; l \in \mathcal{L}^d \tag{6.9}$$

## 6.2 TCP flows

Let us next look at the system where some of the flows are TCP flows. Assume that the UDP flows behave as given above and that they are affected by the presence of TCP flows only through the buffer model presented earlier. The TCP flows on the other hand respond to the congestion in the network by adjusting their sending rate. Here we use the loss probability as the feedback signal to determine the equilibrium sending rate. For the UDP flows the sending rate was constant. The TCP flows are characterized by their round-trip time. Let $RTT(l)$ denote the round-trip time of flows in group $l \in \mathcal{L}^{\text{TCP}}$.

We first introduce the TCP model employed. We then show the method used to parameterize the packet model equations. Parameterization is needed to solve the equilibrium sending rate of TCP flows under *per flow* marking, but will also be used when *per packet* marking is employed.

### 6.2.1 Flow behavior

The models presented in section 2.3 give a closed form expression for TCP sending rate, as a function of packet loss probability. In our model setup depicted in figure 6.1, we have the packet loss probability $q(l)$ for each flow group $l$. Furthermore, assuming that the dynamics of the buffer is faster than that of TCP we can use the steady state TCP throughput expression of, e.g. Kelly. We thus have the expression

$$\nu(l) = \frac{1}{RTT(l)} \sqrt{2 \, \frac{1 - q(l)}{q(l)}}, \; l \in \mathcal{L}^{\text{TCP}}. \tag{6.10}$$

Note that if the queueing delay is not negligible compared to the RTT, the equation is then of form

$$\nu(l) = \frac{1}{RTT(l) + \bar{E}(l)} \sqrt{2 \, \frac{1 - q(l)}{q(l)}}, \; l \in \mathcal{L}^{\text{TCP}},$$

where $\bar{E}(l)$ would be calculated from $\bar{D}(i)$ in the same fashion as $q(l)$ is calculated from $p(i)$.

### 6.2.2 Flow aggregates

The priority levels have discrete values. Therefore, the flow priorities form a step function of $\nu(l)$. From equation (6.3) we see that as the priority levels are discrete, under *per flow* marking, the resulting aggregate arrival intensity $\lambda(i)$ is also a discontinuous function. The discontinuity due to discrete priority levels ($I = 3$) is illustrated in figures 6.4 and 6.5. In figure 6.5 it is assumed that the system has only one flow. In addition, *per flow* marking in assumed in the figures.

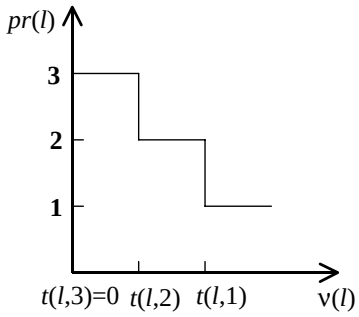



Figure 6.4: Priority level $pr(l)$ as a function of sending rate $\nu(l)$ and when $I = 3$

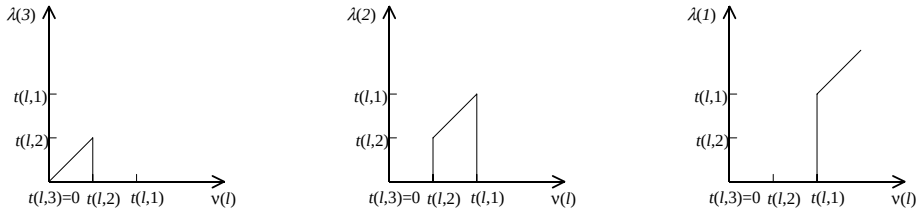



Figure 6.5: Arrival intensities of priority levels of one flow and $I = 3$ under *per flow* marking

#### 6.2.2.1 Parameterization

In order to solve the TCP equilibrium sending rate, we need to make the aggregate arrival intensity piecewise continuous. This is achieved through parameterization of $\nu(l)$ and by introducing a linear function between adjacent priorities.

Assume now that the packet arrival intensity of flow group $l$ is given as a function of an auxiliary variable $x_l$. Define $\mathbf{x} = \{x_l \mid l \in \mathcal{L}\}$. The arrival intensity vector is then $\nu(\mathbf{x}) = \{\nu(l, x_l) \mid l \in \mathcal{L}\}$, where $\nu(l, x_l)$ is chosen to be the following piecewise continuous function. In the range $0 \leq x_l \leq 1$, $\nu(l, x_l) = 0$, and in the range $1 \leq x_l \leq 2$, $\nu(l, x_l) = t(l, I-1)x_l - t(l, I-1)$. From there on the parameterized form is

$$\nu(l, x_l) = \begin{cases} t(l, I-i) & 2i \leq x_l \leq 2i+1,\ i = 1, \ldots, I-1 \\ t(l, I-i)x_l - 2i \cdot t(l, I-i) & 2i+1 \leq x_l \leq 2i+2,\ i = 1, \ldots, I-2. \end{cases}$$

For $x_l \geq 2I - 1$, the function grows linearly towards infinity.

In the same fashion we have the priority vector $\mathbf{pr}(\mathbf{x}) = \{pr(l, x_l) \mid l \in \mathcal{L}\}$, where $pr(l, x_l)$ is a piecewise continuous function. In the range $0 \leq x_l \leq 2$, $pr(l, x_l) = I$, and from there on the parameterized form is

$$pr(l, x_l) = \begin{cases} -x_l + I + 1 + i & 2i \leq x_l \leq 2i + 1,\ i = 1, \ldots, I - 1 \\ I - i & 2i + 1 \leq x_l \leq 2i + 2,\ i = 1, \ldots, I - 2. \end{cases}$$

For $x_l \geq 2I - 1$, $pr(l, x_l) = 1$. Figure 6.6 shows the functions in the parameterized form.

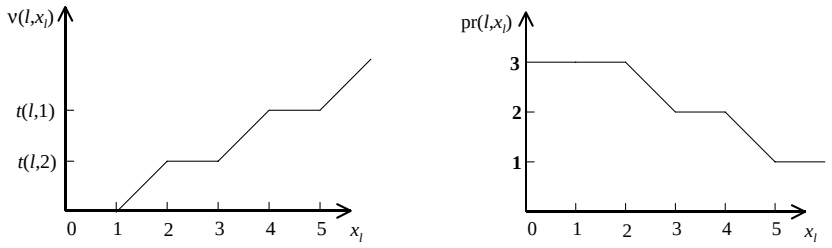



Figure 6.6: Parameterized sending rate $\nu(l, x_l)$ and priority $pr(l, x_l)$, when $I = 3$

As before the marking scheme used determines how the flows are grouped into priority aggregates. We have, in vector form,

$$\lambda^d(\mathbf{x}) = \{\lambda^d(i, \mathbf{x}) \mid i = 1, \ldots I\},$$

where $\lambda^d(i, \mathbf{x})$ is defined by either equation (6.11) or (6.12) given below.

#### 6.2.2.2 Per flow marking

Marking all packets of a flow to the same priority level gives the aggregate arrival intensities in parameterized form as

$$\lambda^d(i, \mathbf{x}) = \sum_{l \in \mathcal{L}^d : |pr(l, x_l) - i| < 1} n(l)\nu(l, x_l)(1 - \mid pr(l, x_l) - i \mid). \tag{6.11}$$

Note that with *per flow* marking the priority levels have discrete values. Then, the parameterization has to be done so that the functions $\lambda^d(i, \mathbf{x})$ become smooth. This is achieved by making the priority function piecewise continuous by introducing a linear function between adjacent priority levels. As an example, when $\lceil pr(l) \rceil > pr(l)$, the fraction $\lceil pr(l) \rceil - pr(l)$ of the flows traffic is in priority level $pr(l)$ and the rest in priority level $pr(l) + 1$. This is illustrated for one flow and $I = 3$ in figure 6.7.

#### 6.2.2.3 Per packet marking

Though parameterization is not needed when *per packet* marking is used, we present here the parameterized form. Marking only overflow packets to the lower priority level gives the aggregate arrival intensity in parameterized form as

$$\lambda^d(i, \mathbf{x}) = \sum_{l \in \mathcal{L}^d : pr(l, x_l) \leq i} n(l)(\min\left[\nu(l, x_l), t(l, i - 1)\right] - \min\left[\nu(l, x_l), t(l, i)\right]). \tag{6.12}$$

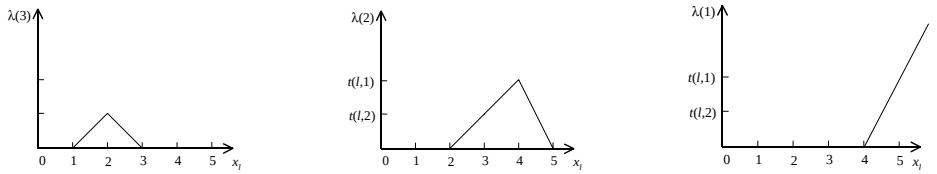



Figure 6.7: Parameterized arrival intensities of priority levels of one flow and $I = 3$ under *per flow* marking

### 6.2.3 Buffer models

By modelling the scheduling unit, we are able to determine the loss probability for each flow aggregate in terms of the packet arrival intensity to the scheduling unit. We consider the two cases presented in section 6.1.2: one queue servicing only traffic of one delay class and two queues one for each delay class.

As a result we have the loss probability

$$\mathbf{p}^d(\mathbf{x}) = \{p^d(i, \mathbf{x}) \mid i = 1, \ldots, I\},$$

where $p^d(i, \mathbf{x})$ is given by equation

$$p(i, \mathbf{x}) = \sum_{j=K(i)}^{K} \pi_j(\lambda(\mathbf{x})) \tag{6.13}$$

for the one buffer case, but can only be given numerically for the two buffer case.

### 6.2.4 Loss probability feedback signal

From the buffer model we have the loss probabilities $\mathbf{p}^d(\mathbf{x})$. In order to solve the arrival intensity per flow group we need to define the loss probability in terms of flow groups as opposed to defining it in terms of priority levels. We have

$$\mathbf{q}(\mathbf{x}) = \{q(l, \mathbf{x}) \mid l \in \mathcal{L}^d\},$$

where $q(l)$ refers to the loss probability of flow group $l$ given by equation (6.14) or (6.15) below.

#### 6.2.4.1 Per flow marking

Under the *per flow* marking mechanism, the packet loss probability, $q(l, \mathbf{x})$, for a flow belonging to group $l$ is

$$\begin{aligned} q(l, \mathbf{x}) &= p^d(\lfloor pr(l, x_l) \rfloor, \mathbf{x})(\lfloor pr(l, x_l) \rfloor + 1 - pr(l, x_l)) \\ &+ p^d(\lfloor pr(l, x_l) \rfloor + 1, \mathbf{x})(pr(l, x_l) - \lfloor pr(l, x_l) \rfloor), l \in \mathcal{L}^d. \end{aligned} \tag{6.14}$$

#### 6.2.4.2 Per packet marking

Under the *per packet* marking scheme, the packet loss probability, $q(l, \mathbf{x})$, for a flow belonging to group $l$ is then

$$q(l, \mathbf{x}) = \sum_{j=1}^{I} p^d(j, \mathbf{x}) \frac{\min\left[\nu(l, x_l), t(l, j-1)\right] - \min\left[\nu(l, x_l), t(l, j)\right]}{\nu(l, x_l)}, \; l \in \mathcal{L}^d. \tag{6.15}$$

### 6.2.5 Fixed point equation

Finally, solving the equilibrium throughput $\nu(\mathbf{x})$ amounts to solving the fixed point equation

$$\nu(l, x_l) = \frac{1}{RTT(l)} \sqrt{2 \frac{1 - q(l, \mathbf{x})}{q(l, \mathbf{x})}}, \; l \in \mathcal{L}^{\mathrm{TCP}}. \tag{6.16}$$

Or in terms of a scalar product of vectors,

$$\nu(\mathbf{x})^T \cdot \nu(\mathbf{x}) = \frac{2}{RTT(l)^2} (\mathbf{e} - \mathbf{q}(\mathbf{x})) \cdot \mathbf{q^{-1}}(\mathbf{x}),$$

where $\mathbf{e}$ is a vector of ones with length $L^{\mathrm{TCP}}$ and $\mathbf{q^{-1}}(\mathbf{x}) = \{1/q(l, \mathbf{x}) \mid l \in \mathcal{L}^{\mathrm{TCP}}\}$.

### 6.2.6 Example solution, one flow

#### 6.2.6.1 One priority level

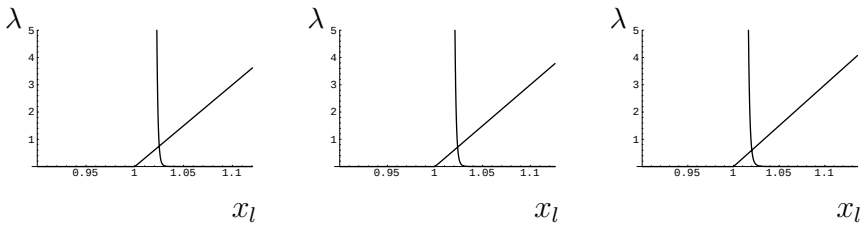
To illustrate the above fixed point equation and the uniqueness of its solution, let us consider the most simplified scenario. We have one TCP flow, $L = L^{\mathrm{TCP}} = 1$, one buffer and one priority $I = 1$. We thus have $\nu(l) = \lambda(1) = \lambda$ and

$$q(l) = p(1) = \frac{\lambda^K (1 - \lambda)}{1 - \lambda^{K+1}},$$

where $p(1)$ is the probability of being in state $K$ in a $M/M/1/K$ queue. The fixed point equation is thus

$$\lambda = \frac{1}{RTT} \sqrt{2 \frac{1 - \lambda^K}{\lambda^K (1 - \lambda)}}.$$

The equation has a unique real solution, illustrated by figure 6.8. For consistency, the pictures are drawn in the parameterized form, i.e. $\lambda = \nu(l, x_l)$.



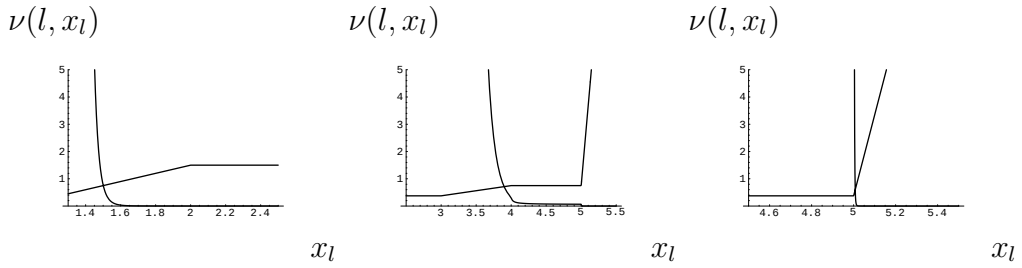

$K = 39$ $\lambda = 0.751163$ | $K = 33$ $\lambda = 0.712244$ | $K = 22$ $\lambda = 0.601435$

Figure 6.8: Solution to the fixed point equation, in a system with one $M/M/1/K$ buffer and $L = 1$ and $I = 1$

#### 6.2.6.2 Many priority levels

Now consider still only one flow, but many priorities. Assume further, for illustrative purposes only, that we are using *per flow* marking. Then with one flow in the system, if we choose the weight $\phi(1)$ large enough, the resulting equilibrium sending rate will be in the highest priority and this solution will correspond to the solution in the case of one priority level. This already gives us a proof of the conjecture made later on in this chapter that the more priority levels there are the better the differentiation.

Consider now a system with $I = 3$ priorities and the buffer partitioned into discarding areas in the following manner: $K(3) = 39$, $K(2) = 33$ and $K(1) = 22$. Then in a system with only one flow, the equilibrium sending rate corresponds to the fixed point equation solution of $I = 1$ and $K = K(pr)$, where $pr$ is the priority of the flow. This is shown in figure 6.9.




$\phi = 3$, $pr = 3$ $\lambda = 0.751163$ | $\phi = 0.75$, $pr = 2$ $\lambda = 0.712244$ | $\phi = 0.375$, $pr = 1$ $\lambda = 0.601435$

Figure 6.9: Solution to the fixed point equation, in a system with one $M/M/1/K$ buffer and $L = 1$ and $I = 3$

From the pictures in figure 6.9 we also notice that the function

$$\frac{1}{RTT(l)}\sqrt{2\frac{1-q(l,\mathbf{x})}{q(l,\mathbf{x})}}$$

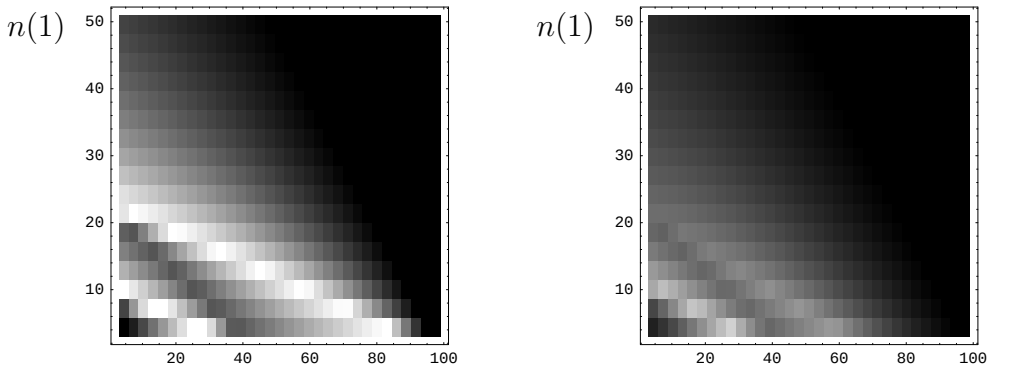
is piecewise continuous as a result of the parameterization.

## 6.3 Numerical results

In order to study the relative services achievable by the given mechanisms we have the following basic scenario: two flow groups, $L = 2$, three priority levels, $I = 3$, $\mu = 1$, $RTT = 1000/\mu$ and $K = 39$. Define $k = \phi(1)/\phi(2)$ and $r = \nu(1)/\nu(2)$. We will consider three cases: one delay class with TCP traffic, one delay class with TCP and non-TCP traffic and finally TCP and non-TCP traffic separated into two delay classes.

### 6.3.1 One buffer with only TCP traffic

Figure 6.10 shows the ratio, $r = \nu(1)/\nu(2)$, of bandwidth allocations for flows with $(\phi(1), \phi(2)) = (0.08, 0.02)$, i.e. $k = 4$, as a function of number of flows $n(1)$ and $n(2)$. The black areas correspond to equal bandwidth allocation, while the white areas correspond to bandwidth allocation ratio equal to $k$. On the left the marking is *per flow*, while on the right the marking is *per packet*.




Per flow marking
$\min(r) = 1$, $\max(r) = 4$

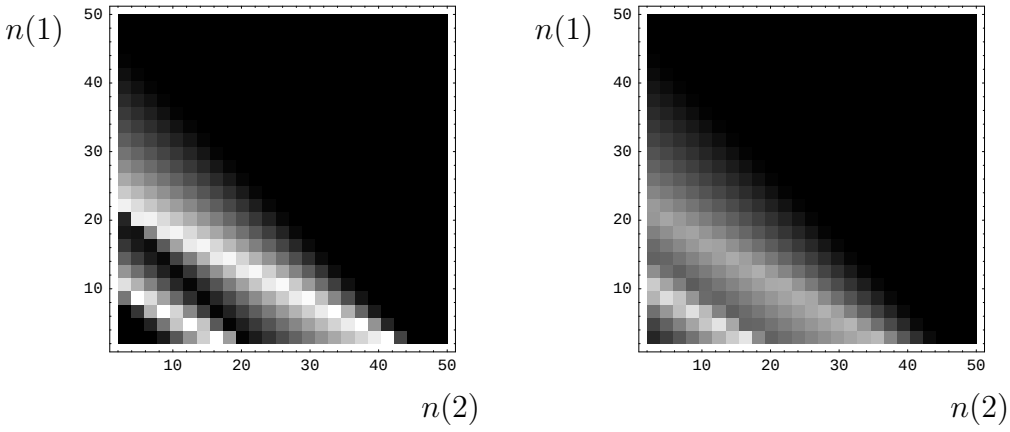
Per packet marking
$\min(r) = 1$, $\max(r) = 3.46$

Figure 6.10: Relative bandwidth allocation $\nu(1)/\nu(2)$ for flows with $k = 4$ as a function of the number of flows. White:$r \geq k$, black: $r \leq 1$, grey:$1 < r < k$.

Figure 6.11 shows the same effect of marking for $(\phi(1), \phi(2)) = (0.08, 0.04)$, i.e. $k = 2$. These figures 6.10 and 6.11 can be compared to figures 5.5 and 5.7 of chapter 5. In chapter 5 the ordering is according to marking while here the ordering is according to ratio $k$. However, the figures depict the bandwidth allocation ratio in the same way and give similar results.

Figure 6.12 shows the two-dimensional cross-section of the diagonals in figure 6.11, i.e. bandwidth allocation at points where both flow groups have the same number of flows. The trajectories of $\nu(1)$ and $\nu(2)$ are grey and black, respectively. The total number of flows is on the $x$-axis.

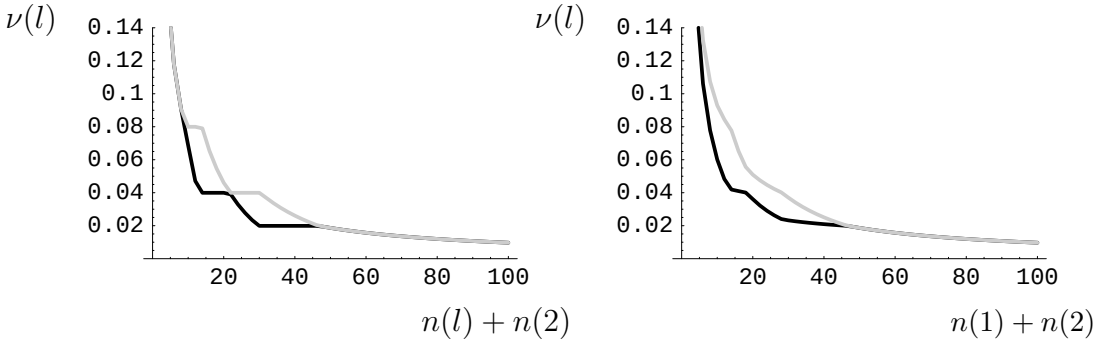
Figure 6.13 shows the effect of increasing the number of priority levels from $I = 3$ to $I = 6$, with $K = 78$ and $k = 2$. Here again we depict the cross section




Per flow marking
$\min(r) = 1$, $\max(r) = 2$

Per packet marking
$\min(r) = 1$, $\max(r) = 1.87$

Figure 6.11: Relative bandwidth allocation $\nu(1)/\nu(2)$ for flows with $k = 2$ as a function of number of flows. White:$r \geq k$, black: $r \leq 1$, grey:$1 < r < k$.




Per flow marking

Per packet marking

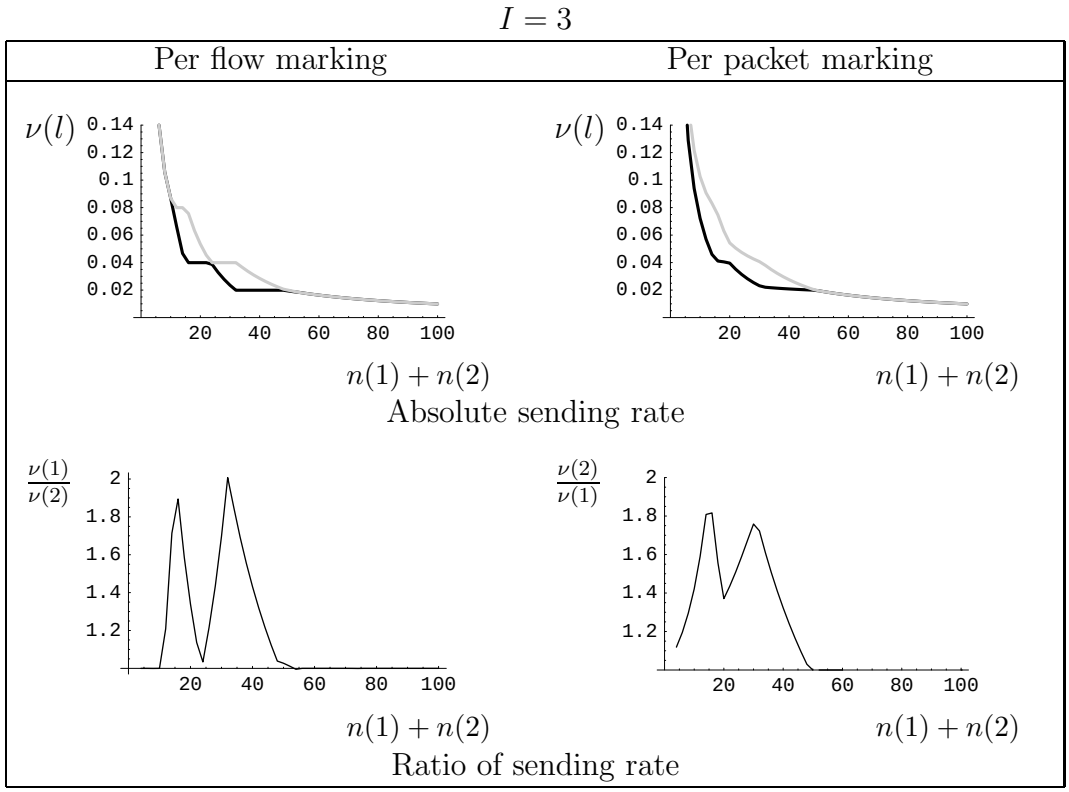
Figure 6.12: Absolute bandwidth allocation for flows when $k = 2$ as a function of the total number of flows. $\nu(1)$ gray, $\nu(2)$ black.

giving the absolute bandwidth allocations $\nu(1)$ and $\nu(2)$ as a function of the total number of flows $n(1) + n(2)$ under the assumption that $n(1) = n(2)$.

Figure 6.13 demonstrates that once the link is congested enough, the TCP mechanism forces the flows to drop their rate low enough to attain the highest priority level and divide bandwidth equally. Under severe congestion there is hence no difference in the marking schemes. By doubling the number of priority levels to $I = 6$, the area where the highest priority level is attained is pushed further on. Thus with enough priority levels, bandwidth is practically never divided equally among flows with different weights in times of high congestion.

From the figures, we notice that under high load conditions the marking schemes are the same. Elsewhere two main differences occur. When the link has low load, the *per flow* marking scheme marks all the packets to the lowest priority level and they share the bandwidth in equal proportions. The *per packet* marking scheme, on the other hand, marks only the overflow packets to the lowest priority level,

meaning that the rest of the packets have the highest or the middle priority.



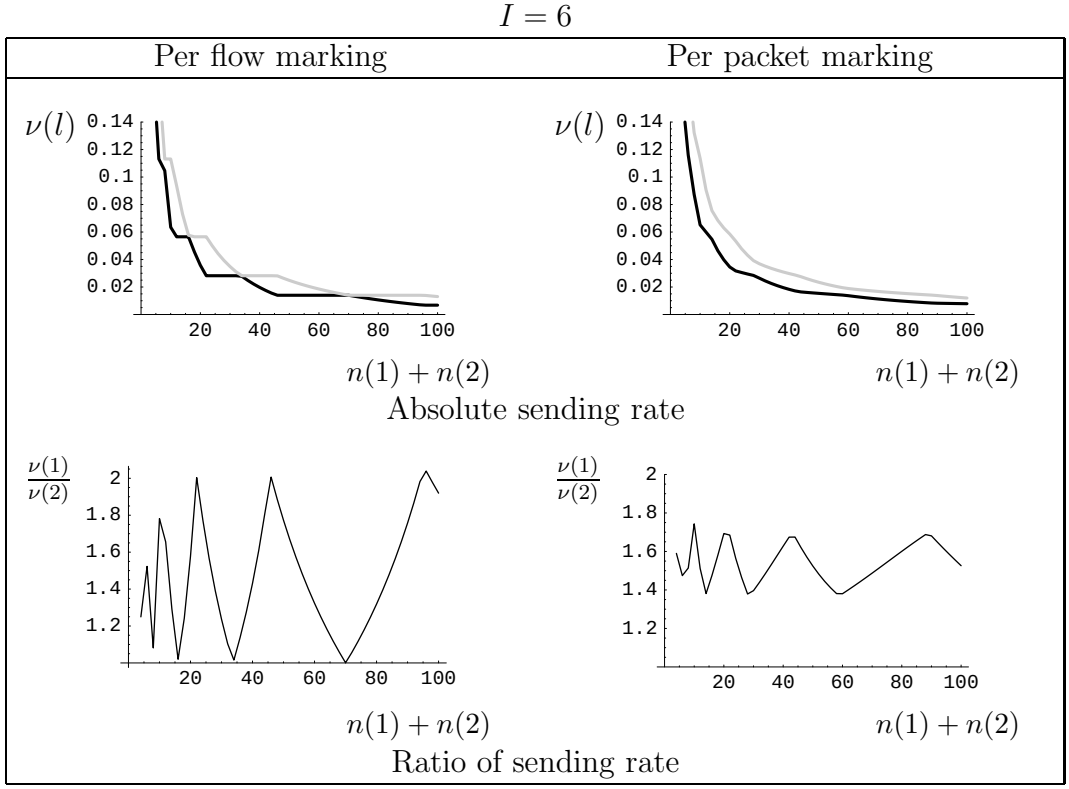




Figure 6.13: Absolute and relative bandwidth allocation for flows when $k = 2$ as a function of the total number of flows under the assumption $n(1) = n(2)$. $\nu(1)$ gray, $\nu(2)$ black.

When the link is medially loaded, the *per flow* marking scheme produces larger difference in bandwidth allocation, at times reaching allocation in proportion to $k$. The *per packet* marking on the other hand is not able to attain differentiation by a factor of $k$, depicted in figure 6.10 by the grayer areas of the picture on the right. The difference stems from the fact that some packets are always also marked to the highest priority. These observations coincide with those made in chapter 5.

However, opposed to the flow level model results of chapter 5, from figure 6.12 we see that though the *per packet* scheme does not attain a maximum differentiation of $k$, its minimum differentiation is always larger than that of the *per flow* marking as long as enough priority levels are employed.

Based on the results it is not clear which marking scheme is fairer in dividing bandwidth among elastic flows. It would seem that though *per packet* marking is not able to divide bandwidth at any time in proportion to the weights $\phi(l)$, it is fairer in the sense that it does not give equal proportions of bandwidth to flows, except when all flows are marked to the highest priority. In order to assess between the marking schemes, we need to model the case of elastic and streaming traffic sharing the link, done in the following two sections.

### 6.3.2 One buffer with TCP and UDP traffic

In the previous section we had TCP traffic scheduled by one buffer for which we solved the fixed point equation (6.16). Now consider still one delay class and one buffer, but assume that the flows in group $l = 1$ are non-TCP flows. The difference is that they have a constant sending rate $\nu(1)$, and the fixed point equation is only solved for flows in group $l = 2$.

We set up the above scenario to demonstrate the relationship between flow priority and achieved throughput. Those flows with a constant sending rate will have constant priority, while the TCP flows that adjust their sending rate according to congestion also adjust their priority.

We study the achieved throughputs $\theta(l) = \nu(l)(1 - q(l))$ under three constant sending rates $\nu(1)$, chosen so that under *per flow* marking, the flows are in the highest, middle, and lowest priority $\nu(1) = 0.039$, $\nu(1) = 0.079$, and $\nu(1) = 0.16$, respectively.

Using our basic scenario for three priority levels: $I = 3$, $\mu = 1$, $RTT = 1000/\mu$, $K = 39$ and $(\phi(1), \phi(2)) = (0.08, 0.04)$, i.e. $k = 2$, we can compare the effect of constant priority of some flows to the results for elastic flows of the previous section.

Figure 6.14 illustrates many features of differentiation by priorities and marking. We notice first of all, the difference between *per flow* marking and *per packet* marking.

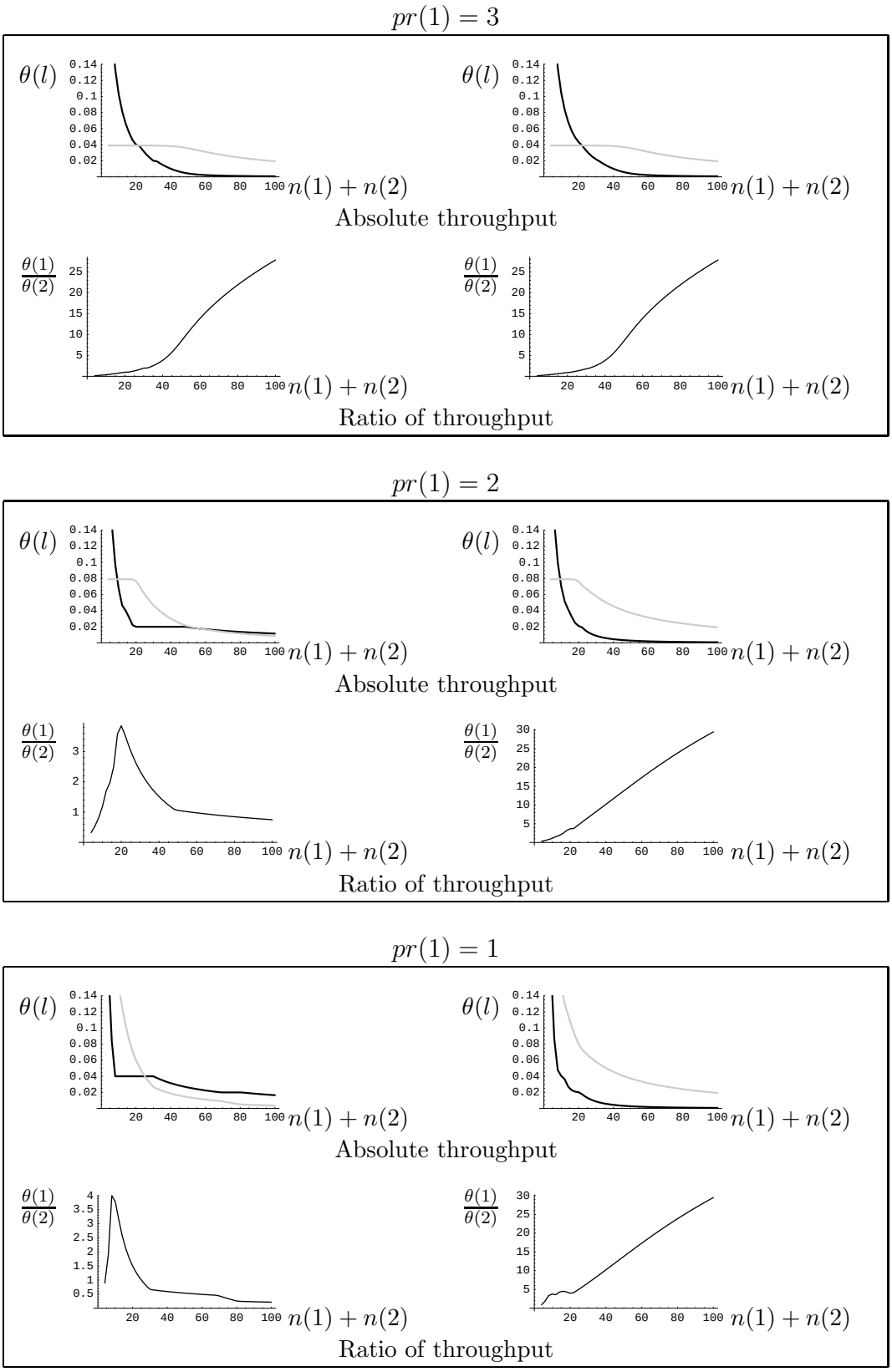



Figure 6.14: One buffer with TCP and non-TCP traffic. Absolute and relative bandwidth allocation for flows when $k = 2$ as a function of the total number of flows under the assumption $n(1) = n(2)$, $I = 3$. $\theta(1)$ gray, $\theta(2)$ black.

When the non-TCP traffic has the highest priority, there is no visible difference between the marking schemes. Furthermore, when the aggressive flow has the highest priority, there is no way of limiting its sending rate. This is the main difference between TCP and non-TCP flows, as can be seen by comparing figure 6.13 and 6.14.

However, when the non-TCP flow has a priority less than $I$, only *per flow* marking is able to constraint the bandwidth received by the aggressive flow. For *per packet* marking there is no visible difference regardless of the priority of the non-TCP flow, especially when the network is heavily congested.

The pictures for *per flow* marking also show that the non-TCP flows do not always receive more bandwidth than the TCP flows. In fact, only when the fixed priority of the non-TCP flows is optimal in relation to the load of the network, the non-TCP flows receive more bandwidth then the TCP flows. In times of low load it is optimal to have low priority and in times of high load to have high priority. The TCP flow is able to adjust its priority to differ from the priority of the aggressive flows and therefore only when the constant priority coincides with the load level of the network, does the aggressive flow get more than the TCP flow.

Finally, given that there are enough priority levels so that the priority of the aggressive flow is less than $I$, we notice that though *per flow* marking is able to constrain the bandwidth share of the aggressive flow, the maximum bandwidth share is twice as much as the ratio of weights indicates. The explanation for this is that the TCP flow protects itself from being starved when its priority is one higher than that of the non-TCP flow. The TCP flow must thus halve its sending rate and as the ratio of weights between the TCP and non-TCP is $1/k = 1/2$ the overall ratio is

$$r = \frac{t(1,i)}{t(2,i+1)} = 2\frac{t(1,i)}{t(2,i)} = 2k.$$

This means that though the ratio is not $k$, with *per flow* marking it is still proportional to $k$.

Figure 6.15 shows the ratio of sending rate in the case of 66% TCP traffic and 33% non-TCP traffic. Here the trajectories are solid, gray, and dashed for $\nu(1) = 0.039$, $\nu(1) = 0.079$, and $\nu(1) = 0.16$, respectively. Thus compared to figure 6.14 each case $pr(1) = 1$, $pr(1) = 2$ and $pr(1) = 3$ are depicted in the same figure. This scenario will be replicated for the case of two delay classes in the next section.

### 6.3.3 Two buffers with nrt TCP traffic and rt UDP traffic

The two buffer model is used to study the combined effect of the three degrees of freedom introduced in the text: marking, weighted capacity, and discarding thresholds. Furthermore, we can compare the result with the one delay class, to see if our conjecture on the difference of marking schemes holds, and how the weighted capacity and discarding affects these results.

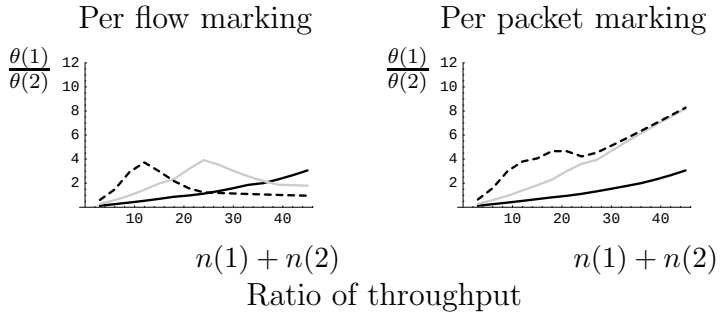



Figure 6.15: One buffer with 66% TCP and 33% non-TCP traffic. Relative bandwidth allocation for flows when $k = 2$ as a function of the total number of flows under the assumption $n(1)/n(2) = 2$, $I = 3$.

The elastic traffic serviced by the nrt buffer is TCP traffic modelled by the equations of section 6.2.1, while the streaming flows serviced by the rt buffer send non-TCP traffic at a constant intensity. Thus the equilibrium sending rate $\nu(l)$ is only solved for the elastic flows, with constant background traffic produced by the non-TCP flows. Notice however, that the priority level of the non-TCP flows is still determined by the marker.

We compare the effect of dividing the link capacity by using the following three scenarios, priority queuing ($w^1 = 1, w^2 = 0$), equal sharing ($w^1 = w^2 = 0.5$) and unequal sharing ($w^1 = 0.75, w^2 = 0.25$). Discarding can be independent or dependent. For the dependent discarding we use the threshold function depicted by the $7 \times 7$ matrix of equation (6.17) derived from equation (4.8), also used in [LSLH00],

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 2 & 2 & 3 \\ 1 & 1 & 1 & 1 & 2 & 2 & 3 \\ 1 & 1 & 1 & 2 & 2 & 2 & 3 \\ 1 & 1 & 2 & 2 & 2 & 3 & 3 \\ 2 & 2 & 2 & 2 & 3 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 \end{pmatrix}. \tag{6.17}$$

The square matrix is scaled to the appropriate buffer spaces so that the elements of the matrix correspond to the lowest priority level that is accepted to the system for each buffer system state.

We have the following scenario in terms of the free parameters: $\mu = 1$, $RTT = 1000/\mu$, $K^1 = 13$, $K^2 = 39$ and $I = 3$. We have two flow groups, $L = 2$, one group of non-TCP flows with $\phi(1) = 0.08$ and one group of TCP flows with $\phi(2) = 0.04$. The ratio between the weights is $k = 2$. The equilibrium sending rate is only solved for TCP flows, $l = 2$. The non-TCP flows have a fixed sending rate $\nu(1)$ of 0.039, 0.079, and 0.16 chosen so that under the *per flow* marking scheme the flows, when $I = 3$, are assigned priorities $pr(1) = 3$, $pr(1) = 2$, and $pr(1) = 1$, respectively. In the figures to follow the trajectories are solid, gray, and dashed for $\nu(1) = 0.039$, $\nu(1) = 0.079$, and $\nu(1) = 0.16$, respectively.

Each set of pictures depicted in figure 6.16 shows the ratio $\frac{\theta(1)}{\theta(2)} = \frac{\nu(1)(1-q(1))}{\nu(2)(1-q(2))}$

between throughputs of flows as a function of the total number of flows, under the condition $n(1)/n(2) = 1/2$. In figure 6.17, we have $n(1) = n(2)$ and in figure 6.18, we have $n(1)/n(2) = 2$. From these figures, we notice that figure 6.15 depicting the same scenario for one delay class roughly corresponds to two delay classes with dependent discarding.

When priority queuing is used to schedule traffic, only with *per flow* marking and dependent discarding TCP flows can be protected from bandwidth exhaustion by the non-TCP flows. However, due to priority queuing the streaming traffic that is admitted to the system is serviced with low delay and jitter.

From the pictures, one notices that marking all packets of the flow to the appropriate priority level encourages the TCP mechanism to optimize the sending rate according to the state of the network. If the link is congested by streaming traffic, the responsive flows can attain a higher priority level by dropping their sending rate. Then due to the dependent discarding the TCP traffic is admitted to the system as its priority level is high enough, and the non-responsive flows sending at a rate too high compared to the congestion of the link are discarded. Only when the non-TCP flows are sending at a rate low enough to attain highest priority are they able to use more than their fair share of the bandwidth. By having enough priority levels, i.e. more than three, this effect is also diminished.

From figure 6.16, we also notice that the use of dependent discarding controls the throughput of non-responsive flows better than independent discarding. The effect is mainly due to the fact that under the fixed threshold mechanism when the nrt buffer is congested, packets in the rt buffer are also discarded to alleviate the congestion.

By giving some minimum weight to the nrt buffer, the TCP traffic can be protected from bandwidth exhaustion by the non-TCP flows. However, there is no clear dependency between the ratio of weights of the scheduler ($w^1/w^2$) and the ratio of weights of the flow ($\phi(1)/\phi(2)$).

Furthermore, the first three identical pairs of pictures demonstrate that when independent discarding is used only the change of scheduler weights affects the division of bandwidth between delay classes.

The main conclusion from the figures 6.16 – 6.18 is that only when both *per flow* marking and dependent discarding are used, does the share of throughput obtained by the streaming traffic depend on the link state. The dependency means that when there are few users on the link, it is optimal for the streaming traffic to send $\nu(1) = 0.16$ and have its packets marked to the lowest priority level. As the number of users increases it would be optimal for the streaming traffic to drop its sending rate to, e.g. $\nu(1) = 0.079$, and as the number of users further increases it would be optimal to send at the highest priority level, with intensity $\nu(1) = 0.039$. In all other cases depicted in the figure, it is always optimal to send as much as possible, even if all or some of the packets are then marked to the lowest priority level. This means that the other combinations of the mechanisms are not able to force or give an incentive to the user to adjust the sending rate according to the state of the network. The use of *per flow* marking

and dependent thresholds thus gives a powerful incentive for flows to be TCP friendly [FF99].

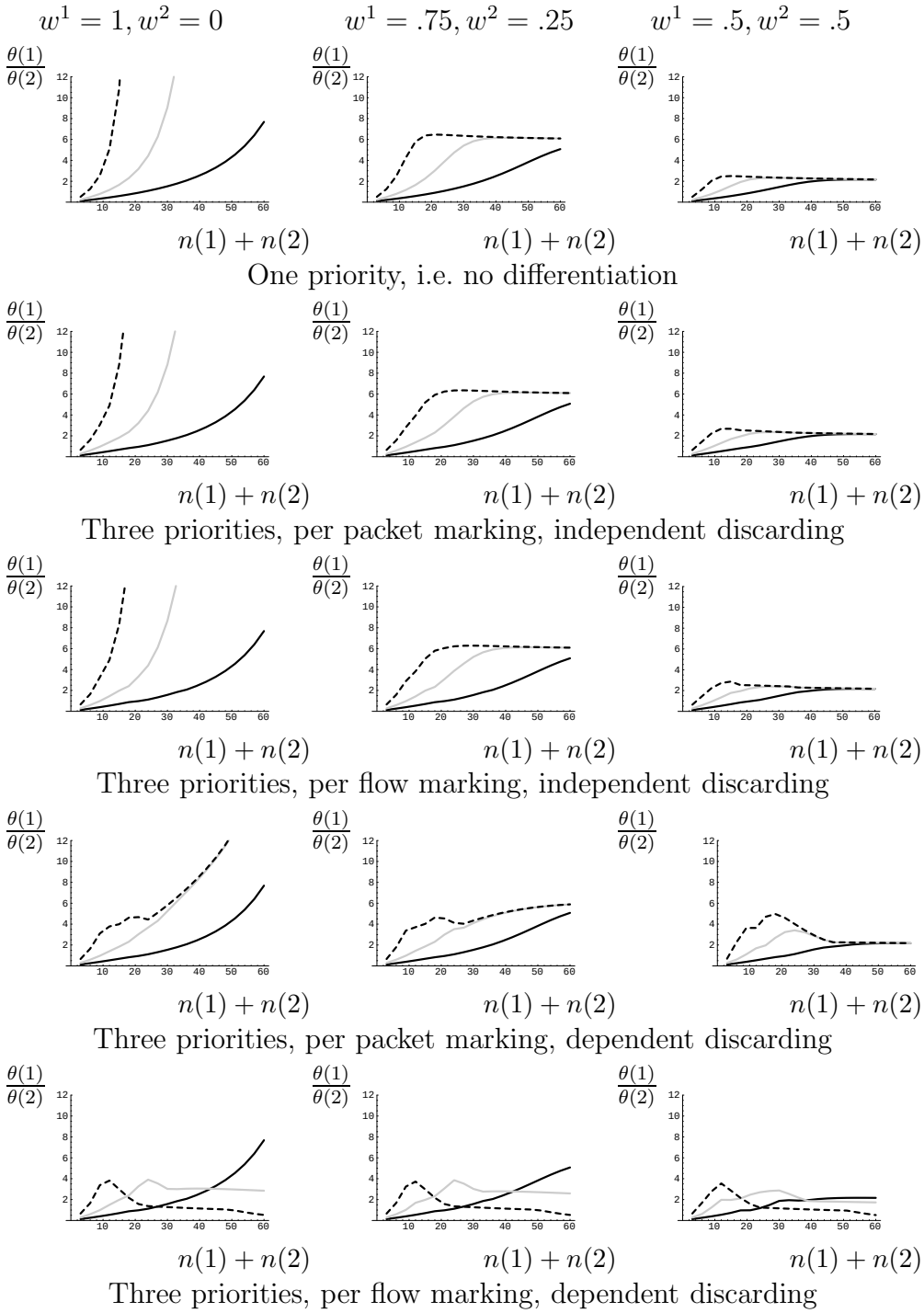



Figure 6.16: Effect of marking and discarding when the minimum weights of the rt buffer and nrt buffer change. 66% of the flows are TCP and 33% non-TCP.

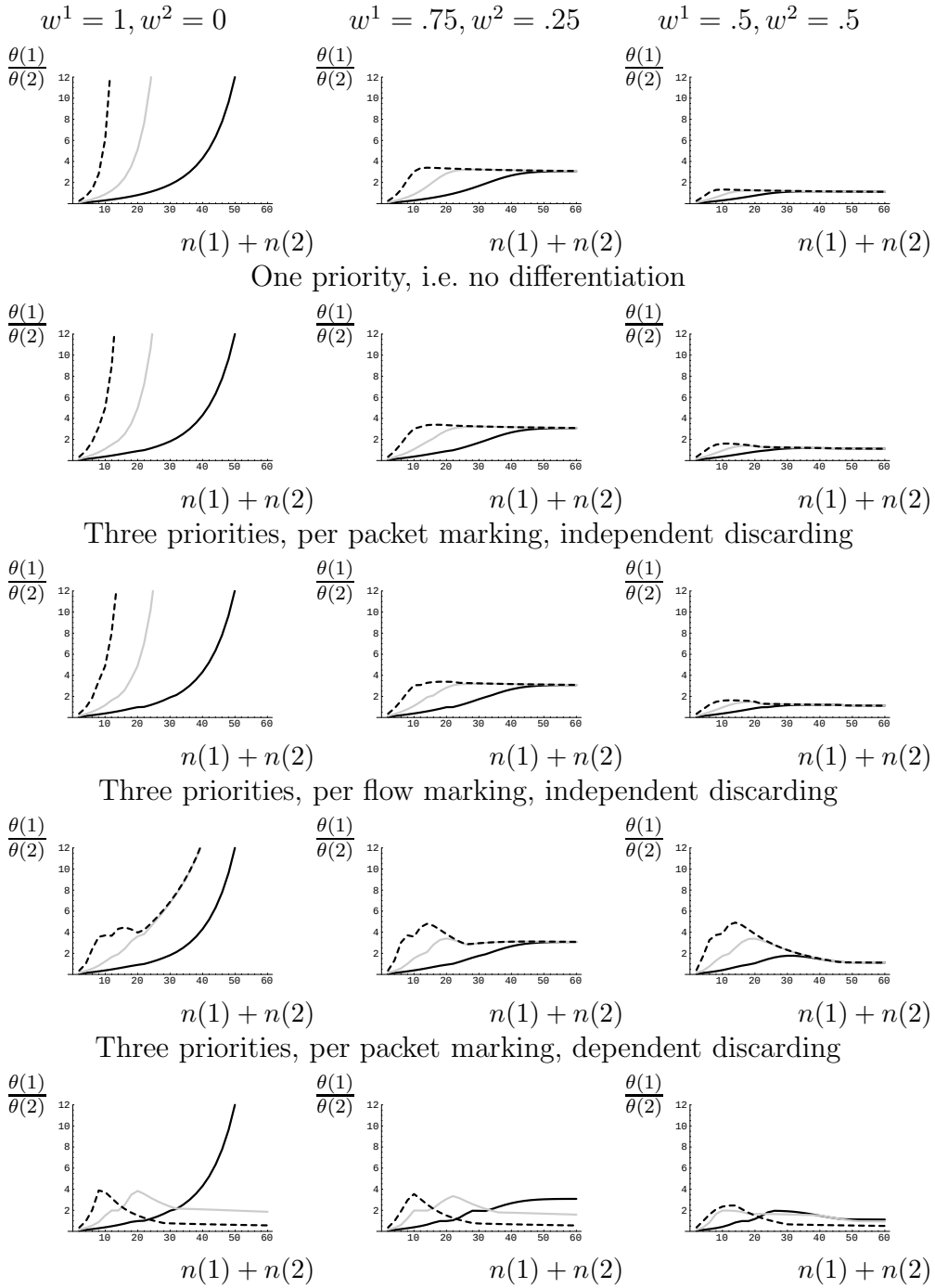



Figure 6.17: Effect of marking and discarding when the minimum weights of the rt buffer and nrt buffer change. 50% of the flows are TCP and 50% non-TCP.

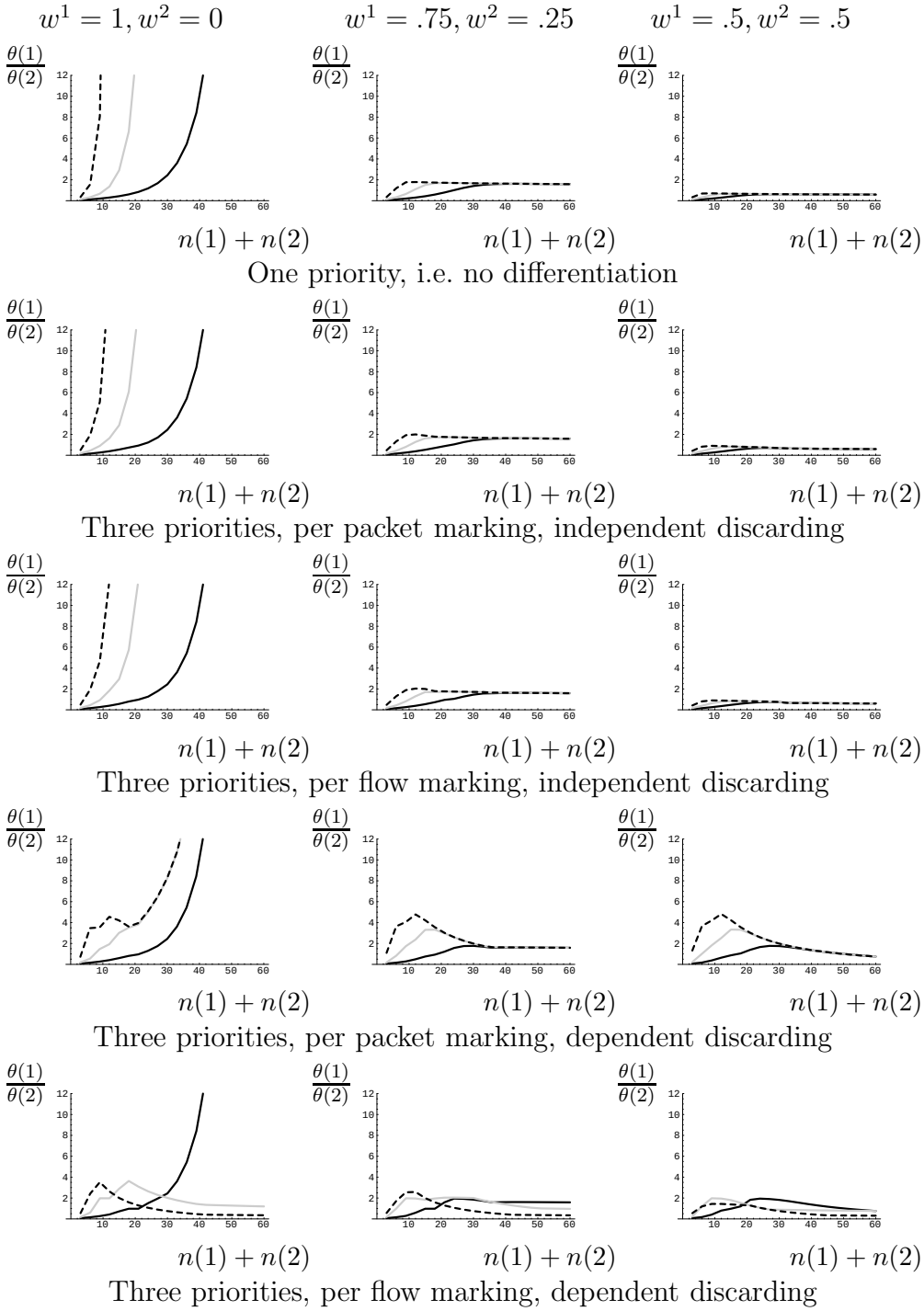



Figure 6.18: Effect of marking and discarding when the minimum weights of the rt buffer and nrt buffer change. 33% of the flows are TCP and 66% non-TCP.

# Chapter 7

# Simulations

We now deepen the study of the previous chapters to include simulations, with special considerations on the effect on differentiation that the metering mechanisms at the boundary nodes have. To this end, we consider the network model of chapter 4: a single bottleneck link of a DiffServ cloud loaded by a fixed number of elastic TCP sources in congestion avoidance mode and with a similar round trip time (RTT). At the boundary of the DiffServ cloud, the traffic of these TCP sources is conditioned, i.e. metered and marked, whereas inside the DiffServ node it is forwarded or discarded by a scheduling unit that includes a single buffer with multiple priority levels.

The two metering and marking mechanisms compared are token bucket and exponential weighted moving average (EWMA). We validate the earlier hypothesis used in our analytical models that the metering and marking result of EWMA, the so called momentary bit rate, can be used to mark packets *per flow*, while the token bucket is a *per packet* metering and marking mechanism.

In EWMA, the parameter $\alpha$ adjusts the memory of the mechanism. Marking is performed based on predefined thresholds of the momentary bit rate. The token bucket mechanism is implemented as a cascaded system of many token buckets; see [HG99b] for an example. Each bucket has the same capacity $c$, but a specified rate in accordance to the marking thresholds of the EWMA system. Using the simulation model, we study the effect that the parameter values have on the metering and marking result and compare them to the assumptions of the analytical model.

As a result, we give further hindsight to the effect that various DiffServ traffic-handling mechanisms have on the fairness perceived by the user. We continue to use the division of bandwidth and the division of packets into priority levels as the decisive factors for our results.

## 7.1 Simulation setting

A single bottleneck link of a DiffServ cloud is loaded by a fixed number $n$ of greedy TCP sources with a similar round trip time (RTT). In the simulations, the RTT may be fixed or random. We model the TCP flows in the congestion avoidance phase, i.e. whenever a packet is lost, the corresponding source halves its window size; otherwise it is linearly increased.

We have the same notation as before. All flows belong to the same delay class and are divided into groups $l \in \mathcal{L}^2$ according to the weight $\phi(l)$. Each group consists of $n(l)$ identical flows. Packets of a flow are marked to priority levels $i \in \mathcal{I} = \{1, ..., I\}$ according to the thresholds $t(l, i)$.

We wish to study the effect of the metering mechanisms and the time parameters on the system. The evaluation is made based on the resulting bandwidth allocation and division into priority classes. We again consider differentiation as a function of the number of sources, as opposed to a function of the load of the network. This is because we are considering elastic sources that adjust their sending rate, and thus adjust the load, according to the congestion of the network.

The marking mechanism assigns a priority level to the flow or its packets. Then, depending on the congestion of the link, some packets may be discarded, namely when the priority of the packet is less than the threshold priority of the scheduler. The TCP mechanism adjusts its window size according to the feedback signal from the buffer system. By gathering information on how the stable sending rate and corresponding priority level allocation depends on the number of flows, we can study the differences between the marking schemes. Figure 7.1 summarizes the simulation setting.

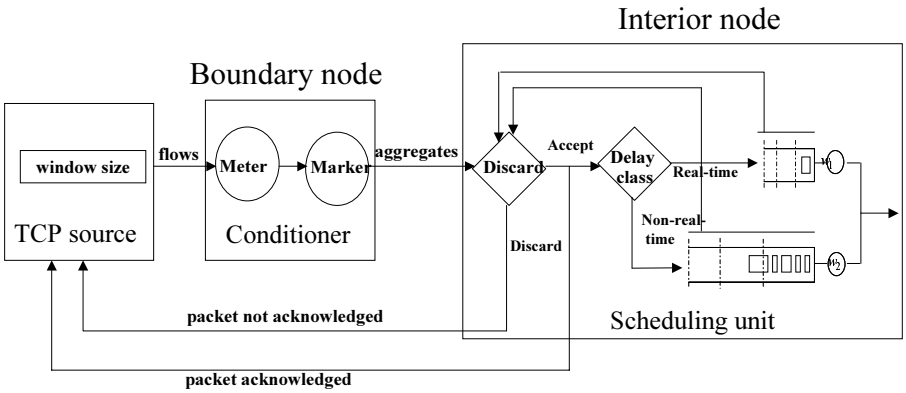



Figure 7.1: The simulation setting

### 7.1.1 Metering flows at the conditioner

Flows are metered at the boundary of the DiffServ cloud either using token buckets or exponential weighted moving average (EWMA) of the bit rate. The time

parameters $c$ and $\alpha$ must be in accordance with the round trip time $RTT$ of the packets.

In order to compare the two metering principles and the time parameters $c$ and $\alpha$, we fix the rate $r(i)$ of token bucket $i$ to correspond to the threshold $t(i)$. As the thresholds are also functions of the flow group $l$, the rate of the token bucket is also defined for each flow group. We thus have

$$r(l,i) = t(l,i) = \phi(l)a(i), i = 1, ..., I-1,$$

where

$$a(i) = 2^{-i+I/2-0.5}.$$

We fix the thresholds of EWMA marking to the rates of the token buckets in order to model the relationship between relative services and the time parameters, $\alpha$ and $c$.

For the EWMA metering we calculate the momentary bit rate ($mbr$) of the flow as given by equation (4.2), using the following definition

$$\alpha = \frac{5}{KD},$$

where $K$ is the size of the buffer in the scheduling unit and $D$ is a free time parameter. The priorities are then determined based on the thresholds $t(l,i)$. The $j$:th packet of flow $k \in l$ has priority $i$, if

$$t(l,i) \leq mbr(k,j) < t(l,i-1).$$

### 7.1.2 TCP model

The TCP sources are modelled in congestion avoidance mode, adjusting their sending rate according to the feedback signal from the forwarding unit.

The round trip time is the time it takes for an acknowledgment to reach the TCP source after the packet has been sent. We let $RTT$ be random and from an exponential distribution.

The window size is initialized to $w = 1$. It refers to the number of unacknowledged packets that can be sent at a time. Once a packet is sent the counter *unack* is incremented by one. If the packet is accepted to the scheduling unit, that is it has a high enough priority not to be discarded, in $RTT/2$ time the acknowledgement reaches the source and the *unack* counter is decremented by one. The window size is updated to

$$\begin{aligned} inc &= 1/w, \\ w &= w + inc. \end{aligned}$$

The number of new packets sent after the update is $\lfloor w \rfloor - unack$.

On the other hand, if the priority of the packet is too low it is discarded and not accepted to the scheduling unit. After $RTT/2$ the information reaches the source

and the window size is halved and the *unack* counter is decremented by one. If the counter *unack* is 0 after the halving of the window, a new packet is sent. The window size is thus always at least 1.

### 7.1.3 Scheduling unit

The packets of the sources are forwarded inside the DiffServ node by a scheduling unit consisting of one FIFO buffer and exponentially distributed service times. The buffer space is divided by $I$ discarding thresholds. Denote the discarding threshold of priority level $i \in I$ by $K(i)$, with $K(I) = K$, the size of the buffer. If the buffer state is greater than $K(i)$ only packets with priority greater than $i$ are accepted to the scheduling unit, other packets are discarded.

## 7.2 Numerical results

The basic scenario is as follows: two flow groups, $L = 2$, three precedence levels, $I = 3$, $\mu = 1$ and mean $RTT = 50/\mu$, $100/\mu$ or $1000/\mu$. Set $K(1) = 22$, $K(2) = 33$ and $K = 39$ packets. Define $k = \phi(1)/\phi(2)$ as before.

We study the equilibrium bandwidth allocation in terms of the ratio of throughputs and ratio of sending rate between flows in the two groups. In the simulations, we consider the two metering mechanisms: token bucket and EWMA. In the analytical model, we have the token bucket mechanism modelled as *per packet* marking and EWMA modelled as *per flow* marking. We wish to validate the assumptions of the analytical model and study the effect of the time parameters $c$ and $\alpha$ using simulations.

### 7.2.1 EWMA parameter $\alpha$

The set of pictures depicted in figure 7.2 summarizes the relationship between $\alpha$ and $RTT$. Note that we vary the free parameter $D$ of $\alpha$. The relationship is shown in terms of the ratio of throughputs for flows with weights $(\phi(1), \phi(2)) = (0.08, 0.04)$, i.e. $k = 2$, as a function of number of flows $n(1)$ and $n(2)$. The black areas correspond to equal bandwidth allocation, the white areas correspond to bandwidth allocation at least equal to $k$, and the gray areas to bandwidth allocation between 1 and $k$. Table 7.1 gives the corresponding numerical values for ratios of throughput and table 7.2 the numerical values for ratios of sending rate.

From figure 7.2 we make the following observations on the EWMA principle and the parameter $D$.

1. The minimum ratio of throughputs and sending rate is always approximately 1.

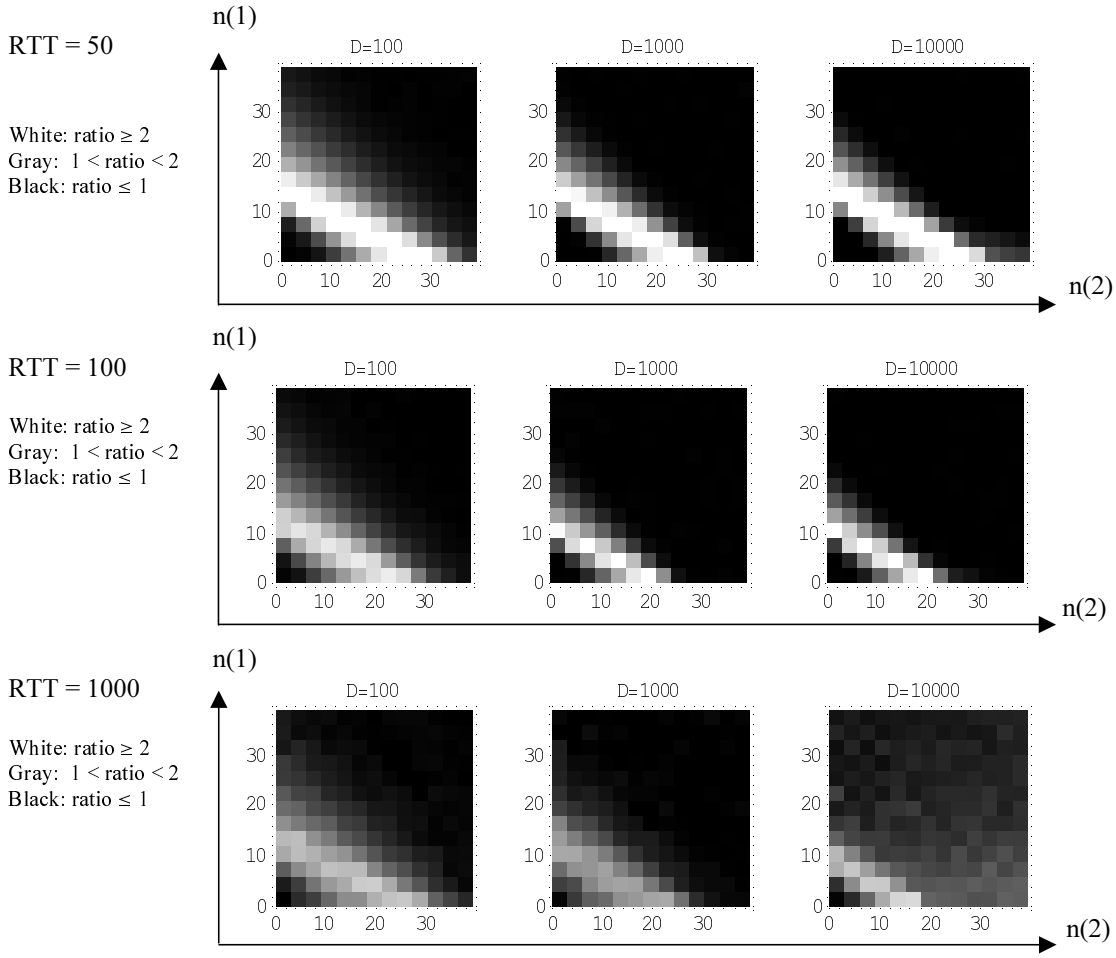



Figure 7.2: The effect of parameter $D$ on bandwidth division under *per flow* marking in the simulation setting

| $RTT$ | | $D = 100$ | $D = 1000$ | $D = 10000$ |
|---|---|---|---|---|
| 50 | Min | 0.978437 | 0.978525 | 0.988121 |
| | Max | 2.31008 | 2.48056 | 2.58277 |
| 100 | Min | 0.982882 | 0.983543 | 0.988843 |
| | Max | 1.92585 | 2.10297 | 2.16211 |
| 1000 | Min | 0.951966 | 0.95911 | 1.01888 |
| | Max | 1.80139 | 1.65922 | 1.84981 |

Table 7.1: Relationship between $\alpha$ and $RTT$, when $k = 2$ in terms of throughput

| $RTT$ | | $D = 100$ | $D = 1000$ | $D = 10000$ |
|---|---|---|---|---|
| 50 | Min | 0.988076 | 0.987168 | 0.993539 |
| | Max | 1.88536 | 1.98159 | 2.01127 |
| 100 | Min | 0.985907 | 0.986997 | 0.990926 |
| | Max | 1.79279 | 1.95817 | 2.01054 |
| 1000 | Min | 0.95285 | 0.959991 | 1.01887 |
| | Max | 1.79738 | 1.65611 | 1.84697 |

Table 7.2: Relationship between $\alpha$ and $RTT$, when $k = 2$ in terms of sending rate.

2. The maximum ratio of throughputs and sending rate is always of the same order as $k$. Thus we can deduce that the maximum bandwidth of a flow inside group $l$ is in proportion to price paid, i.e. in proportion to the weight $\phi(l)$.

3. There is a clear dependency between the two time parameters $D$ and $RTT$. As $RTT$ decreases, the area of differentiation widens and deepens, i.e. the maximum ratio increases. The free parameter $D$ is also sensitive to the time scale of the simulation, the $RTT$. If we wish that the maximum differentiation ratio is equal to the ratio $k$ of the weights and the minimum differentiation is not less than 1, we could then choose $D = 100$ when $RTT = 100$. On the other hand, when $RTT = 100$, the loss probabilities are not negligible and if we want the sending rate to have maximum ratio of $k$, we could choose, based on Table 7.2, $D = 1000$ or $D = 10000$. Furthermore, we also made simulations with $D = 1$ and practically no area of differentiation resulted. Therefore, it is essential that the parameter $D$ is not too small compared to $RTT$.

4. With three precedence levels, $I = 3$, only one area of maximum differentiation is obtained. As was shown by analytical results in chapter 6 and will be shown in section 7.2.3, there are two areas of differentiation when $I = 3$. However, by increasing the number of priority levels we are able to obtain a larger area of differentiation and more than one area of maximum differentiation. Figure 7.3 depicts the cases $I = 2$, 3, 6 and 8, with $RTT = 100$ and $D = 100$. Table 7.3 gives the corresponding values for the minimum and maximum ratios of throughputs, and we note that for $I = 6$, the minimum value of the ratio of throughputs is 1.45. Therefore, the flow group with higher weight always receives more bandwidth than the group with lower weight.

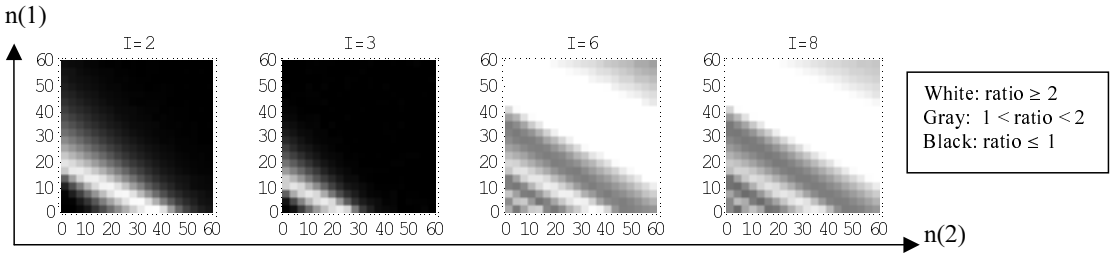



Figure 7.3: Effect of increasing number of priority levels $I$

| $I$ | Min | Max |
|---|---|---|
| 2 | 0.992562 | 1.98859 |
| 3 | 0.982882 | 1.92585 |
| 6 | 1.45289 | 2.39094 |
| 8 | 1.41926 | 2.39341 |

Table 7.3: Minimum and maximum ratios of throughputs for varying $I$

### 7.2.2 Token bucket parameter $c$

The set of figures depicted in figure 7.4 summarizes the relationship between $c$ and $RTT$. As with the EWMA principle, we study the ratio of throughputs for flows with weights $(\phi(1), \phi(2)) = (0.08, 0.04)$, i.e. $k = 2$, as a function of number of flows $n(1)$ and $n(2)$. The black areas correspond to equal bandwidth allocation, the white areas correspond to bandwidth allocation at least equal to $k$, and the gray areas to bandwidth allocation between 1 and $k$. Table 7.4 gives the corresponding numerical values for ratios of throughput and table 7.5 the numerical values for ratios of sending rate.

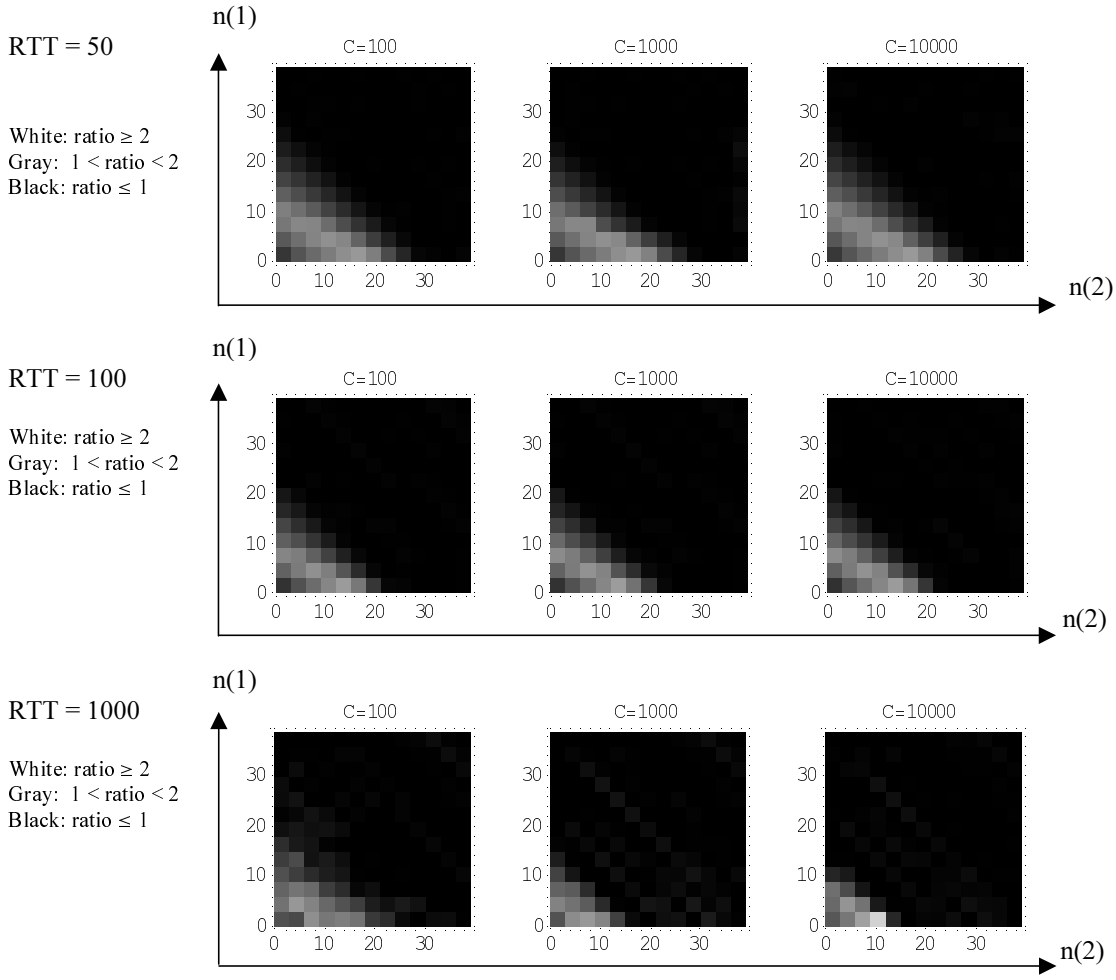



Figure 7.4: The effect of parameter $c$ under *per packet* marking, in the simulation setting

From Figure 7.4 we make the following observations on the token bucket principle and the parameter $c$.

1. The minimum ratio of throughputs and sending rate is approximately 1, though not as clearly as with the EWMA principle.

2. The maximum ratio of throughputs and sending rate is clearly less than $k$.

| $RTT$ | | $c = 100$ | $c = 1000$ | $c = 10000$ |
|---|---|---|---|---|
| 50 | Min | 0.991143 | 0.990008 | 0.982195 |
| | Max | 1.59379 | 1.59379 | 1.60033 |
| 100 | Min | 0.961958 | 0.966481 | 0.976683 |
| | Max | 1.60769 | 1.61122 | 1.60857 |
| 1000 | Min | 0.926936 | 0.904089 | 0.937499 |
| | Max | 1.59202 | 1.59522 | 1.82334 |

Table 7.4: Relationship between $c$ and $RTT$, when $k = 2$ in terms of throughput

| $RTT$ | | $c = 100$ | $c = 1000$ | $c = 10000$ |
|---|---|---|---|---|
| 50 | Min | 0.993397 | 0.993311 | 0.988377 |
| | Max | 1.47559 | 1.47559 | 1.48043 |
| 100 | Min | 0.967552 | 0.970519 | 0.979241 |
| | Max | 1.55598 | 1.56103 | 1.55799 |
| 1000 | Min | 0.927288 | 0.904419 | 0.93769 |
| | Max | 1.5914 | 1.59459 | 1.82228 |

Table 7.5: Relationship between $c$ and $RTT$, when $k = 2$ in terms of sending rate

A flow inside group $l$ does not receive maximum bandwidth in proportion to price paid, i.e. in proportion to the weight $\phi(l)$, to the same extent as with the EWMA principle.

3. There is dependency between the time parameters $c$ and $RTT$. The main difference occurs, when $RTT = 1000$ and $c$ changes from 1000 to 10000. Furthermore, simulations made with $c = 5$ and $RTT = 100$ showed that the area of differentiation reduces as $c$ decreases. Therefore, it is essential that the parameter $c$ is not too small compared to $RTT$.

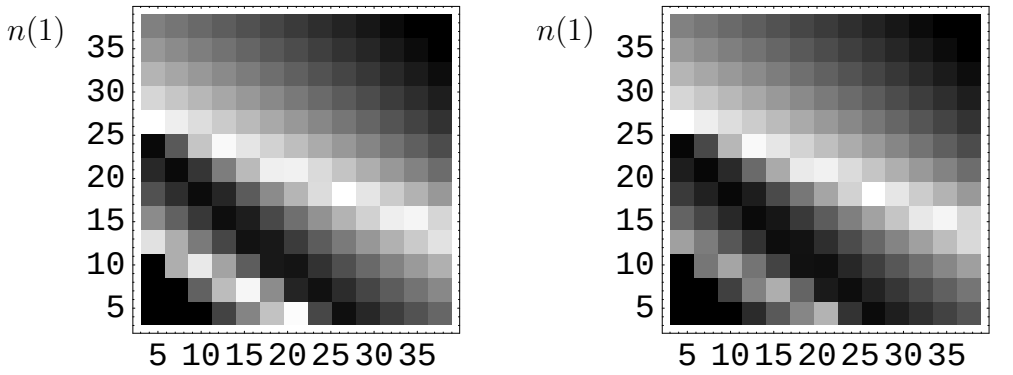
Though corresponding figures are not included in this work, we have simulated the token bucket principle with more than three precedence levels. As a result, the areas of differentiation increased as $I$ increased in the same way as in Figure 7.3.

### 7.2.3 Analytical results

The analytical packet level model of chapter 6 gives us the equilibrium sending rate $\nu(l)$ and the corresponding throughput $\theta(l) = \nu(l)(1 - q(l))$ for each flow group. Using the ratio of throughput, we can study the effect that marking has in dividing bandwidth between elastic flows and compare the results given by the simulations. The case $RTT = 1000$ was already studied in chapter 6, here we present the results for $RTT = 100$ and compare the simulation results to the results given by the packet level model.

In Figure 7.5 marking is *per flow*, while in Figure 7.6 marking is *per packet*. Both figures show the ratio, $r_\nu = \nu(1)/n(2)$, of sending rate and the corresponding ratio of throughputs $r_\theta = \theta(1)/\theta(2)$ for flows with weights $(\phi(1), \phi(2)) = (0.08, 0.04)$,

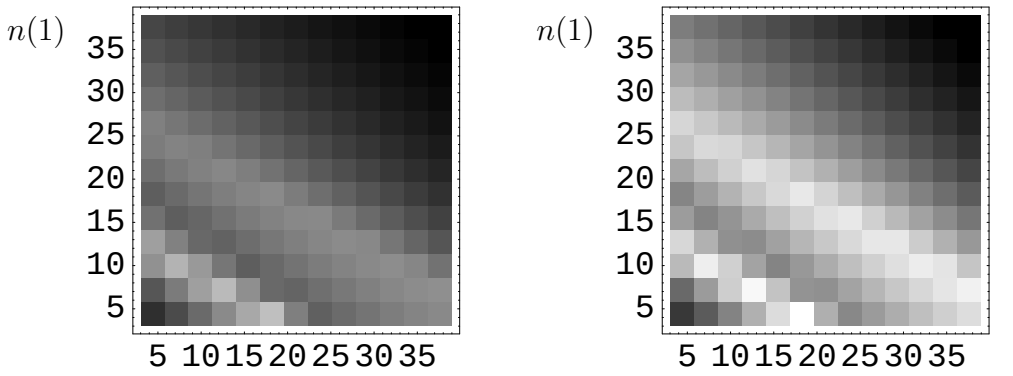
i.e. $k = 2$, as a function of number of flows $n(1)$ and $n(2)$. The black areas correspond to equal bandwidth allocation, the white areas correspond to bandwidth allocation at least equal to $k$ and the gray areas to bandwidth allocation between 1 and $k$.




Sending rate
$\min(r_\nu) = 1$, $\max(r_\nu) = 2$

Throughput
$\min(r_\theta) = 1$, $\max(r_\theta) = 2.66666$

Figure 7.5: *Per flow marking*: Relative bandwidth allocation $r_\nu = \nu(1)/\nu(2)$ and $r_\theta = \theta(1)/\theta(2)$ for flows with $k = 2$ as a function of number of flows. White: $r \geq k$, black: $r \leq 1$, grey: $1 < r < k$.




Sending rate
$\min(r_\nu) = 1$, $\max(r_\nu) = 1.74732$

Throughput
$\min(r_\theta) = 1$, $\max(r_\theta) = 1.85529$

Figure 7.6: *Per packet marking*: Relative bandwidth allocation $r_\nu = \nu(1)/\nu(2)$ and $r_\theta = \theta(1)/\theta(2)$ for flows with $k = 2$ as a function of number of flows. White: $r \geq k$, black: $r \leq 1$, grey: $1 < r < k$.

The following observations made in chapter 6 can be made on figures 7.5 and figures 7.6 and will be reviewed here:

1. When the marking is *per flow* a maximum ratio of $k$ is achieved. On the other hand, when the marking is *per packet* the maximum bandwidth ratio $\nu(1)/\nu(2)$ is less than $k$. The difference stems from the fact that with *per packet* marking some packets are always also marked to highest precedence.

2. Bandwidth is divided in equal proportions more often with the *per flow* marking scheme than with the *per packet* marking scheme. When only the overflow packets are marked to lower precedence levels, the rest of the packets have highest or medium precedence and the ratio is higher than 1.

## 7.3 Metering mechanisms compared

The simulation and analytical results are the same qualitatively, though they differ a bit quantitatively. The following similarities are observed:

1. Both models result in bandwidth divisions with similar minimum and maximum values, as long as the parameters $D$ and $c$ used in the simulations are large enough compared to RTT.

2. Furthermore, simulations done using the token bucket mechanism give, in terms of minimum and maximum bandwidth division, the same results as the analytical study for *per packet* marking. The same is observed for simulations using the EWMA principle and analytical results using *per flow* marking.

3. The number of differentiation areas increases as $I$ increases as shown by simulations in figure 7.3 and by analytical results in earlier sections.

The following disparity is observed:

1. There are a different number of differentiation areas. For the analytical model there are always $I - 1$ differentiation areas, while the simulations resulted in only one area of differentiation for $I = 3$.

The results thus confirm our hypothesis that the token bucket and EWMA methods can be modelled as *per packet* and *per flow* metering and marking mechanisms, respectively. Furthermore, the analytical results are qualitatively consistent with the simulation outcome and can be used to understand how differentiation mechanisms should be designed.

# Chapter 8

# Conclusions

Some form of differentiation is needed in networks that service traffic with different quality requirements. We have chosen to divide flows into two delay classes: one for elastic traffic and the other for non-elastic traffic. However, in order to control the use of scarce resources, in this case bandwidth, we need to further divide the traffic into priority levels based on a contracted rate, to which a price may be associated.

If Differentiated Services (DiffServ) is the Quality of Service (QoS) architecture used and no admission control is employed, then relative services is a better service objective than assured services. In assured services the flow should receive a rate at least equal to its contracted rate, while in relative services the rate of the flow should be proportional to the contracted rate.

Previous work has already shown that assured services cannot be met without admission control. We show that relative services can be achieved in a DiffServ network given that enough priority levels are used.

The key differentiation mechanisms are *per flow* marking and dependent discarding. *Per flow* marking is achieved using a traffic meter based on exponential moving average of the sending rate, given that the metering time parameter is of the same magnitude as the round trip time of the flow. Dependent discarding means that the dropping thresholds of a scheduling unit take into account the buffer state of all the queues in the scheduling unit.

We showed, that in the case of two delay classes, the real time traffic and elastic traffic discarding thresholds of a priority level have to be dependent and take into account the total traffic inside a scheduler. This prevents non-conforming UDP traffic from exhausting the bandwidth at the expense of conforming TCP traffic. Furthermore, using dependent discarding together with *per flow* marking encourages all flows regardless of their congestion control mechanism to conform their sending rate to the load level of the network. The weights of a WFQ scheduler and corresponding buffer sizes are then more relevant in bounding the delay than in dividing the throughput between delay classes.

The congruence between the flow level models, packet level models and simula-

tions shows that each modelling approach can be used to assess the differentiation mechanisms. The choice of the modelling level depends on what aspects and mechanisms need to be evaluated.

Further research on the topic presented in the thesis include a dynamic flow model, where the number of flows varies randomly, and including short-lived TCP flows. In the packet level model, considering stability questions of the interaction between TCP congestion control and DiffServ mechanisms would require extending the TCP model to a more detailed and dynamic traffic model. In the flow level models, considering larger networks would be an interesting topic for further research on TCP fairness in differentiated networks.

# Bibliography


[AAB00] Eitan Altman, Kostya Avrachenkov, and Chadi Barakat. A stochastic model of TCP/IP with stationary random losses. In *Proceedings of ACM SIGCOMM*, pages 231–242, 2000.

[ADG+00] M. Allman, S. Dawkins, D. Glover, J. Griner, D. Tran, T. Henderson, J. Heidemann, J. Touch, H. Kruse, S. Ostermann, K. Scott, and J. Semke. *Ongoing TCP Research Related to Satellites*, February 2000. RFC 2760.

[AN02] Samuli Aalto and Eeva Nyberg. Flow level models of diffserv packet level mechanisms. In P. Lassila, E. Nyberg, and J. Virtamo, editors, *Proceedings of the Sixteenth Nordic Teletraffic Seminar, NTS 16*, pages 194–205, Espoo, Finland, August 2002.

[APS99] M. Allman, V. Paxson, and W. Stevens. *TCP Congestion Control*, April 1999. RFC 2581.

[Bar01] Chadi Barakat. TCP/IP modeling and validation. *IEEE Network*, 15(3):38–46, May 2001.

[BBC+98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. *An Architecture for Differentiated Service*, December 1998. RFC 2475.

[BCS94] R. Braden, D. Clark, and S. Shenker. *Integrated Services in the Internet Architecture: an Overview*, June 1994. RFC 1633.

[BF01] H. Balakrishnan and S. Floyd. *Enhancing TCP's Loss Recovery Using Limited Transmit*, January 2001. RFC 3042.

[BM01] Thomas Bonald and Laurent Massoulié. Impact of fairness on Internet performance. In *Proceedings of ACM SIGMETRICS*, pages 82–91, 2001.

[DSR99] Constantinos Dovrolis, Dimitrios Stiliadis, and Parameswaran Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. In *Proceedings of ACM SIGCOMM*, pages 109–120, 1999.

[dVLK01] G. de Veciana, T.-J. Lee, and T. Kontantopoulos. Stability and performance analysis of networks supporting elastic services. *IEEE/ACM Transactions on Networking*, 9(1):2–14, February 2001.

[ECP] O. Elloumi, S. De Cnodder, and K. Pauwels. Usefulness of three drop precedences in Assured Forwarding service. IETF Draft July 1999.

[FBAPR01] S. Ben Fredj, T. Bonald, G. Régnié A Proutiere, and J.W. Roberts. Statistical bandwdith sharing: A study of congestion at flow level. In *Proceedings of ACM SIGCOMM*, pages 111–122, August 2001.

[FF99] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

[FH99] S. Floyd and T. Henderson. *The NewReno Modification to TCP's Fast Recovery Algorithm*, April 1999. RFC 2582.

[Flo91] S. Floyd. Connections with multiple congested gateways in packet-switched networks part 1: One-way traffic. *Computer Communication Review*, 21(5):30–47, October 1991.

[Flo01] Sally Floyd. A report on recent developments in TCP congestion control. *IEEE Communications Magazine*, 39(4):84–90, April 2001.

[FMMP00] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. *An Extension to the Selective Acknowledgement (SACK) Option for TCP*, July 2000. RFC 2883.

[GDJL] M. Goyal, A. Durresi, R. Jain, and C. Liu. Performance analysis of Assured Forwarding. IETF Draft October 1999.

[GK99] R. J. Gibbens and F. P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 35:1969–1985, 1999.

[HBWW99] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. *Assured Forwarding PHB Group*, June 1999. RFC 2597.

[HG99a] J. Heinanen and R. Guerin. *A Single Rate Three Color Marker*, September 1999. RFC 2697.

[HG99b] J. Heinanen and R. Guerin. *A Two Rate Three Color Marker*, September 1999. RFC 2698.

[HKL+00] J. Harju, Y. Koucheryavy, J. Laine, S. Saaristo, K. Kilkki, J. Ruutu, H. Waris, J. Forsten, and J. Oinonen. Performance measurements and analysis of TCP flows in a differentiated services WAN. In *Proceedings of the Local Computer Networks conference (LCN 2000), Tampa, Florida, USA*, pages 1 – 10, November 2000.

[JNP99] V. Jacobson, K. Nichols, and K. Poduri. *An Expedited Forwarding PHB*, June 1999. RFC 2598.

[Kel97] F. Kelly. Charging and rate control for elastic traffic. *Eur. Trans. Telecommun*, 8:33–37, 1997.

[Kel99] F. Kelly. Mathematical modelling of the Internet. In *Proc. of Fourth International Congress on Industrial and Applied Mathematics*, pages 105–116, 1999.

[Kil97] K. Kilkki. Simple Integrated Media Access. Available at http://www-nrc.nokia.com/sima, 1997.

[KMT98] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 15(49):237–255, 1998.

[KP91] P. Karn and C. Partridge. Improving round-trip time estimates in reliable transport protocols. *ACM Transactions on Computer Systems*, 9(4):364–373, November 1991.

[KR98] K. Kilkki and J. Ruutu. Simple Integrated Media Access (SIMA) with TCP. In *the 4th INFORMS Telecommunications conference Boca Raton, FL, USA*, March 1998.

[LSLH00] J. Laine, S. Saaristo, J. Lemponen, and J. Harju. Implementation and measurements of simple integrated media access (SIMA) network nodes. In *Proceedings for IEEE ICC 2000*, pages 796–800, June 2000.

[MBDM99] M. May, J. Bolot, C. Diot, and A. Jean Marie. Simple performance models for Differentiated Services schemes for the Internet. In *Proceedings of IEEE INFOCOM*, pages 1385–1394, March 1999.

[MMFR96] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. *TCP Selective Acknowledgement Options*, October 1996. RFC 2018.

[MR99] L. Massoulié and J. Roberts. Bandwidth sharing: Objectives and algorithms. In *Proceedings of IEEE INFOCOM*, pages 1395–1403, 1999.

[MSMO97] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review*, 27(3), July 1997.

[MW00] Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, 2000.

[NA02] Eeva Nyberg and Samuli Aalto. How to achieve fair differentiation. In *Networking 2002*, pages 1178–1183, Pisa, Italy, May 2002. Springer-Verlag.

[NAS02] Eeva Nyberg, Samuli Aalto, and Riikka Susitaival. A simulation study on the relation of DiffServ packet level mechanisms and flow level QoS requirements. In *International Seminar, Telecommunication Networks and Teletraffic Theory*, St. Petersburg, Russia, January 2002.

[NAV01] Eeva Nyberg, Samuli Aalto, and Jorma Virtamo. Relating flow level requirements to DiffServ packet level mechanisms. Technical Report TD(01)04, COST279, October 2001.

[PFTK98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. In *Proceedings of ACM SIGCOMM*, pages 303–314, 1998.

[PG93] Abhay K. Parekh and Robert G. Gallagher. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, 1993.

[Pos81] J. Postel. *Transmission Control Protocol*, September 1981. RFC 793.

[PSN99] P. Pieda, N. Seddigh, and B. Nandy. The dynamics of TCP and UDP interaction in IP-QoS Differentiated Service networks. In *3rd Canadian Conference on Broadband Research (CCBR)*, November 1999.

[RF99] K. Ramakrishnan and S. Floyd. *A Proposal to Add Explicit Congestion Notification (ECN) to IP*, January 1999. RFC 2481.

[Sch99] G. Schultz. A simulation study if the SIMA priority scheme. Unpublished, 1999.

[SNT+00] S. Sahu, P. Nain, D. Towsley, C. Diot, and V. Firoiu. On achievable service differentiation with token bucket marking for TCP. In *Proceedings ACM SIGMETRICS'00*, pages 23–33, June 2000.

[SPG97] S. Shenker, C. Partridge, and R. Guerin. *Specification of Guaranteed Quality of Service*, September 1997. RFC 2212.

[STK99] S. Sahu, D. Towsley, and J. Kurose. A quantitative study of differentiated services for the Internet. In *Proc. IEEE Global Internet'99*, pages 1808–1817, December 1999.

[V.88] Jacobson V. Congestion avoidance and control. In *Proceedings of the SIGCOMM '88 Symposium*, pages 314–329, August 1988.

[VBB00] Milan Vojnovic, Jean-Yves Le Boudec, and Catherine Boutremans. Global fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip times. In *Proceedings of IEEE INFOCOM*, pages 1303–1312, 2000.

[Wro97a] J. Wroclawski. *Specification of the Controlled-Load Network Element Service*, September 1997. RFC 2211.

[Wro97b] J. Wroclawski. *The Use of RSVP with IETF Integrated Services*, September 1997. RFC 2210.

[YR01] Ikjun Yeom and A. L. Narasimha Reddy. Modeling TCP behavior in a differentiated services network. *IEEE/ACM Transactions on Networking*, 9(1):31–46, 2001.